Processing of Erroneous and Unsafe Data

Verwerking van foutieve en onveilige data

Proefschrift

ter verkrijging van de graad van doctor aan de
Erasmus Universiteit Rotterdam
op gezag van de
Rector Magnificus

Prof.dr.ir. J.H. van Bemmel

en volgens besluit van het College voor Promoties.

De openbare verdediging zal plaatsvinden op

donderdag 19 juni 2003 om 16:00 uur

door

Anthonie Geradus de Waal

geboren te Den Haag

Promotiecommissie

Promotor: Prof.dr.ir. R. Dekker

Overige leden: Prof.dr. J.G. Bethlehem
Prof.dr.ir. P. Kooiman
Prof.dr. A.P.M. Wagelmans

Erasmus Research Institute of Management (ERIM)
Rotterdam School of Management / Rotterdam School of Economics
Erasmus University Rotterdam

Internet: http://www.erim.eur.nl

ERIM Electronic Series Portal: http://hdl.handle.net/1765/1

ERIM Ph.D. Series Research in Management 24

ISBN 90 – 5892 – 045 – 3

Tiger got to hunt,
Bird got to fly;
Man got to sit and wonder, "Why, why, why?"

Tiger got to sleep,
Bird got to land;
Man got to tell himself he understand.

Kurt Vonnegut, *Cat's Cradle*

Evelyn, a dog, having undergone
Further modification
Pondered the significance of short-person behaviour
In pedal-depressed panchromatic resonance
And other highly ambient domains…
Arf, she said

Frank Zappa, *Evelyn, A Modified Dog*

To Sandra and Anouchka

Acknowledgements

During my period as a student at the University of Leiden, my interest in mathematics dropped to an all-time low. After my graduation I even promised myself that I would never write a Ph.D. thesis on mathematics. As one can see from this book, I broke my promise. There are several people to blame for this. I would like to thank those people.

In particular, I would like to thank:

Peter Kooiman for giving me the opportunity to work at Statistics Netherlands.

Leon Willenborg for introducing me to statistical disclosure control and Frank van de Pol for introducing me to statistical data editing.

My co-authors Leon Willenborg, Jeroen Pannekoek and Ronan Quere for the opportunity to work with them so closely.

Many colleagues and students for the opportunity to work with them on either statistical data editing or statistical disclosure control, and/or for stimulating me to write this thesis.

Rommert Dekker for carefully reading several draft versions and his valuable suggestions to improve the thesis.

The members of the review committee, Jelke Bethlehem, Peter Kooiman, and Albert Wagelmans, for their comments on an earlier version of the thesis.

Lieneke Hoeksma and Stephen Smith for their corrections on the text. Needless to say all remaining mistakes in this thesis are entirely my own fault.

Statistics Netherlands for granting me the time and the opportunity to work on my thesis.

The Erasmus Research Institute of Management for printing this thesis in their Research in Management series.

My brothers and sisters, especially my brothers, Jan and Wim, who volunteered to be my *paranimfen*.

Most of all, I would like to thank Sandra and Anouchka for their love and their presence in my life.

Contents

1. Introduction and Summary

1.1. Introduction

The process of producing statistical information as practised at national statistical institutes can be broken down into a number of steps. Willeboordse (1998), for instance, distinguishes the following phases in the statistical process for business surveys, and a similar division can be made for social surveys:

- setting survey objectives;
- form design and sampling design;
- data collection and data entry;
- data processing and data analysis;
- publication and data dissemination.

Each phase itself can be subdivided into several steps. When the surveys objectives are set, user groups for the statistical information under consideration are identified, user needs are assessed, available data sources are explored, potential respondents are contacted, the survey is embedded in the general framework for business surveys, the target population and the target variables of the intended output are specified, and the output table is designed.

When the form is designed and sampling design is constructed, the potential usefulness of available administrative registers is determined, the frame population in the so-called Statistical Business Register is compared with the target population, the sampling frame is defined, the sampling design and estimation method are selected, and the questionnaire is designed.

During the data collection and data entry phase, the sample is drawn, data are collected and are entered into the computer system at the statistical office. During this phase the statistical office tries to minimise the response burden for businesses and to minimise non-response. There is a decision process on how to collect the data: paper questionnaires, personal interviews, telephone interviews, or electronic data interchange.

In the processing and analysis phase the collected data are edited, missing and erroneous data are imputed, raising weights are determined, population figures are estimated, the data are incorporated in the integration framework, and the data are analysed (for example to adjust for seasonal effects).

The publication and dissemination phase includes setting out a publication and dissemination strategy, protecting the final data (both tabular data and microdata, i.e. the data of individual respondents) against disclosure of sensitive information, and lastly publication of the protected data.

This book examines two different, but related, subjects mentioned above. The first is statistical data editing, which takes places during the processing and analysis phase. The aim of statistical data editing is to identify errors in the data and to correct them. To achieve this the data are enriched with the aid of subject-matter knowledge and statistical modelling techniques. That is, we try to create more information than we have observed. The second subject is statistical disclosure control, which takes place during the publication and data dissemination phase to prevent sensitive information about individual respondents, or small groups of respondents being disclosed in the published data. To achieve this, data are deleted, or the information contained in the data is reduced by adding noise or by recoding variables. In other words, we try to reduce the information content of the data.

On the surface, statistical data editing and statistical disclosure control seem to be opposed - and therefore unrelated - subjects. At a deeper level, however, the two subjects are related. In both areas we try to retain as much information as possible. The main difference between the two areas is formed by the constraints that have to be satisfied in order to achieve this. In statistical data editing errors are usually detected by means of certain rules, referred to as edits. Often it is assumed that a minimum number of errors occur in the observed data. Given this assumption, we then try to change as few values as possible such that all edits become satisfied. On the other hand, in statistical disclosure control we often assume that safe microdata should satisfy certain frequency count rules. We then try to delete as few observed values as possible such that the frequency count rules become satisfied. In both cases we are faced with a mathematical optimisation problem. In statistical disclosure control this particular optimisation problem can be formulated as a set-covering problem, in statistical data editing the corresponding optimisation problem can partly be solved by means of set-covering algorithms.

The similarity between the mathematics of both subjects becomes even clearer in Chapters 8 and 20 of this book. While in Chapter 8 Fourier-Motzkin elimination is used to detect errors in data, in Chapter 20 it is applied to produce safe tabular data. In fact, in Chapter 20 we use Fourier-Motzkin elimination to generate so-called elementary aggregations, a fundamental concept in disclosure control of tabular data. In statistical data editing parlance, such an aggregation would be referred to as an implicit edit, a fundamental concept in automatic data editing. We briefly discuss statistical data editing and statistical disclosure control in Sections 1.2 and 1.3, respectively.

1.2. Statistical data editing

In order to make well-informed decisions, managers, politicians and other policy makers need high quality statistical information about social, demographic, industrial, economic, financial, political, and cultural aspects of society. National statistical institutes (NSIs) fulfil a very important role in providing such statistical information. The task of NSIs is considerably complicated by the rapid changes in present-day society. Moreover, the power of modern computers also enables end-users to process and analyse huge amounts of statistical information themselves. As a result, these users are demanding statistical information with a greater level of detail and higher quality. To fulfil their role successfully NSIs also need to produce these high-quality data within a shorter span of

time. Most NSIs face these challenges while their financial budgets are constantly diminishing.

It is difficult to produce high-quality data in a short space of time. A major complicating factor is that collected data generally contain errors. The data collection stage in particular is a potential source of errors. A lot can go wrong in the process of asking questions and recording the answers. For instance, a respondent may give a wrong answer (intentionally or not), errors can occur at the statistical office when the data are transferred from the questionnaire to the computer system, etc. The presence of errors makes it necessary to carry out an extensive data editing process of checking the collected data, and when necessary, correcting them.

Traditionally, statistical agencies have always put a lot of effort and resources into data editing, as they considered it a prerequisite for publishing accurate statistics. In traditional survey processing, data editing was mainly an interactive activity intended to correct all data in every detail. Detected errors or inconsistencies were reported and explained on a computer screen and corrected after consulting the questionnaire, or contacting respondents: time and labour-intensive procedures.

It has long been recognised, however, that it is not necessary to correct all data in every detail. Several studies (see for example Granquist, 1984; Granquist, 1997; Granquist and Kovar, 1997; Houbiers, 1999a; Houbiers, Quere and De Waal, 1999) have shown that in general it is not necessary to remove all errors from a data set in order to obtain reliable publication figures. The main products of statistical offices are tables containing aggregate data, which are often based on samples of the population. This implies that small errors in individual records are acceptable. First, because small errors in individual records tend to cancel out when aggregated. Second, because if the data are obtained from a sample of the population there will always be a sampling error in the published figures, even when all collected data are completely correct. In this case an error in the results caused by incorrect data is acceptable as long as it is small in comparison to the sampling error. In order to obtain data of sufficiently high quality it is usually enough to remove only the most influential errors. The above-mentioned studies have been confirmed by many years of practical experience at several statistical offices.

In the past, and often even in the present, too much effort was spent on correcting errors that did not have a noticeable impact on the ultimately published figures. This has been referred to as “over-editing”. Over-editing not only costs money, but also a considerable amount of time, making the period between data collection and publication unnecessarily long. Sometimes over-editing even becomes “creative editing”: the editing process is then continued for such a length of time that unlikely, but correct, data are “corrected”. Such unjustified alterations can be detrimental for data quality. For more about the danger of over-editing and creative editing see for example Granquist (1995), Granquist (1997) and Granquist and Kovar (1997).

The ever-increasing power of modern computers not only provides challenges for NSIs, but also the opportunities to solve them. One of the solutions for the earlier mentioned challenge of providing huge amounts of high-quality data in a short space of time is to improve the traditional editing and imputation process. Well-known, although still relatively modern, techniques such as selective editing, (graphical) macro-editing, and

automatic editing can be applied instead of the traditional interactive approach. Selective editing entails dividing the data in two: one part that probably contains influential errors and one that is unlikely to contain influential errors. Subsequently, only the former part is edited using the traditional interactive approach. In macro-editing, the plausibility of aggregates is checked. Only when aggregates are not plausible are the corresponding data edited in the traditional manner. The most drastic modern editing technique is automatic editing. It is the opposite of the traditional approach to the editing problem, where each record is edited manually. With automatic editing the records are edited entirely by means of a computer. These modern techniques take less time and cost less than the traditional interactive method, while preserving or often even increasing quality. More details about these techniques can be found in Chapter 2.

In principle, automatic editing offers great prospects in terms of saving time and money. It is one of the main subjects of this book. In fact, we concentrate on one important and complicated aspect of automatic data editing, namely the problem of localising the erroneous (or more precisely: implausible) values in the data. Our aim is to develop a general algorithm for solving this error localisation problem that can be applied to a wide range of data sets without requiring specific subject-matter knowledge other than a specification of the edits that correct data have to satisfy. Such an algorithm should be able to solve the error localisation problem for data sets involving thousands of records and several dozens of complex edits within a few hours at most. Finding solutions to the error localisation problem for a set of complex edits amounts to solving a difficult combinatorial puzzle for which techniques from operations research can be used. While solving this puzzle, one should not forget that the determined solutions should be useful from a statistical point of view. In order to solve the error localisation problem successfully, therefore, expertise on two different areas is required: operations research and statistics. We examine the error localisation problem in detail in Chapters 2 to 13 of this book. Section 1.4 contains a brief summary of these chapters.

Literature on the error localisation problem is scarce. In fact, the only articles in scientific journals we are aware of are: Freund and Hartley (1967), Fellegi and Holt (1976), Garfinkel, Kunnathur and Liepins (1986 and 1988), McKeown (1984), Schaffer (1987), and Ragsdale and McKeown (1996).

1.3. Statistical disclosure control

The other subject of this book is statistical disclosure control, one of the last steps in the statistical process. On the one hand, it is the duty of statistical offices to release as much statistical information as possible. On the other hand, they are also bound by legal restrictions and moral obligations to protect the information of individual respondents. In particular, it should be practically impossible for potential intruders to disclose confidential data of individual respondents or small groups of respondents. It is usually difficult to provide absolute protection, as in many cases this would entail that hardly any information may be released. Statistical disclosure control therefore aims to limit the risk of disclosure to an acceptable level.

The main challenge of statistical disclosure control is to put oneself in the shoes of a potential intruder trying to disclose sensitive information. Talking about the famous chess

grandmaster Tigran Petrosjan, Bobby Fischer once said that he had the ability to prevent potential attacking possibilities of his opponent long before these possibilities had even entered his opponent's head. We are trying to do the same as Petrosjan. We are trying to protect sensitive information against potential attacks by potential intruders before such potential intruders have even realised that such attacks might be possible. As many potential attack paths are possible, statistical disclosure control obviously requires expertise in many different areas, of which again operations research and statistics are the most important.

Statistical offices release different data in various formats. Two well-known formats are microdata sets and tabular data. Microdata sets consist of data on individual respondents. These microdata have been anonymised, i.e. personal identifiers such as name and address, are not released. In addition, various disclosure control techniques are applied to further limit the risk of disclosure, as respondents may be recognisable from other data that are released. Tabular data are aggregate data released in the form of tables. Like microdata aggregate data have to be protected against disclosure. Of course, tabular data do not contain personal identifiers, but other information might enable a potential intruder to identify a respondent and misuse the information.

There is more literature on statistical disclosure control for tabular data than on disclosure control for microdata. To fill this void, we focus on statistical disclosure control for microdata in this book. In contrast to the first part of this book where we basically concentrate on one problem of statistical data editing, the error localisation problem, we examine various aspects of statistical disclosure control in the second part of this book. Examples are: protection of microdata in general, synthetic and combined estimators to estimate population frequencies, and an optimisation model to minimise the number of suppressed values in a protected microdata set. A brief summary of the chapters on statistical disclosure control is given in Section 1.4 below.

There is quite a lot of literature on statistical disclosure control. Four books have been published: Willenborg and De Waal (1996 and 2001), Doyle et al. (2001), and Domingo-Ferrer (2002). At least two Ph.D. theses have been written on statistical disclosure control (see Sullivan, 1989, and Kelly, 1990). Lastly, several overview articles on this area have been published (see e.g. Duncan and Lambert, 1986, and Fienberg, 1994), and special issues of journals have devoted to the subject (cf. the 1993 special issue of the *Journal of Official Statistics* on confidentiality and data access).

1.4. Summary

The first part of this book, Chapters 2 to 13, concentrates on data editing, in particular on the so-called error localisation problem. In Chapter 2 we consider the data editing problem, i.e. the problem of cleaning up statistical data, in general. We also consider some techniques for (efficiently) solving this problem. One of the techniques we mention in this chapter is automatic editing, the editing technique we focus on in subsequent chapters.

As mentioned earlier, we concentrate on one aspect of automatic data editing, namely the error localisation problem. This problem is usually considered either for categorical data, i.e. discrete data without any arithmetic structure such as "Gender" or "Profession", or for continuous data in literature. This book is an exception: we consider the error localisation

problem for a mix of categorical and continuous data, a problem that occurs quite often in practice. A precise mathematical formulation of the error localisation problem for a mix of categorical and continuous data is given in Chapter 3. This mathematical formulation is based on simple concepts from mathematical logic. In particular, the formulation uses the implication operator (IF/THEN statement). We are the first to give such a formulation of the error localisation problem, which seems more natural than a mixed integer programming formulation. In the same chapter we also formulate the error localisation problem as a mixed integer programming problem. In principle, this mathematical optimisation problem can be solved by a solver for mixed integer programming problems. A drawback of such a solver is, however, that it usually determines only one optimal solution to the mathematical optimisation problem, whereas we would like to determine all optimal solutions so we can choose one of them according to an additional criterion. In order to determine all optimal solutions by means of a commercial solver for mixed integer programming problems special steps need to be taken. It remains to be examined whether available solvers for mixed integer programming problems are sufficiently fast to determine all optimal solutions for practical instances of the error localisation problem.

Chapters 4 to 10 examine several dedicated methods for solving the error localisation problem. The motivation for examining so many methods stems from the significance of efficient editing of business survey data in particular for Statistics Netherlands. For purely numerical data and non-conditional linear edits, we first studied the methods described in Chapters 4 and 5, which are known from available reports. We decided to implement the method of Chapter 5 for such kinds of data and edits in our production software. Later, the wish to develop algorithms for more general data and edits slowly grew. The first attempts to develop such algorithms were unsuccessful. We tried to extend the algorithm of Chapter 5 to include categorical data, which led to a rather complicated algorithm; too complicated for our purposes. We subsequently examined and developed the methods of Chapters 6, 7, and 8. We were not satisfied with the algorithms of Chapters 6 and 7. The algorithm of Chapter 8 was considered satisfactory. We extended that algorithm to include integer-valued data (see Chapter 9). Lastly, we decided to compare the algorithm of Chapter 8 with cutting plane algorithms similar to cutting plane algorithms known from the literature. This resulted in Chapter 10.

Chapter 4 describes a method developed by Fellegi and Holt (1976) based on generating so-called implicit edits. This chapter hardly contains new results, except for a simple proof that the method works for numerical data and the brief remark that the method can, in principle, be applied to a mix of categorical and numerical data.

Chapter 5 describes vertex generation methods. These methods, in particular a method by Chernikova (1964, 1965) have been applied to solve the error localisation problem in continuous data. In the literature the possibility of extending the methods to a mix of categorical and continuous data is briefly mentioned (Sande, 1978a), although no details are given. In Chapter 5 we provide such details, and show that many results for continuous data also hold true for a mix of categorical and continuous data. We also mention that vertex generation methods other than Chernikova's algorithm may be used to solve the error localisation problem.

Chapter 6 describes the error localisation problem as a so-called dynamic disjunctive-facet problem. Glover, Klingman and Stutz (1974) describe a cutting plane algorithm for solving

the disjunctive-facet problem. Originally, we tried to formulate the error localisation problem as a disjunctive-facet problem, but this turned out to be not completely possible. We had to extend the disjunctive-facet problem to a *dynamic* disjunctive-facet problem. The cutting plane algorithm for solving the disjunctive-facet problem had to be extended accordingly. Moreover, we show that the algorithm of Glover, Klingman and Stutz does not necessarily terminate after a finite number of steps. We extend our algorithm so it is guaranteed to be finite.

Chapter 7 describes a method developed by Pergamentsev (1998) based on local search heuristics. Pergamentsev was a post-graduate student at Eindhoven University of Technology who worked on the error localisation problem at Statistics Netherlands. In Chapter 7 we propose several improvements on his method.

Chapter 8 describes a branch-and-bound algorithm for solving the error localisation problem for a mix of categorical and continuous data. Quere, another post-graduate student from Eindhoven University of Technology who worked on the error localisation problem at Statistics Netherlands, first developed a similar branch-and-bound algorithm for the error localisation problem for continuous data. Together with Quere we later extended this algorithm to a mix of categorical and continuous data. This extended algorithm is the subject of Chapter 8.

In Chapter 9 we extend the problem formulated in Chapter 3 to include integer data. We propose an extension of the algorithm of Chapter 8 in order to solve it. The developed algorithm uses Fourier-Motzkin elimination in integer-valued data as developed by Pugh (1992). Our algorithm efficiently combines Fourier-Motzkin elimination in integer-valued data with the algorithm of Chapter 8.

Chapter 10 starts by describing a cutting plane algorithm based on algorithms due to Garfinkel, Kunnathur and Liepins (1986 and 1988). The original algorithms developed by Garfinkel, Kunnathur and Liepins were designed for categorical and continuous data respectively only. In Chapter 10 we extend these algorithms to a mix of categorical and continuous data. Subsequently, we use the theory of Chapters 8 and 9 to improve on these cutting plane algorithms and a similar cutting plane algorithm by Ragsdale and McKeown (1996). Finally, we extend the improved algorithm to a mix categorical, continuous and integer data.

Computational results for the algorithms of Chapter 3 (based on a standard mixed integer programming formulation), Chapter 5 (based on vertex generation), Chapter 8 (based on branch-and-bound), and Chapter 10 (based on cutting planes) on several numerical data sets are presented in Chapter 11.

Chapter 12 considers imputation, and in particular the problem of consistent imputation, i.e. the problem of filling in values for erroneous and missing data in such a way that the resulting data are both acceptable from a statistical point of view and internally consistent. We propose an algorithm, similar to the algorithm proposed in Chapters 8 and 9, for solving the problem of consistent imputation. We developed the basic ideas of the algorithm presented in Chapter 12. Later a post-graduate student from Delft University of Technology, Kartika, implemented the algorithm and in the course of that work filled in many implementation details. In the same chapter we also briefly describe WAID, imputation software we have developed in a European project.

Chapter 13 concludes the part of this book on statistical data editing by considering some practical issues of error localisation. The final section of that chapter also briefly discusses the impact of the developed methodology on the daily practice at Statistics Netherlands.

The error localisation problem is related to many other problems, both mathematical and non-mathematical. Examples of related mathematical problems are: the imputation problem (or: how to estimate missing or erroneous data), the outlier detection problem (how to identify univariate or multivariate outliers in statistical data) and the weighting problem (how to calculate raising weights). Examples of non-mathematical problems are: the logistics of handling large amounts of data, and the ICT problem of developing computer programs for data editing as part of the software suite for the entire survey process. Such problems are for the main part outside the scope of this book.

The second part of this book, Chapters 14 to 20, concentrates on statistical disclosure control. Chapter 14 gives an overview of the area, focusing on a general approach to statistical disclosure control for microdata of mainly social surveys applied at several statistical offices, including Statistics Netherlands. According to this approach, statistical disclosure control rules for microdata for social surveys should primarily be based on requiring minimum population frequencies for certain characteristics. The subsequent chapters elaborate on this general approach.

Chapter 15 examines the worst possible populations from a statistical disclosure point of view. Such populations give rise to the highest possible expected number of unique individuals in randomly selected samples. We are the first to propose this problem, and a solution to it.

Population frequencies of characteristics are usually unknown. Standard statistical methods can be applied to estimate them for large areas. To estimate frequencies for small areas, however, one has to resort to small area estimation techniques: synthetic and combined estimators, for example. In Chapter 16 we try to estimate population frequencies for small areas by means of such synthetic and combined estimators. The estimators of Chapter 16 have been developed and tested in conjunction with Pannekoek.

If the population frequency of a certain characteristic is below the required minimum, it is considered too unsafe for release, and may not be released as such. In such a case, measures should be taken to protect respondents with this characteristic. One possible technique is suppression, where the characteristic, or part of it, is deleted. As a statistical office aims to publish as much information as is legally and morally permissible, we try to minimise the number of suppressed values subject to the constraint that the resulting data set is considered safe. In collaboration with Willenborg we have developed optimisation models for a number of such problems. The mathematical formulations of these optimisation models are presented in Chapter 17.

Apart from suppression, there are other statistical disclosure control techniques to limit the risk of disclosure: recoding and perturbation, for example. All these techniques, i.e. suppression, recoding and perturbation, lead to a certain loss of information. As our goal is to publish as much information as possible, subject to the condition that the released data are sufficiently protected, we want to compare the amounts of information lost as a result of the various techniques. Chapter 18 offers a general framework based on entropy to make such comparisons. This framework has again been developed jointly with Willenborg.

Release of sampling, or better: raising, weights together with the corresponding microdata set requires some additional attention. These weights may enable a potential intruder to extract more information from the released data than is permitted under the disclosure control rules. Chapter 19 describes this danger of releasing sampling weights, and suggests methods to overcome it. Once again, the material in Chapter 19 has been developed together with Willenborg.

Lastly, Chapter 20 concludes with a description of the so-called cell suppression problem in tabular data. In particular, the chapter points out that the usually applied definition of a safe table is inconsistent. We propose an alternative for this inconsistent definition, based on so-called elementary aggregations. The use of elementary aggregations to determine whether a table is safe or not was already suggested by Sande in the late 70's (see, e.g., Sande, 1977). Sande's concept of a safe table is not always correct, however. We have corrected the flaw in his concept of a safe table, and we demonstrate that our definition does not lead to suppressed cells that can be re-calculated (too) precisely.

Chapter 20 is a good conclusion to this book as it returns to concepts from the first part, such as implicit edits and Fourier-Motzkin elimination. Elementary aggregations can be considered as implicit edits; and they can be derived by means of Fourier-Motzkin elimination. In a sense, Chapter 20 demonstrates that studying statistical disclosure control is essentially the same as studying automatic error localisation.

Several parts of this book have previously been published as articles in scientific journals. We indicate this in the relevant chapters.

2. Statistical Data Editing

2.1. Introduction

In this chapter we discuss several data editing techniques, ranging from computer-assisted editing to automatic editing. The latter technique, in particular the problem of automatically identifying erroneous (or more precisely: implausible) values in the data, is one of the main topics of this book.

In Section 2.2 we discuss how data are edited in general. In particular we explain the use of edit rules. The advantages and disadvantages of computer-assisted editing are discussed in Section 2.3. Section 2.4 gives a global description of the modern editing techniques selective editing and (graphical) macro-editing. A more detailed discussion of another modern editing technique, automatic editing, follows in Section 2.5. Section 2.6 concludes the chapter with a short discussion of editing techniques and how they could be combined.

For a more detailed overview of statistical data editing we refer to Ferguson (1994), and Hoogland, Houbiers and De Waal (2002).

2.2. Statistical data editing in general

Errors in data can be detected by specifying certain constraints that have to be satisfied by the individual *records*, i.e. data of individual respondents. These constraints are called *edit rules* (or *edits* for short). They are specified by subject matter specialists. When a record fails an edit, it is considered erroneous. When a record satisfies all edits, it is considered correct. The values in an erroneous record have to be modified in such a way that the resulting record is a better approximation of the true data of the corresponding respondent.

At Statistics Netherlands editing of business surveys is a much bigger problem than editing of social surveys on households and persons. Business surveys mainly consist of numerical data. Typically, business surveys in the Netherlands are not very large. Large and complicated surveys may have somewhat over 100 variables and 100 edits. The number of records in a business survey is usually a few thousand. In countries where in contrast to the Netherlands a census on persons is held, editing census data is generally a big problem. Census data are mostly categorical data. They do not contain a high percentage of errors, but the number of edits (a few hundred), number of variables (a few hundred), and especially the number of records can be high (several millions).

To modify an erroneous record two steps have to be carried out. First, the incorrect values in such a record have to be localised. This is called the *error localisation problem*. Second, after the faulty fields in an erroneous record have been identified these faulty fields have to be *imputed*, i.e. the values of these fields have to be replaced by better, preferably the correct, values. The error localisation problem must be solved in such a way that the fields that are considered faulty can indeed be imputed consistently, i.e. that these fields can be imputed in such a way that the resulting modified record satisfies all edits.

The error localisation problem is traditionally solved by humans, possibly assisted by a computer. Examples of methods that are used to solve the error localisation problem are: re-contacting the respondent, comparing the respondent's data to his data from previous years, comparing the respondent's data to data from similar respondents, and using subject-matter knowledge. The major drawbacks of the traditional approach are that it is time and money consuming.

For erroneous records the error localisation problem and the imputation problem are closely related. Often it is hard to distinguish where the error localisation phase ends and where the imputation phase starts. When humans edit data, they frequently look at possible ways of imputing a record before completing the error localisation phase. So, for erroneous records the imputation problem is traditionally also solved by humans, again possibly assisted by a computer. Methods are similar to "manual" methods for solving the error localisation problem, namely re-contacting the respondent, using the respondent's data from previous years to impute for erroneous data from this year, using data from similar respondents to impute for erroneous data, and using subject-matter knowledge.

For records for which "only" some values are missing, but that otherwise are correct, the imputation is traditionally solved automatically by means of a computer program. For those records there is no interaction with the error localisation phase, so no human intervention is deemed necessary.

2.3. Computer-assisted editing

2.3.1. The mainframe age

The use of computers in the editing process started many years ago. In the early years their role was, however, restricted to checking which edits were violated. Subject-matter specialists entered data into a mainframe computer. Subsequently, the computer checked whether these data satisfied all specified edits. For each record all violated edits were listed. Subject-matter specialists then used these lists to correct the records. That is, they retrieved all paper questionnaires that did not pass all edits and corrected these questionnaires. After they had corrected the data, these data were again entered into the mainframe computer, and the computer again checked whether the data satisfied all edits. This iterative process continued until (nearly) all records passed all edits.

A major problem of this approach was that during the manual correction process the records were not checked for consistency. As a result, a record that was "corrected" could still fail one or more specified edits. Such a record hence required more correction. It was not exceptional that some records had to be corrected four times in this way. It is therefore not surprising that editing in this way was very costly, both in terms of money as well as in terms of time. In literature it was estimated that 25 to 40 per cent of the total budget was spent on editing (see e.g. Federal Committee on Statistical Methodology, 1990; Granquist and Kovar, 1997).

2.3.2. Interactive editing in the PC age

The introduction of PC's and computer systems such as Blaise, the integrated survey-processing system developed by Statistics Netherlands (see e.g. *Blaise Reference Manual*, 2002, and *Blaise Developer's Guide*, 2002), led to a substantial efficiency improvement of the editing process. The potency of Blaise is that during data entry the consistency of the entered data is checked. It is no longer necessary to edit the data in several iterations, consisting of a checking phase and a correction phase. When a system such as Blaise is used, checking and correction can be combined into one step. After this step the data are consistent.

Blaise not only applies edits, but also so-called routing rules. Frequently, different questions are posed to different kinds of respondents. For instance, it is not useful to ask a male respondent whether he has ever been pregnant as the answer to this question would not provide any additional information. A system like Blaise ensures that each respondent is asked the questions that are applicable to this kind of respondent. Due to this functionality Blaise is an excellent system for CAPI (Computer Assisted Personal Interviewing) and CATI (Computer Assisted Telephone Interviewing). When CAPI is used to collect the data, an interviewer visits the respondent and enters the answers directly into a laptop. When CATI is used to collect the data, the interview is carried out during a telephone call. When an inconsistency between the answers of two or more questions is noted, this is reported by Blaise. The error can then be resolved during the interview by asking the respondent these questions again. Data collected by means of CAPI, and to a slightly lesser extent by CATI, therefore hardly contain errors. CAPI and CATI may hence seem to be ideal ways to collect data, but – unfortunately – they too have their disadvantages.

The first disadvantage is that CAPI especially is very expensive. Sending a paper questionnaire to a respondent is much cheaper than sending an interviewer to a respondent.

A second, very important, disadvantage is that CAPI and CATI are much less suited for surveys on enterprises than for surveys on persons and households. A prerequisite for CAPI and CATI is that the respondent is able to answer the questions during the interview. For a survey on persons and households, this is often the case. The respondent knows the answers to the questions, or is able to retrieve these answers quickly. For a survey on enterprises, the situation is quite different. Often it is impossible to retrieve the correct answers quickly, and often the answers are not even known by one person or one department of an enterprise. Frequently, data from different departments have to be collected in order to answer a questionnaire from a national statistical office. Finally, even in the exceptional case that one person knew all answers to the questions, the statistical office would generally not know the name of this person. So, it would be very likely that the statistical office would interview the wrong person.

For the above-mentioned reasons Statistics Netherlands, and many other national statistical institutes, frequently use CAPI and CATI to collect data on persons and household but only rarely for data on enterprises.

An alternative approach to collect data on, for example, enterprises is to use the Internet. Enterprises can then fill in an electronic questionnaire and send it electronically to the statistical office as soon as the questionnaire has been completed. Part of the edits can be

checked while the respondent is filling in the questionnaire. When answers are detected to be inconsistent, the respondent can immediately be prompted to fill in other answers. Using the Internet to collect data is potentially a very attractive approach, and it has therefore been tested in several pilot studies. These pilot studies have revealed that collecting data via the Internet may indeed be a very useful option in future, but also that many problems have to be solved before it is fully operational in practice. A few technical problems with data collection via the Internet are:

- the software - and the Internet – should be fast and reliable;
- the security of the transmitted data should be guaranteed;
- the underlying software should be flexible enough in order to allow respondents to fill in part of their answers, and continue filling in their answers at a later time.

An example of a non-technical problem is that sometimes a respondent simply does not know the answer to a certain question. If the edits keep on reporting that the answer is incorrect, the respondent may get annoyed and refuse to fill in the questionnaire anymore.

Although a system like Blaise is rarely used to collect data on enterprises, it is often used to enter data on enterprises into the computer system at statistical offices. During data entry (part of) the specified edits are then immediately checked.

Data on persons and households collected by Statistics Netherlands hardly contain errors because for these data CAPI or CATI is used. Data on enterprises, however, do contain a lot of errors, because these data frequently are obtained by means of paper questionnaires. The emphasis in the remainder of this book will therefore be on editing of economic, i.e. mainly numerical, data.

Computer-assisted editing is nowadays the standard way to edit data. It can be used to edit both categorical and numerical data. The number of variables, edits and records may, in principle, be high. Given a flexible system such as Blaise, it is easy to compare data from the current period to data from a previous period. Generally, the quality of data editing in a computer-assisted manner is considered high.

As we already mentioned in Section 2.2, when computer-assisted editing is used to clean data, imputation is usually carried out by re-contacting the respondent, by comparing the respondent's data to his data from a previous period, by comparing the respondent's data to data of similar respondents, or by using subject-matter knowledge to estimate values for erroneous fields.

Interactive editing of data, using PC's and a system like Blaise, has the major advantage in comparison to editing in the mainframe age that checking and correction is done at the same time. Each record has to be edited only once, after which it satisfies all edits. The fundamental problem of editing in this way is that, even though each record has to be edited only once, still *all* records have to be edited. In Chapter 1 we have already mentioned that this can, and usually does, lead to "over-editing" and even to "creative editing". In the following section we consider ways to limit computer-assisted editing to only those records that are likely to contain the most influential errors.

2.4. Modern editing techniques

In this section we describe two editing techniques that aim to identify the most influential errors quickly. This implies that they are more appropriate for numerical data than for categorical data, because it is more natural to define the term "most influential" for numerical data than for categorical data. An error in numerical data is influential if the difference between the recorded value and the actual value is large. It only remains to define what is meant by "large". In categorical data, however, there often is no concept of a large deviation between the recorded data and the actual data.

On the other hand, although the techniques we describe in this section are more appropriate for numerical data than for categorical data, some versions of these techniques are still applicable to categorical data.

2.4.1. Selective editing

We have already mentioned in Section 1.2 that it is not necessary for each individual record to be absolutely correct. Statistical offices publish aggregate data often based on samples of the population, hence small errors in individual records are acceptable as long as the raising weights of these records are not very high compared to the weights of other records.

Selective editing is an umbrella term for several methods to identify the influential errors, i.e. the errors that have a substantial impact on the publication figures. The aim of selective editing is to split the data into two streams: the critical stream and the non-critical stream. The critical stream consists of records that are the most likely ones to contain influential errors; the non-critical stream consists of records that are unlikely to contain influential errors. The records in the critical stream, the critical records, are edited in a traditional computer-assisted manner. The records in the non-critical stream, the non-critical records, are not edited in a computer-assisted manner. They may later be edited automatically.

At present no accepted theory for selective editing methods exists. Consequently, most of these methods are simple ad-hoc methods based on common sense. It is hardly possible to describe here all selective editing methods that have been developed over the years, nor is this necessary to understand the rest of this book. We only mention here that a score function or OK-index is often used to split the records into the critical stream and non-critical stream (see e.g. Hidiroglou and Berthelot; 1986; Van de Pol and Molenaar, 1995). Such a score function or OK-index measures the distance between the recorded values and "expected" values. "Expected" values are, for example, medians in certain groups of records or values from a previous year. Besides, score functions and OK-index measures usually also take the sampling weight and "importance" of a record into account.

Selective editing is a relatively new technique. It is gradually becoming a popular method to edit business (numerical) data. Increasingly more statistical offices use selective editing to clean their data, or experiment with selective editing.

The scope of selective editing is limited to business (numerical) data. In business (numerical) data some respondents can be more important than other respondents, simply because the magnitude of their contributions are higher. Social (categorical) data are count data where respondents contribute more or less the same, namely their raising weight, to

the estimated population total. In social data it is therefore difficult to differentiate between respondents. In business data such a differentiation is much easier to provide.

In selective editing data from the current period can easily be compared to data of a previous period. There is no limit on the number of edits. A problem of selective editing at the moment is that the number of variables may not be too high. At present, good techniques are not available to combine local scores, for example, those based on the distance between the recorded value and the expected value for each variable, into one global score for each record if there are many variables. This may be a mere technical problem that will be solved in due course, but it may also be a fundamental problem of selective editing. More research is needed to answer this question.

2.4.2. (Graphical) macro-editing

Macro-editing offers a solution to some of the problems of micro-editing. Particularly, macro-editing can deal with editing tasks related to the distributional aspect. We distinguish between two forms of macro-editing. The first form is sometimes called the *aggregation method* (see e.g. Granquist, 1990). It formalises and systematises what every statistical agency does before publication: verifying whether figures to be published seem plausible. This is accomplished by comparing quantities in publication tables with the same quantities in previous publications. Examples of this form of macro-editing are the foreign trade surveys of the Netherlands (Van de Pol and Diederen, 1996) and Canada (Laflamme et al., 1996). Only if an unusual value is observed, a micro-editing procedure is applied to the individual records and fields contributing to the quantity in error.

A second form of macro-editing is the *distribution method*. The available data are used to characterise the distribution of the variables. Then, all individual values are compared with the distribution. Typically, measures of location and spread are computed. Records containing values that could be considered uncommon (given the distribution) are candidates for further inspection and possibly for editing.

There is an area in statistics providing all kinds of techniques for analysing the distribution of variables, namely *Exploratory Data Analysis* (EDA) (see e.g. Tukey, 1977). Many EDA techniques can be applied in macro-editing. Advocates of EDA stress the importance of the use of graphical techniques. These techniques provide much more insight in the behaviour of variables than numerical techniques do. This also applies to macro-editing. Graphs of the distribution of the data show a lot of information, and are capable of showing unexpected properties that would not have been discovered if just numerical quantities were computed. For more information on (graphical) macro-editing we refer to De Waal, Renssen and Van de Pol (2000).

Macro-editing has always been applied in some form at statistical offices. Non-graphical macro-editing techniques can be used for both business (numerical) data and social (categorical) data. The use of graphical techniques seems to be restricted to business (numerical) data.

In macro-editing, data from the current period can easily be compared to data of a previous period. There is no limit on the number of edits. A drawback of graphical techniques is that the number of variables may not be too high. For humans it is usually very difficult to

visualise points in a, say, 10-dimensional space. Discovering outliers in such a high-dimensional space in a visual manner is a non-trivial task.

2.5. Automatic editing

The aim of automatic editing is to let a computer do all the work. The main role of the human is to provide the computer with meta-data, such as edits, imputation models and rules to guide the error localisation process. After the meta-data have been provided, the computer edits the data and all the human has to do is examine the output generated by the computer. In case the quality of the edited data is considered too low, the meta-data have to be adjusted or some records have to be edited in another way.

In the 60's and early 70's automatic editing was usually based on predetermined rules of the following kind: if a certain combination of edits is violated in a certain way than a certain action has to be undertaken to correct the data. There are some problems with this deterministic approach. First, it is often too difficult to develop predetermined rules that ensure that the data satisfy all edits after they have been edited. This may lead to a complex iterative process where "edited" records that still fail some edits are again edited. Moreover, for some records this process may not converge.

Second, even if we do not aim to develop predetermined rules that lead to records that satisfy all edits, the set of predetermined rules will be very large and very difficult to handle. Basing a computer program on such a complex set of rules will be even more difficult.

Freund and Hartley (1967) propose an alternative approach based on minimising the total deviation between the original values in a record and the "corrected" values plus the total violation of the edits (the more an edit is violated, the more this edit contributes to the objective function). In this way only the edits had to be specified. The "corrected" values are subsequently determined by minimising a quadratic function. The approach by Freund and Hartley never became popular, probably because edits may – and often are – still be violated after correction of the data.

In 1976 Fellegi and Holt published a landmark paper in the *Journal of the American Statistical Association* that was a major breakthrough. In their paper Fellegi and Holt describe a paradigm for localising errors in a record automatically. According to this paradigm the data of a record should be made to satisfy all edits by changing the values of the fewest possible number of variables. In Chapter 3 we will discuss a generalised version of this paradigm. That generalised paradigm is the basis of several algorithms and computer programs for localising errors in records automatically. It is the de facto standard for modern automatic editing systems.

At Statistics Netherlands the first computer program for automatic editing based on the generalised Fellegi-Holt paradigm was CherryPi (see e.g. De Waal, 1996 and 1998b). CherryPi was designed for numerical data only, and could handle linear equalities and inequalities as edits. The range of the variables is not a priori restricted to non-negative values. Positivity, or more precisely non-negativity, of variables can be achieved by defining the corresponding edits. The algorithm on which CherryPi is based is explained in Chapter 5.

Automatic editing has already been used in the 60's and 70's. Nevertheless, it has never become really popular. For this several reasons can be pointed out. First, in former days computers were too slow to edit data automatically. Second, development of a system for automatic editing is considered too complicated and too costly by many statistical offices. Statistical offices often lack the expertise to develop and maintain such a system. Third, many statistical offices assume that data edited automatically are of unacceptably low quality. This is a serious point. It has been given ample attention in experiments on actual data, but more evaluation studies are required.

Automatic editing can be used for both categorical and numerical data. Most systems that have been developed are, however, suitable for either categorical data or numerical data. As far as the author is aware only Sande, Statistics Canada and Statistics Netherlands have developed systems that can handle both types of data simultaneously. The systems by Sande and Statistics Netherlands are based on the earlier mentioned Fellegi-Holt paradigm, the system by Statistics Canada on a slightly different paradigm. That latter system, NIM (see Bankier, 1999; Bankier et al., 2000), has originally been developed to edit demographic data automatically. It has later been extended to handle also numerical data to a limited extent. The system uses donor records to replace implausible values in erroneous records. The implausible values are found based on the edits and the available donor records.

In automatic editing data from the present period may be compared to data from a previous period. The simplest way to do this in, for example, CherryPi is to combine the record from a respondent in the present period with the record from the same respondent in a previous period into one large, combined record. The data from the previous period are set to "fixed" in CherryPi, which means that these data may not be changed during the automatic editing process. That record can then be edited in the normal way by CherryPi.

When automatic editing is used to clean the data, imputation may be carried out by a great variety of automatic imputation methods. Well-known classes of imputation models are regression imputation and donor imputation. Which (class of) method is best suited for a certain data set depends on the characteristics of the data set. For instance, for many business surveys regression imputation is a good option, and for many social surveys hot-deck donor imputation is a good option. In Chapter 12 we briefly discuss some aspects of automatic imputation.

The publicly available literature on automatic editing is quite limited. Most papers on automatic editing have only appeared as reports of statistical offices. Papers that are published as articles in mathematical and statistical journals include Freund and Hartley (1967), Fellegi and Holt (1976), Garfinkel, Kunnathur and Liepins (1986 and 1988), McKeown (1984), Schaffer (1987), and Ragsdale and McKeown (1996). The author of this book is not aware of other papers on automatic editing that are published as articles in scientific journals. Two papers by the author have been submitted for publication in statistical journals, namely De Waal (2003), and De Waal and Quere (2003). Finally, in the early 80's Liepins, who then worked at the Oak Ridge National Laboratory, wrote several papers on automatic editing (see Liepins, 1980, 1981, 1983 and 1984).

2.6. Discussion

In our view the ideal edit strategy is a combination of selective, automatic micro-editing and (graphical) macro-editing (see De Jong, 1996; Van de Pol and Bethlehem, 1997; Van de Pol et al., 1997; De Waal, Renssen and Van de Pol, 2000). After data entry, simple checks and simple (semi-automatic) corrections are applied. Examples of simple checks are range checks, examples of simple corrections are cases in which it is clear that a respondent filled in a financial figure in Euros instead of the requested thousands of Euros.

After that stage selective editing is applied to split the data into the critical stream and the non-critical stream. The critical records are edited in a computer-assisted manner, the non-critical records are either not edited or are edited automatically.

The latter approach, editing the records in the non-critical stream automatically, clearly has our preference. The sum of the errors in the non-critical records may have an influential impact on the publication figures, although each error itself is non-influential. Moreover, many non-critical records will be internally inconsistent. This may lead to problems when publication figures are calculated. Automatic editing helps to reduce the errors in the data, and makes sure that the records become internally consistent.

In our opinion, automatic editing should only be applied if just a few fields have to be changed in order to make a record pass all edits. We feel that records that require many changes should not be edited automatically. We regard the quality of such a record to be too low to allow for automatic correction. Such a record should either be edited manually, i.e. should be part of the critical stream, or be discarded completely.

In our view macro-editing should be used as a final check just before publication. Macro-editing cannot be missed, because it reveals errors that will go unnoticed with micro-editing.

We consider the combined use of selective editing, automatic editing and (graphical) macro-editing to be an efficient and effective way of cleaning data. In comparison with the traditional computer-assisted approach, the quality of the data can be maintained, while the resources needed to clean the data are substantially reduced and the timeliness of publication of the statistical data is clearly improved.

3. The Mathematical Error Localisation Problem

3.1. Introduction

In Section 3.2 of this chapter we give a detailed mathematical formulation of the error localisation problem for a mix of categorical and continuous data. In Section 3.3 we discuss a naive approach for solving the error localisation problem, and show that this approach does not work. Section 3.4 provides a mixed integer programming formulation for the problem. This formulation is similar to a formulation given by McKeown (1984) for purely continuous data, and a formulation by Schaffer (1987) for a mix of categorical and continuous data. Finally, we discuss the use of commercial mixed integer programming problem solvers in Section 3.5. We argue that large-scale application of such solvers at statistical offices for solving the error localisation problem is an unlikely event.

The mathematical formulation of the error localisation problem in Section 3.2 uses simple concepts from mathematical logic, in particular the implication operator (IF/THEN statement). This formulation for the error localisation problem is more natural than the mixed integer programming formulation in Section 3.4 or the mixed integer formulations by McKeown (1984) and Schaffer (1987). We are the first to give such a formulation of the error localisation problem.

3.2. A mathematical formulation of the error localisation problem

In this section we give a mathematical formulation of the error localisation problem for a mix of categorical and numerical data. This formulation is taken from De Waal (1998a). We start by introducing some notation and terminology. Categorical, or discrete, data are data that can assume only a finite number of values, categories, and that do not have an arithmetic structure. Examples of categorical variables are "Gender" and "Profession". The variable "Gender" can assume only the values "Male" and "Female". The variable "Profession" can assume a finite number of categories. These categories depend on the classification scheme used for "Profession". Numerical data are data that do possess an arithmetic structure. Throughout this book "numerical data or values" will mean "continuous numerical data or values", i.e. data that can assume any real number, unless otherwise noted. Integer-valued data are only considered in Chapters 9, 10 and 12.

We denote the categorical variables by v_i (i=1,…,m) and the numerical variables by x_i (i=1,…,n). For categorical data we denote the domain, i.e. the set of the possible values, of variable i by D_i. We assume that every edit, i.e. constraint that has to be satisfied by correct data, E^j (j=1,...,J) is written in the following form: edit E^j is satisfied by a record $(v_1,...,v_m,x_1,...,x_n)$ if the following statement

$$\begin{aligned} &\textsf{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\textsf{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}, \end{aligned} \tag{3.1a}$$

or

$$\begin{aligned} &\textsf{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\textsf{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}, \end{aligned} \tag{3.1b}$$

holds. The a_{ij} are assumed to be rational numbers. $F_i^j \subseteq D_i$ for all i and j. Note that without loss of generality all numerical expressions in (3.1) may assumed to be inequalities, because any equality can be expressed as two inequalities. We will not make this assumption here, however, because in later chapters we occasionally treat equalities in a different manner as inequalities for efficiency reasons.

The edits given by (3.1) are simply linear numerical conditions that are triggered by certain combinations of categorical values. Non-linear numerical conditions occur hardly ever in practice. Such non-linear edits are usually too hard to specify and too hard to handle. In principle, for each combination of categorical values different numerical conditions may be specified. One may even specify a self-contradicting numerical condition, such as "$0 \geq 1$" for a combination of categorical values. That edit then means that this particular combination of categorical values may not occur.

All edits given by (3.1) have to be satisfied simultaneously. We assume that the edits can indeed be satisfied simultaneously, i.e. we assume that the set of edits is consistent. Without loss of generality we also assume that the set of edits cannot be split into several disjoint subsets, i.e. subsets without any overlapping variables. If a set of edits can be split into disjoint subsets, we assume that this has already been done and that the error localisation problem is solved for each subset separately.

The condition after the IF-statement, i.e. "$v_i \in F_i^j$ for all i=1,…,m", is called the IF-condition of the edit. The condition after the THEN-statement is called the THEN-condition. A categorical variable v_i is said to *enter* an edit E^j given by (3.1) if $F_i^j \subset D_i$ and $F_i^j \neq D_i$, i.e. if F_i^j is strictly contained in the domain of variable i. That edit is then said to be *involved with* this categorical variable. A numerical variable x_i is said to *enter* the THEN-condition of edit E^j given by (3.1) if $a_{ij} \neq 0$. That THEN-condition is then said to be *involved with* this numerical variable.

We assume that none of the values of the variables entering the edits may be *missing*. That is, we assume that for each variable entering the edits a value has to be filled in. Any field for which the value is missing is hence considered to be erroneous.

The set in the THEN-condition of (3.1) may be the empty set or the entire n-dimensional real vector space. If the set in the THEN-condition of (3.1) is the entire n-dimensional real vector space, then the edit is always satisfied. Such an edit may be discarded. If the set in the THEN-condition of (3.1) is empty, then the edit is failed by any record for which the IF-condition holds true, i.e. for any record for which $v_i \in F_i^j$ for i=1,…,m. Likewise, F_i^j in (3.1) may be the empty set or equal to D_i. If a set $F_i^j = \varnothing$ (for some i=1,...m), the edit is always satisfied, and can be discarded. If the IF-condition does not hold true for a particular record, the edit is satisfied, irrespective of the values of the numerical variables.

For each record $(v_1^0,...,v_m^0,x_1^0,...,x_n^0)$ in the data set that is to be edited automatically we now have to determine, or more precisely: have to ensure the existence of, a synthetic record $(v_1,...,v_m,x_1,...,x_n)$ such that (3.1) becomes satisfied for all edits j=1,…,J and such that

$$\sum_{i=1}^{m} w_i \delta(v_i^0, v_i) + \sum_{i=1}^{n} w_{m+i} \delta(x_i^0, x_i) \qquad (3.2)$$

is minimised. Here w_i is the so-called reliability weight of variable i, $\delta(y^0, y) = 1$ if $y^0 \neq y$, and $\delta(y^0, y) = 0$ if $y^0 = y$. The reliability weight expresses how reliable the value of the corresponding variable is. The higher the reliability weight, the more reliable the value. The objective function (3.2) is simply the weighted number of variables that have to be changed. The variables for which the values in the synthetic record differ from the original values plus the variables for which the original value was missing together form an optimal solution to the error localisation problem.

The error localisation problem can be formulated compactly as:

> Minimise (3.2) so that (3.1) is satisfied for all edits.

Note that the above formulation is a mathematical formulation of the generalised Fellegi-Holt paradigm. Note also that there may be several optimal solutions to a specific instance of the error localisation problem.

Our aim is to find and enumerate all optimal solutions to the error localisation problem. For an optimisation problem this is quite an unnatural aim. The reason for pursuing this goal is that the actual *statistical* problem of automatic data editing is more comprehensive than the above optimisation problem. This statistical problem is "simply" the problem of obtaining high quality data from a data set with errors in an efficient manner. Solving the above-mentioned error localisation problem is only one step in this process. For instance, after the erroneous fields have been identified, they must be imputed (hereby carefully taking potentially outliers into account) and later the records must be re-weighted. To solve the underlying statistical problem effectively it is not sufficient to simply solve the optimisation problem. In particular, during the error localisation phase one should somehow take into account that the identified errors can indeed be imputed in such a way that the resulting records are of sufficient quality. One should also somehow take into account that the final re-weighted data set is of sufficient quality. The statistical problems related to the imputation phase and re-weighting phase need to be taken into account

during the error localisation phase. These problems are at least as important, and difficult, as solving the above-mentioned mathematical error localisation problem.

By generating all optimal solutions to the mathematical error localisation problem, we gain the option to later select one of these optimal solutions, using a secondary, more statistical criterion. The variables involved in the selected solution for which the values in the synthetic record differ from the original values are set to missing. In most practical cases, there is only one optimal solution to a record, but records with more solutions do regularly occur. In the worst cases we have experienced so far, records had many thousands optimal solutions. So many optimal are extremely rare, however. In practice, records with 10 or more optimal solutions already occur rarely.

In the present book we will not explore the problem of selecting an optimal solution to the error localisation problem from several optimal solutions in more detail.

The weights w_i are fixed for each record that is to be edited, but may differ for different records. In practice, the weights may be calculated before a record is edited automatically. In this way the probability that a particular value in a particular record is incorrect can be taken into account (see also Section 13.4.3).

In Chapter 9 the problem described in this section is extended further to include numerical variables that have to be integer-valued, rather than only numerical variables that are continuous.

The error localisation problem is NP-complete as the so-called satisfiability problem can be transformed into an error localisation problem in polynomial time (see also Liepins, Garfinkel and Kunnathur, 1982; Willenborg, 1988; see, for example, Schrijver, 1986, for an explanation of the term NP-complete). So, assuming $P \neq NP$, for any algorithm we can design problems that take a more than polynomial time (in the number of edits and number of variables) to solve. Therefore, our aim will not be to develop algorithms that solve the error localisation problem efficiently for the worst cases, but rather to develop algorithms that solve the problem efficiently for average cases.

The error localisation problem for purely numerical data is a special case of the so-called (linear) fixed charge problem (see e.g. Hirsch and Dantzig, 1968; Walker, 1976; McKeown, 1975 and 1981; McKeown and Ragsdale, 1990; Ragsdale and McKeown, 1991). The fixed-charge problem can be formulated as

$$\text{Minimise } \sum_{i=1}^{n} (c_i x_i + d_i y_i) \tag{3.3}$$

subject to

$$A\mathbf{x} = \mathbf{b}\,, \tag{3.4}$$

$$\mathbf{x} \geq \mathbf{0}\,, \tag{3.5}$$

$$y_i = \begin{cases} 1 & \text{if} \quad x_i > 0 \\ 0 & \text{if} \quad x_i = 0 \end{cases} \qquad \text{for } i=1,\ldots,n. \tag{3.6}$$

The fixed-charge problem reduces to a linear programming problem if **d = 0**. It reduces to the error localisation problem for purely numerical data if **c = 0**. The fixed-charge problem is relatively easy to solve if the continuous, or "unit", costs c_i dominate the fixed, or "setup", costs d_i. If the fixed costs dominate the continuous costs, the fixed-charge problem is quite difficult to solve (for computational "evidence" for this claim we refer to the above-mentioned papers on the fixed-charge problem). In the error localisation problem the fixed costs completely dominate the continuous costs, which are zero by definition. Algorithms for the general fixed-charge problem therefore do not offer an efficient way for solving the error localisation problem.

3.3. A naive approach

It is clear that at least one value per violated edit should be changed. Let us assume for a while that it is sufficient to change any value per violated edit. In that case the error localisation problem reduces to an associated set-covering problem. To formalise this, we define variables y_i (i=1,...,n), where y_i equals 1 if the value of variable i should be changed, and equals 0 otherwise. Of course, y_i equals 1 if the value of variable i is missing. For a general set of edits the set-covering problem is given by: minimise the objective function given by

$$\sum_{i=1}^{n} w_i y_i \tag{3.7}$$

subject to the condition that in each violated edit at least one variable should be changed. We define

$$a_{ij} = \begin{cases} 1 & \text{if variable } i \text{ is involved in edit } j \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

Then the constraints can be written as

$$\sum_{i=1}^{n} a_{ij} y_i \geq 1 \quad \text{for } j \in \Omega_V\,, \tag{3.9}$$

where Ω_V is the set of edits violated by the record under consideration.

Unfortunately, our assumption that any value per violated edit can be changed generally does not hold. The solution to the associated set-covering problem is usually not a solution to the corresponding error localisation problem. This is illustrated by the following example.

Example 3.1:
Suppose the explicitly specified edits are given by

$$T = P + C\,, \tag{3.10}$$

$$0.5 \leq \frac{C}{T} \leq 1.1, \tag{3.11}$$

$$0 \leq \frac{T}{N} \leq 550, \tag{3.12}$$

$$T \geq 0, \tag{3.13}$$

$$C \geq 0, \tag{3.14}$$

and

$$N \geq 0. \tag{3.15}$$

where T denotes the turnover of an enterprise, P its profit, C its costs, and N the number of employees. The turnover, profit and costs are given in thousands of Euros. Edit (3.10) says that the turnover of an enterprise should be equal to the sum of the profit and the costs. Edit (3.11) gives bounds for the costs of an enterprise in terms of the turnover, edit (3.12) gives bounds for the turnover in terms of the number of employees, and edits (3.13) to (3.15) say that the turnover, the costs of an enterprise and the number of employees are non-negative. Edits (3.10), (3.13), (3.14) and (3.15) are edits that can be logically derived, and hold for every enterprise. Edits (3.11) and (3.12) cannot be logically derived, and will hold only for certain classes of enterprises.

Let us consider a specific record with values $T = 100$, $P = 40{,}000$, $C = 60{,}000$ and $N = 5$. Edits (3.12) to (3.15) are satisfied, whereas edits (3.10) and (3.11) are violated. We assume that the reliability weights of variables T, P and C equal 1, and the reliability weight of variable N equals 2. That is, the value of variable N, the number of employees, is considered trustworthier than the values of the financial variables T, P and C.

The set-covering problem associated to the error localisation problem has the optimal solution that the value of T should be changed, because this variable covers the violated edits and has a minimal reliability weight. The optimal value of the objective function (3.7) of the set-covering problem equals 1. However, to satisfy edit (3.10) by changing the value of T the imputation value of T should be 100,000, but in that case edit (3.12) would be violated. The optimal solution of the set-covering problem is not a feasible solution to the error localisation problem, because variable T cannot be imputed consistently. The error localisation problem has the optimal solution that variables P and C should both be changed. The optimal value of the objective function (3.2) to the error localisation problem equals 2. This is larger than the optimal value of the objective function of the associated set-covering problem. Possible imputation values for variables P and C are $P = 40$ and $C = 60$. The resulting, imputed, record passes all edits. Note that in this example the respondent probably forgot that the values of P and C should be given in thousands of Euros. ■

A feasible solution to the error localisation problem is a feasible solution to the associated set-covering problem, but not vice versa. Hence, the value of the optimal solution to the error localisation problem is at least equal to the value of the optimal solution to the associated set-covering problem.

3.4. A mixed integer programming formulation

In this section we assume, for notational convenience, that all edits are of type (3.1a). An equality can, of course, be represented by two inequalities, so the generality of our model is not reduced by this choice. We also assume that the values of the numerical variables are bounded. That is, we assume that for the *i*-th numerical variable (*i*=1,...,*n*) constants α_i and β_i exist such that

$$\alpha_i \leq x_i \leq \beta_i \,. \tag{3.16}$$

In practice, such values α_i and β_i always exist, because numerical variables that occur in statistical data are bounded. If the value of the *i*-th numerical variable is missing we code this by assigning either a value less than α_i or a value larger than β_i to x_i.

Denote the number of categories of the *i*-th categorical variable by g_i (*i*=1,...,*m*), i.e. $g_i = \mid D_i \mid$. For the *k*-th value c_{ik} of categorical variable *i* we introduce a binary variable γ_{ik} such that

$$\gamma_{ik} = \begin{cases} 1 & \text{if the value of categorical variable } i \text{ equals } c_{ik} \\ 0 & \text{otherwise.} \end{cases} \tag{3.17}$$

To the *i*-th categorical variable a vector $(\gamma_{i1},...,\gamma_{ig_i})$ corresponds such that $\gamma_{ik} = 1$ if and only if the value of this categorical variable equals c_{ik}, otherwise $\gamma_{ik} = 0$. For each categorical variable *i* the relation

$$\sum_k \gamma_{ik} = 1 \tag{3.18}$$

has to hold true. A vector $(\gamma_{i1},...,\gamma_{ig_i})$ will also be denoted by $\boldsymbol{\gamma_i}$. If the value of the *i*-th categorical variable (*i*=1,...,*m*) is missing we set all γ_{ik} equal to zero (*k*=1,…, g_i).

In terms of the binary variables γ_{ik} an edit *j* given by (3.1a) can be written as

$$a_{1j}x_1 + ... + a_{nj}x_n + b_j \geq M\left(\sum_{i=1}^{m}\left(\sum_{c_{ik} \in F_i^j} \gamma_{ik} - 1\right)\right), \tag{3.19}$$

where *M* is a sufficiently large positive number. If the IF-condition of (3.1a) holds true, the right-hand side of (3.1a) equals zero. Consequently, the THEN-condition of (3.1a) has to hold true for the numerical variables. If the IF-condition of (3.1a) does not hold true, the right-hand side of (3.19) equals a large negative value. Consequently, (3.19) holds true irrespective of the values of numerical variables (provided that the value of the right-hand side of (3.19) is sufficiently small).

Alternatively, inequality (3.19) may be replaced by

$$\sum_{i=1}^{m}\left(\sum_{c_{ik}\in F_i^j}\gamma_{ik}-1\right)\le -\frac{u}{2}, \tag{3.20}$$

and

$$a_{1j}x_1+\ldots+a_{nj}x_n+b_j\ge u\sum_{i=1}^{n}a_{ij}\alpha_i\,, \tag{3.21}$$

where u is a binary variable. If $v_i\in F_i^j$ for i=1,…,m, then u has to equal zero, and the THEN-condition of (3.1a) has to be satisfied. If u equals one, then (3.21) is always satisfied.

If (3.1a) is not satisfied by a record $(v_1^0,...,v_m^0,x_1^0,...,x_n^0)$, or equivalently if (3.19) is not satisfied by $(\gamma_{\mathbf{1}}^{\mathbf{0}},...,\gamma_{\mathbf{m}}^{\mathbf{0}},x_1^0,...,x_n^0)$, then we seek values e_{ik}^P (k=1,..., g_i; i=1,...,m), e_{ik}^N (k=1,..., g_i; i=1,...,m), z_i^P (i=1,...,n) and z_i^N (i=1,...,n). The e_{ik}^P and the z_i^P correspond to positive changes in value of γ_{ik}^0 and x_i^0, respectively. Likewise, the e_{ik}^N and the z_i^N correspond to negative changes in value of γ_{ik}^0 and x_i^0, respectively. The vector $(e_{i1}^P,...,e_{ig_i}^P)$ will also be denoted as $\mathbf{e_i^P}$, and the vector $(e_{i1}^N,...,e_{ig_i}^N)$ as $\mathbf{e_i^N}$.

The above values have to be determined in such a way that

$$\sum_{i=1}^{m}w_i\left(\sum_k e_{ik}^N\right)+\sum_{i=1}^{n}w_{m+i}\left(\delta(z_i^P)+\delta(z_i^N)\right), \tag{3.22}$$

where w_i is the reliability weight of variable i, $\delta(x)=1$ if and only if $x\neq 0$ and $\delta(x)=0$ otherwise, is minimised subject to the constraints:

$$e_{ik}^P,e_{ik}^N\in\{0,1\}\,, \qquad (i=1,...,m) \tag{3.23}$$

$$z_i^P,z_i^N\ge 0\,, \qquad (i=1,...,n) \tag{3.24}$$

$$e_{ik}^P+e_{ik}^N\le 1 \qquad (i=1,\ldots,m) \tag{3.25}$$

$$\sum_k e_{ik}^P\le 1\,, \qquad (i=1,...,m) \tag{3.26}$$

$$e_{ik}^N=0 \ \text{ if } \gamma_{ik}^0=0 \qquad (i=1,...,m) \tag{3.27}$$

$$\sum_k(\gamma_{ik}^0+e_{ik}^P-e_{ik}^N)=1\,, \qquad (i=1,...,m) \tag{3.28}$$

$$\alpha_i\le x_i^0+z_i^P-z_i^N\le\beta_i \qquad (i=1,...,n) \tag{3.29}$$

and

$$\sum_{i=1}^{n} a_{ij}(x_i^0 + z_i^P - z_i^N) + b_j \geq M\left(\sum_{i=1}^{m}\left(\sum_{c_{ik}\in F_i^j}(\gamma_{ik}^0 + e_{ik}^P - e_{ik}^N) - 1\right)\right) \qquad (3.30)$$

for all edits j=1,...,K.

Relation (3.25) expresses that the same negative and positive correction may not be applied to a categorical variable. Relation (3.26) expresses that at most one value may be imputed, i.e. estimated and filled in, for a categorical variable, and relation (3.27) that a negative correction may not be applied to a categorical variable if the original value is not equal to the corresponding category. Relation (3.28) ensures that a value for each categorical variable is filled in, even if the original value was missing. Relation (3.29) states that the value of a numerical variable must be bounded by the appropriate constants. In particular, relation (3.29) also states that the value of a numerical variable may not be missing. Finally, relation (3.30) expresses that the modified record should satisfy all edits given by (3.1a).

After solving this optimisation problem the resulting, modified record is given by

$$(\gamma_1^0 + \mathbf{e_1^P} - \mathbf{e_1^N}, ..., \gamma_m^0 + \mathbf{e_m^P} - \mathbf{e_m^N}, x_1^0 + z_1^P - z_1^N, ..., x_n^0 + z_n^P - z_n^N).$$

A solution to the above mathematical problem corresponds to a solution to the error localisation problem. A solution to the error localisation problem is simply given by naming the variables of which the values have to be changed, without specifying their new values. Given a modified record corresponding to an optimal solution to the above problem, the corresponding solution to the error localisation problem is given by the variables for which the value in this modified record differs from the original value. As we already mentioned in Section 3.2, our aim is to find all optimal solutions.

The above optimisation problem is a translation of the generalised Fellegi-Holt paradigm in mathematical terms. The objective function (3.22) is the sum of the reliability weights of the variables of which the original values must be modified. Note that minimising (3.22) is equivalent to minimising

$$\sum_{i=1}^{m} w_i\left(\sum_k e_{ik}^P\right) + \sum_{i=1}^{n} w_{m+i}\left(\delta(z_i^P) + \delta(z_i^N)\right). \qquad (3.31)$$

The objective function (3.31) is the sum of the reliability weights of the variables for which a new value must be imputed. To be precisely, the value of this objective function is equal to the value of the objective function (3.22) plus the sum of reliability weights of the categorical variables for which the value was missing.

We end this section with two remarks. First, note that in practice only one e_{ik}^N-variable for each variable i is needed, namely for the index k for which $\gamma_{ik}^0 = 1$. The other e_{ik}^N equal zero. In the present chapter we use g_i binary e_{ik}^N-variables for each variable i to keep the notation as simple as possible. Of course, in a practical implementation one would use only one e_{ik}^N-variable for each variable i instead of g_i e_{ik}^N-variables. Second, note that in an

optimal solution to the above optimisation problem either $z_i^P = 0$ or $z_i^N = 0$ (or both), and that, similarly, in an optimal solution either $e_{ik}^P = 0$ or $e_{ik}^N = 0$ (or both).

3.5. Using standard algorithms

Perhaps the best known class of algorithms for minimising a linear objective function subject to linear constraints where some of the variables involved are binary variables consists of the so-called branch-and-bound algorithms (see for example Nemhauser and Wolsey (1988), Walukiewicz (1990), Hillier and Lieberman (1995), Wolsey (1998) for an introduction to branch-and-bound algorithms). Before we can apply a standard branch-and-bound algorithm to the general error localisation problem we first have to introduce some additional variables.

The objective function (3.22) contains a non-linear function, namely δ. By introducing additional binary variables d_i^P and d_i^N for i=1,...,n that satisfy the following relations:

$$d_i^P, d_i^N \in \{0,1\} \tag{3.32}$$

$$d_i^P \leq M z_i^P, \tag{3.33}$$

$$M d_i^P \geq z_i^P, \tag{3.34}$$

$$d_i^N \leq M z_i^N \tag{3.35}$$

and

$$M d_i^N \geq z_i^N, \tag{3.36}$$

where M is again a sufficiently large positive number, we can rewrite (3.22) in the following linear form:

$$\sum_{i=1}^{m} w_i \left(\sum_k e_{ik}^N \right) + \sum_{i=m+1}^{m+n} w_i (d_i^P + d_i^N). \tag{3.37}$$

The objective function (3.37) should be minimised subject to the constraints (3.23) to (3.30) and (3.32) to (3.36). Relations (3.33) and (3.34) express that $d_i^P = 1$ if and only if $z_i^P \neq 0$, otherwise $d_i^P = 0$. Similarly, relations (3.35) and (3.36) express that $d_i^N = 1$ if and only if $z_i^N \neq 0$, otherwise $d_i^N = 0$. If and only if a continuous variable differs from zero, the associated binary variable equals one. The fact whether or not a continuous variable differs from zero is incorporated in the objective function by means of the associated binary variable. Stated in this way the general error localisation problem becomes a mixed integer programming problem.

A branch-and-bound algorithm is an iterative algorithm for solving (mixed) integer programming problems. A branch-and-bound algorithm basically consists of three steps: branching, bounding and fathoming. These three steps are performed during each iteration.

During the branch-step a linear programming (LP) problem is split up into two separate linear subproblems by fixing a binary variable at either 0 or 1. For each of these subproblems a bound is determined during the bound-step. This bound is determined by solving the LP relaxation to the subproblem, i.e. by solving the linear subproblem without taking into account that the binary variables that have not been fixed yet can only assume the values 0 or 1. Finally, during the fathom-step it is determined whether the subproblems themselves have to be split up into even smaller subproblems. A subproblem does not have to be split up into smaller subproblems if the optimal solution to the LP relaxation of the subproblem is a solution to the subproblem as well (in that case the optimal solution to the LP relaxation is also an optimal solution to the subproblem), if the bound obtained from the LP relaxation is worse than a bound obtained from a previously found solution to the (mixed) integer programming problem, or if the LP relaxation has no feasible solution.

The application of a branch-and-bound algorithm for solving the general error localisation problem is, in principle, possible. Modern commercial solvers for mixed integer programming (MIP) problems, such as ILOG CPLEX (see *ILOG CPLEX 7.5 Reference Manual*, 2001), are powerful enough to determine an optimal solution to the error localisation problem within an acceptable amount of time. There are a few problems related to applying a commercial MIP solver to the error localisation problem, however.

A technical problem is that preferably we would like to generate *all* optimal solutions to the error localisation problem. Standard commercial MIP solvers seem to be less suited for this task. A special-purpose algorithm designed to find all optimal solutions to the error localisation problem may lead to better results than standard MIP solvers.

There are also several non-technical problems with using a commercial MIP solver. First, the error localisation problem is just a part of a statistical process to clean records. Statistical offices like to have complete control over how this statistical process works. They want to be able to mould the error localisation problem so it fits into the rest of the statistical production process. Statistical offices do not want to end up in a situation where the commercial MIP solver restricts them in their actions.

Second, statistical offices do not like to depend on commercial software vendors in general. Changes in the software of the commercial software vendor may have a major impact on the software systems at the statistical office. It may be difficult to maintain software systems, especially if the commercially acquired software is mathematically relatively complicated, as is the case for MIP solvers.

Third, commercial MIP solvers are quite expensive. In a statistical office there are potentially many users of an automatic data editing system. This would be especially true if the automatic data editing system were integrated into the software for computer-assisted editing. Such integration would allow the human editor to ask the computer for a suggestion on how to clean a particular record. They can then accept this suggestion, modify this suggestion, or reject the suggestion. Such integration is, for example, envisaged at Statistics Netherlands. Buying hundreds of licences for a commercial MIP solver could be too costly for a statistical office.

In Chapter 11 we compare computational results of a program based on a commercial MIP solver, ILOG CPLEX, to several other programs for automatic error localisation. These other programs are based on algorithms described in later chapters of this book.

4. The Fellegi-Holt Method

4.1. Introduction

In Section 2.5 we have introduced the well-known paradigm due to Fellegi and Holt (1976) that says that an erroneous record should be made to satisfy all edits by changing the values of the fewest possible number of variables. A formulation for the resulting mathematical problem has been presented in Section 3.2.

In their paper Fellegi and Holt not only propose their paradigm, they also propose an important method for solving the resulting mathematical problem. This method can be applied to both categorical data and numerical data, and is the best-known method for solving the error localisation problem. It is also the most often applied technique for solving the error localisation problem for categorical data. In this chapter we explain the method developed by Fellegi and Holt.

This chapter hardly contains original results. An exception is a proof that the Fellegi-Holt method works for numerical data. Such a proof appears to be lacking in literature. Another exception is the brief remark in Section 4.5 that the Fellegi-Holt method can, in principle, also be used to solve the error localisation problem in a mix of categorical and numerical data. In literature no one seems to have noted this before.

This chapter is organised in the following way. In the next three sections we restrict ourselves to categorical data. In Section 4.2 we illustrate the basic idea of the Fellegi-Holt approach without any mathematical rigour. Mathematical details are sketched in Section 4.3. Proofs are not provided. The reader is referred to the original papers for these proofs. Improvements on the method proposed by Fellegi and Holt are examined in Section 4.4. Numerical and mixed data are discussed in Section 4.5. The chapter concludes with a brief discussion in Section 4.6.

4.2. The basic idea of the Fellegi-Holt approach

The method developed by Fellegi and Holt is based on generating so-called *implicit*, or *implied*, edits. Such implicit edits are logically implied by the explicitly specified edits. Implicit edits can be defined for numerical as well as categorical data.

Although implicit edits are redundant, they can reveal important information about the feasible region defined by the explicitly defined edits. This information is, of course, already contained in the explicitly defined edits, but there that information may be rather hidden. Implicit edits sometimes allow one to see relations between variables more clearly. We illustrate this point by means of a simple example, which is taken from Daalmans (2000). We will refer to this example more often in this chapter.

Example 4.1:
In a small survey respondents are asked to choose one of the possible alternatives for the following three questions:
1. What is the most important reason for you to buy sugar?
2. Do you drink coffee with sugar?
3. What is the average amount of sugar you consume in one cup of coffee?

The alternatives for the first question are:
- 'I consume sugar in my coffee';
- 'I use sugar to bake cherry pie';
- 'I never buy sugar';
- 'Other reason'.

The alternatives for the second question are:

- 'Yes';
- 'No'.

The alternatives for the last question are:
- '0 grams';
- 'more than 0 grams but less than 10';
- 'more than 10 grams'.

The following (explicit) edits have been defined:
1. The main reason to buy sugar is not to consume it with coffee, for someone who does not drink coffee with sugar.
2. The average amount of sugar consumed in one cup of coffee by someone who drinks coffee with sugar is not equal to 0 gram.
3. Someone who never buys sugar does not consume more than 0 gram of sugar in his coffee on average.

An example of an implicit edit is:

4. Someone who never buys sugar does not consume sugar in his coffee.

This edit is implied by the second and third explicit edit (since the second explicit edit implies that somebody who drinks sugar in his/her coffee must consume more than 0 gram of sugar per cup of coffee on average and the third explicit edit says that somebody who drinks more than 0 gram of sugar per cup of coffee on average sometimes buys sugar).

Edit 4 is by definition a redundant edit, because this information is already present in the second and third explicit edit. However, this edit makes the relation between buying sugar and consuming sugar more clearly. This relation is less clear if one only looks at the second and third explicit edit. The benefits of generating implicit edits become more apparent later when we continue this example.

A trivial example of another implicit edit is the following:

- Explicit edit 2 or explicit edit 3 has to hold true.

This is an implicit edit, but it is rather useless as it does not make any relation between the variables clearer. ■

Note that for categorical data many 'useless' implicit edits can be derived like the second implicit edit in Example 4.1 above. For numerical data the set of edits that are logically implied by the explicitly specified edits contains infinitely many elements. A simple example is given below.

Example 4.2:
If $x \geq 1$ is an edit, then $\lambda x \geq \lambda$ is an implied edit for all $\lambda \geq 0$. ■

Generating all implicit edits is out of the question for numerical data, and is a waste of time and memory for categorical data.

The method proposed by Fellegi and Holt starts by generating a well-defined sufficiently large set of implicit and explicit edits. This set of edits is referred to as the *complete set of edits*. It is referred as the *complete* set of edits not because all possible implicit edits are generated, but because this is the set of (implicit and explicit) edits that is sufficient and necessary to translate the error localisation problem into a set-covering problem. The mathematical details on how to generate the complete set of edits are provided in Section 4.3.

In particular, once a complete set of edits has been generated it suffices to find a set of variables *S* that covers the violated (explicit and implicit) edits, i.e. in each violated edit at least one variable of *S* should be involved.

For categorical data a formal definition of implicit edits and the complete set of edits will be given later in this chapter. Here we restrict ourselves to giving the complete set of edits for Example 4.1.

Example 4.3:
In our example the complete set of edits is given by edits 1 to 4, and

5. The average amount of sugar consumed per cup of coffee by someone whose main reason to buy sugar is to consume it with coffee is not equal to 0 gram. ■

Why the complete set of edits is important is precisely explained in mathematical terms in the next section. Here we illustrate the idea with an example.

Example 4.4:
Suppose the explicit edits are given again by the explicit edits of Example 4.1. Suppose also that the answers recorded for one of the respondents are:
1. The most important reason for buying sugar: I never buy sugar;
2. Do you drink coffee with sugar: Yes;

3. What is the average amount of sugar per cup of coffee: 0 grams.

Note that this record does not satisfy the second explicit edit. Obviously either the answer to the second question or the answer to the third question has to be changed. Note that changing the answer to the first question alone cannot result in a consistent record.

An approach to see which value can be changed is simply trial and error. One possibility is to change the third answer to "more than 0 but less than 10 grams". As a consequence, the second explicit edit will become satisfied, but unfortunately the third explicit edit will become failed. So, changing the third answer to "more than 0 but less than 10 grams" is not such a good idea.

Let us try to change the third answer to "more than 10 grams". This is neither a good idea, because the second explicit edit will become satisfied through this change, but again the third explicit edit will become failed.

Now, suppose the answer to the second question is changed to "No", while the third answer is fixed to its original value. Now all edits have become satisfied and a solution to the error localisation problem has been found. In this small example a solution is found after a few steps using the trial and error approach. But for large problems the trial and error approach is not so efficient. In the worst case all $\prod_{i=1}^{m} |D_i|$ possible records have to be checked in order to find all optimal solutions. Here the implicit edits show their importance as we illustrate in the paragraph below.

Consider (implicit) edit 4 of Example 4.1, i.e. the edit that says: "Someone who never buys sugar does not consume sugar in his coffee". Note that to determine whether this edit is satisfied or not we only have to consider the answers to the first two questions. That is, whether the edit is satisfied or not does not depend on the answer to the third question. Note also that this edit is failed by the record under consideration. Changing the answer to the third question cannot make this edit satisfied. So, we do not have to consider changing only the value of the third question.

This edit is implied by the second and third explicit edits, so it is obviously redundant. However, as we see it does contain useful information that helps us to identify the most implausible values. ■

4.3. Fellegi-Holt approach for categorical data: mathematical details

For purely categorical data, the edits that we consider in this book are given by:

$$\text{IF } v_i \in F_i^j \text{ (for } i=1,\ldots,m\text{) THEN } \varnothing. \qquad (4.1)$$

An edit given by (4.1) is violated if $v_i \in F_i^j$ for all $i=1,\ldots,m$. Otherwise, the edit is satisfied.

Alternatively, we will write a categorical edit, E^j, given by (4.1) as

$$P(E^j) = \prod_{i=1}^{m} F_i^j \,, \tag{4.2}$$

where $\prod$ denotes the Cartesian product. That is, edit E^j is failed if and only if the values v_i of the record under consideration lie in the space given by the right-hand-side of (4.2). Fellegi and Holt (1976) refer to (4.2) as the *normal form* of (categorical) edits.

A set of edits is satisfied if all edits given by (4.2) are satisfied, i.e. a set of edits is failed if at least one edit is failed. If we denote the set of edits E^j (j=1,…,J) by $\overline{E}$, then a record **v** fails this set of edits if and only if

$$\mathbf{v} \in P(\overline{E}) \,, \tag{4.3}$$

where

$$P(\overline{E}) = \bigcup_{j=1}^{J} P(E^j) \,. \tag{4.4}$$

If the value set F_i^j is a proper subset of the domain D_i of variable i, then variable i is said to *enter* edit E^j, and edit E^j is said to *involve* variable i. Fellegi and Holt show that any system of categorical edits can equivalently be expressed in normal form.

A simple example, illustrating the introduced concepts, is given below.

Example 4.5:
Suppose there are three variables: “Age”, “Marital Status” and “Sex”. The variable “Age” assumes three values: 1, 2 and 3 (i.e. $D_1 = \{1,2,3\}$), representing respectively: “Age=0-14”, “Age=15-80” and “Age>80”. The variable “Marital Status” only has two possible values: 1 and 2 (i.e. $D_2 = \{1,2\}$), representing respectively “Married” and “Not married”. The variable “Sex” assumes two possible values: 1 and 2 (i.e. $D_3 = \{1,2\}$), representing respectively “Male” and “Female”.

The statement:

“IF (Age<15) THEN (Marital Status = Not Married)”

is identical to the statement that a failure occurs if both (Age=0-14) and (Marital Status = Married) hold true, irrespective of the value of variable “Sex”. In more mathematical notation: $F_1^1 = \{1\}$, $F_2^1 = \{1\}$ and $F_3^1 = D_3 = \{1,2\}$, and in normal form the edit is given by: $P(E^1) = \{1\} \times \{1\} \times \{1,2\}$.

“Age” and “Marital Status” enter this edit because F_1^1 is a proper subset of D_1 and F_2^1 is a proper subset of D_2, but the edit does not involve “Sex” since F_3^1 is not a proper subset of D_3. ■

In Example 4.6 below we return to Example 4.1.

Example 4.6:
In normal form the explicit edits of Example 4.1 are given by

$$P(E^1) = \{1\}\times\{2\}\times\{1,2,3\} \tag{4.5}$$

$$P(E^2) = \{1,2,3,4\}\times\{1\}\times\{1\} \tag{4.6}$$

$$P(E^2) = \{3\}\times\{1,2\}\times\{2,3\} \tag{4.7}$$ ■

In Section 4.2 we have already illustrated the concept of implicit edits. In the algorithm of Fellegi and Holt a subset of all possible explicit and implicit edits, the so-called *complete set of edits*, has to be generated. The following lemma of Fellegi and Holt is the basis of generating implicit edits.

Lemma 4.1. Let $\overline{E}_g$ be an arbitrary set of edits, explicit or already generated implicit ones, that involve a field g ($g \in \{1,\ldots,m\}$). Let E^* be the edit defined by:

$$F_j^* = \bigcap_{k, E^k \in \overline{E}_g} F_j^k \qquad j \in \{1,\ldots,m\}, j \neq g$$

$$F_g^* = \bigcup_{k, E^k \in \overline{E}_g} F_j^k .$$

If $F_j^* \neq \varnothing$ for every $j \in \{1,\ldots,m\}$, then E^* is an implicit edit in normal form.

Fellegi and Holt (1976) prove that all implicit edits required for their complete set of edits can be generated via repeated application of this lemma. Note that this lemma implies that in general implicit edits can be deduced from one, two, three, four, or even more explicit or implicit edits.

The subset $\overline{E}_g$ is called the *contributing set,* it contains the *contributing edits*, or the edits that imply E^*. Field g is called the *generating field* of edit E^*.

In the example below we again return to Example 4.1.

Example 4.7:
In Example 4.1 the (implicit) edit 4, denoted as E^4, can be obtained using Lemma 4.1 with the third field as the generating field and contributing set $\overline{E}_g = \{E^2, E^3\}$, i.e. the second and third edits.

Besides E^4 there are two other implicit edits: E^5 with $P(E^5) = \{1\} \times \{1,2\} \times \{1\}$ and E^6, with $P(E^6) = \{1, 3\} \times \{2\} \times \{2,3\}$. Edit E^5 can be obtained by using Lemma 4.1 with contributing set $\{E^1, E^2\}$ and generating field 2. Edit E^4 can be obtained by using Lemma 4.1 with generating field 1 and contributing set $\{E^1, E^2\}$. In fact E^6 and E^4 could also be combined, using variable 2 as generating field but the resulting implied edit would be identical to E^3. ■

Fellegi and Holt show that it is not necessary to generate all implicit edits by means of Lemma 4.1 in order to solve the error localisation problem. To this end they introduce the concept of an essentially new implicit edit. An *essentially new* implicit edit E^* is an implicit edit that does not involve its generating field *g* (i.e. $F_g^* = D_g$). The set of explicit edits together with the set of all essentially new implicit edits is called the complete set of edits.

Example 4.8:
In Example 4.7 it is easy to see that E^4 and E^4 are both essentially new implicit edits, but E^6 is not. The complete set of edits $\overline{E}_C$ is given by $\{E^1, E^2, E^3, E^4, E^5\}$. ■

An essentially new implicit edit E^* can be interpreted as the projection of the edits $\overline{E}_g$ on their entering fields except generating field *g*. That is, it defines a relation that has to hold for these entering fields except field *g*. By generating essentially new implicit edits, hidden relations between various variables can be clarified. Once the hidden relations have been made explicit, solving the error localisation problem is relatively straightforward.

After generation of the complete set of edits all failed edits are selected for each record. According to Theorem 4.1, i.e. Corollary 2 to Theorem 1 of Fellegi and Holt (1976), mentioned below, sets of variables *S* have to be found that cover the set of failed edits (i.e. at least one variable contained in *S* is involved in each failed edit).

Theorem 4.1. If *S* is any set of variables that covers the complete set of failed edits, then a set of values exists for the variables in *S* that together with the set of original values for all other variables will result in a consistent record. That is, the variables in *S* can be imputed consistently.

Theorem 4.1 says that some set of variables S is a (possibly suboptimal) solution to the error localisation problem if S covers the set of failed edits. Note that in order to obtain a consistent record at least one of the entering fields of each failed edit has to be imputed. Therefore it also holds true that a variable set S cannot be imputed consistently, if S does not cover the set of failed edits. Consequently, all solutions to the error localisation problem can be found by finding all covering sets of variables. According to the generalised paradigm of Fellegi and Holt all sets with a minimal sum of reliability weights among all sets of variables that cover the violated edits are the optimal solutions to the error localisation problem (see also Section 4.4.1`).

Example 4.9:
Suppose that in Example 4.1 the reliability weights are 1 for the first field, 2 for the second field and 1 for the third field. Now consider the record **v** of Example 4.4. The edits E^2 and E^4 (see also Examples 4.7 and 4.8) are violated by this record. Note that the second field enters both failed edits. In other words, field 2 covers the set of failed edits $\{E^2, E^4\}$. This implies that the variable sets $\{v_1, v_2\}$, $\{v_2, v_3\}$ and $\{v_1, v_2, v_3\}$ also cover the complete set of failed edits. However, according to the generalised paradigm of Fellegi and Holt these latter sets of variables should not be imputed.

There is one remaining variable set that also covers $\{E^2, E^4\}$, namely $\{v_1, v_3\}$. The sum of the reliability weights of the two variables in this set equals two, exactly the same as the reliability weight of the second field. This means that $\{v_2\}$ and $\{v_1, v_3\}$ are the optimal solutions to the error localisation problem. ■

Note that Theorem 4.1 implies that if the variables in a subset S cannot be imputed consistently then there is a failed (explicit or implicit) edit in which none of the variables of S are involved.

To prove Theorem 4.1 Fellegi and Holt apply a principle that we will refer to as the *lifting principle*. Define Ω_K as that subset of the set of complete edits that involves only fields 1,…,K. The lifting principle, i.e. Theorem 1 of Fellegi and Holt (1976), then states the following.

Theorem 4.2. If v_i^0 (i=1,…,K-1) are values for the first K-1 variables that satisfy all edits in Ω_{K-1}, then there exists some value v_K^0 such that the values v_i^0 (i=1,…,K) satisfy all edits in Ω_K.

Theorem 4.2 states that if the values for K-1 variables satisfy all corresponding (explicit and essentially new implicit) edits, then there exits a value for the K-th variable such that all edits corresponding to the first K variables become satisfied. In other words, the

possibility to satisfy the edits involving only K-1 variables is *lifted* to K variables. The lifting principle is a very important principle. In Chapters 8 and 9 we will return to it.

N.B. The term lifting principle as it is used in this book should not be confused with lifting ground resolutions to more abstract general resolutions as is done in predicate logic. In predicate logic "lifting" refers to translating a property for relatively concrete cases to a property for more abstract cases. In this book "lifting" simply refers to translating a property involving a certain number of variables to a property involving more variables.

Note that by repeated application of Theorem 4.2 we can show the following corollary, where m denotes the number of categorical variables.

Corollary 4.2. If v_i^0 (i=1,…,K-1) are values for the first K-1 variables that satisfy all edits in Ω_{K-1}, then there exists values v_i^0 (i=K,…,m) such that the values v_i^0 (i=1,…,m) satisfy all edits in the complete set of edits.

The correctness of Theorem 4.1 follows immediately from Corollary 4.2.

Using implicit edits to solve the error localisation problem was an important methodological breakthrough. The concept of implicit edits was, however, not a completely new idea. For instance, the concept of implicit edits is similar to that of surrogate constraints in linear or integer programming. Moreover, the Fellegi-Holt method to generate implicit edits for categorical data can be considered as a special case of the resolution technique that is used in machine learning and mathematical logic (see for example Robinson, 1965 and 1968; Russell and Norvig, 1995; Williams and Brailsford, 1996; Marriott and Stuckey, 1998; Warners, 1999; Hooker, 2000; Ben-Ari, 2001).

4.4. Improvements on the Fellegi-Holt method for categorical data

In the previous section we have described that a subset of the implicit edits is sufficient in order to solve the error localisation problem, namely the subset of the essentially new implicit edits. This subset of essentially new implicit edits is much smaller than the set of all implicit edits. Nevertheless, the practical problem of the method of Fellegi and Holt is that the number of (essentially new) implicit edits can be high. As a result the method may be too slow in practice, or even worse, the number of (essentially new) implicit edits may even be too high to be handled by a computer.

Suppose Lemma 4.1 is applied to generate the essentially new implicit edits with field j as generating field. Remember that implicit edits can be deduced from at least two edits. So if N_j denotes the number of edits (explicit and already generated implicit ones) that involve field j (j=1,…,m), then an upper bound on the number of essentially new implicit edits that can be generated by applying Lemma 4.1 is:

$$\sum_{i=2}^{N_j} \binom{N_j}{i} = 2^{N_j} - N_j - 1 . \qquad (4.8)$$

So, generation of all essentially new implicit edits grows exponentially with the number of entering fields in the original edits. Winkler (1999) even reports that the amount of computation required to generate the complete set of edits for J explicit edits is of the order exp(exp(J)). He also reports that the complete set of edits cannot always be generated in practice, due to the computer memory and computations required.

The complete set of edits can be extremely large. There are two reasons for this of which the first one plays the by far more important role in most practical situations. This first reason is that to generate the complete set of edits one should, in principle, consider all possible subsets of variables. Each of these subsets should be eliminated from the edits in order to obtain the complete set of edits. The total number of subsets of variables from a set of variables is 2^m, where m is the number of variables.

The second reason is that the number of new edits that are obtained after a variable (or subset of variables) has been eliminated may also be large. For instance, suppose there are s edits, where s is even. Suppose, furthermore, that the domain of the variable to be edited, variable g, equals $D_g = \{1,2\}$. If $F_g^j = \{1\}$ for j=1,…,s/2, and $F_g^j = \{2\}$ for j=s/2 + 1,…,s, then the total number of new edits may in the worst case equal $s^2/4$ (see also (4.8)).

Reducing computing time and required computer memory are therefore the most important aspects in implementing a Fellegi-Holt based algorithm.

Garfinkel, Kunnathur and Liepins (1986) made an important contribution to the theory of the categorical error localisation problem. In their paper they start by reformulating the categorical error localisation problem as an equivalent integer programming problem, which is a mathematical formulation of the ideas of Fellegi and Holt. We give this formulation in Subsection 4.4.1.

Further, Garfinkel, Kunnathur and Liepins provide two Fellegi-Holt based algorithms. Their first algorithm consists of rules for the implicit edit generation. In comparison with the original method proposed by Fellegi and Holt less implicit edits have to be generated, leading to a reduced computing time. Their idea is to generate only a subset of the essentially new implicit edits, namely the *non-redundant* essentially new implicit edits. This set is sufficiently large to solve the error localisation problem. The method developed by Garfinkel, Kunnathur and Liepins has been further improved by Winkler (1995). This method is sketched in Subsection 4.4.2.

The second algorithm of Garfinkel, Kunnathur and Liepins is a cutting plane algorithm. It solves a sequence of small set-covering problems (SCP's) for each failing record. In the algorithm of Fellegi and Holt the complete set of edits has to be generated and for each record an SCP has to be solved, while in this algorithm for each record only a few implicit edits have to be generated and some SCP's have to be solved. This algorithm can lead to a significant improvement in computing time in comparison with the algorithm proposed by Fellegi and Holt, especially in cases in which many fields are involved in the edits. In this book we discuss the second algorithm of Garfinkel, Kunnathur and Liepins in Chapter 10. In that same chapter we also extend the algorithm to mixed data, and propose improvements.

The Fellegi-Holt Method

Besides the method described in Subsection 4.4.2 there are other approaches to develop a system based on the Fellegi-Holt method. For instance, if the total number of essentially new implicit edits is too high, then one can resort to generating a set of implicit edits for each violated record separately. Barcaroli and Venturi (1996) show that in order to solve the error localisation problem for a certain record it is sufficient to generate the set of (essentially new) implicit edits implied by the violated explicit edits and the explicit edits in which at least one variable is involved that is also involved in a violated explicit edit.

For categorical data and edits a number of systems based on the Fellegi and Holt method have been developed, for example DAISY (Barcaroli and Venturi, 1997), SCIA (Barcaroli et al., 1995) and DISCRETE (Winkler and Petkunas, 1997).

4.4.1. The categorical error localisation problem as an SCP

Denote the set of explicit edits by $\overline{E}_E$. Suppose a record $\mathbf{v}_0 \in P(\overline{E}_E)$ is to be edited. We denote by $\mathbf{v}$ the consistent version of $\mathbf{v}_0$ (in the sense that it is the corrected record) and by z_i (i=1,…,m) a binary variable such that

$$z_i = \begin{cases} 1 & \text{if } v_i \neq v_i^0 \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

The integer programming model, equivalent to the error localisation problem, for $\mathbf{v}$ is then given by:

$$\text{Minimise } \sum_{i=1}^{m} w_i z_i \tag{4.10}$$

$$\text{Subject to: } \mathbf{v} \in D - P(\overline{E}_E) \tag{4.11}$$

Here m denotes, as usual, the number of categorical variables.

As we noted before, a necessary condition for a vector $\mathbf{v}$ to solve the error localisation problem is that it solves the following SCP (see Sections 3.3 and 4.3):

$$\text{Minimise } \sum_{i=1}^{m} w_i y_i \tag{4.12}$$

$$\text{Subject to: } \sum_{i=1}^{m} a_{ji} y_i \geq 1 \quad \text{for } E^j \in \overline{E}_F(x_0) \tag{4.13}$$

$$y_i \in \{0,1\} \qquad \text{for } i=1,\ldots,m \tag{4.14}$$

where

$$a_{ji} = \begin{cases} 1 & \text{if field } i \text{ enters } E^j \\ 0 & \text{otherwise} \end{cases} \tag{4.15}$$

$$\overline{E}_F(\mathbf{v}^0) \text{ : set of explicit edits violated by } \mathbf{v}^0. \tag{4.16}$$

The inequalities say that if an edit is violated, then the value of at least one variable involved with that edit has to be changed. In Section 4.3 we noted that if $\overline{E}_F$ is expanded to include the set of implicit edits, then a sufficient and necessary condition for z to solve the error localisation problem is that z solves the SCP. The correctness of this observation is, of course, guaranteed by Theorem 4.1 (see also Garfinkel, Kunnathur and Liepins, 1986).

Winkler (1995) notes that the upper bound on the computing time for the SCP integer programming problem is proportional to $2^t tm$, where t denotes the total number of (explicit and implicit) edits. So, the computing time can be reduced drastically if the number of implicit edits can be reduced.

4.4.2. Improved implicit edit generation

Winkler (1995) provides an adaptation of the Fellegi-Holt algorithm for categorical data sets. His result is an alternative for the first algorithm of Garfinkel, Kunnathur and Liepins. Winkler reduces the number of implicit edits needed to solve the categorical error localisation problem. For large problems Winkler's algorithm leads to a large reduction in computing time in comparison with the algorithm of Fellegi and Holt.

One of his observations concerns the so-called *redundant edits*. An edit E^r is called *redundant*, if there exists another edit E^d that *dominates* edit E^r, i.e. $P(E^r) \subseteq P(E^d)$. Note that E^d is failed, if E^r is failed. Note also that the entering fields of E^d are covered by the entering fields of E^r. So, if some set of variables S covers one of the entering variables of E^d, then S covers at least one of the entering variables of E^r. This implies that redundant edits are not needed to find sets of variables that cover the complete set of failed edits. Consequently, redundant edits are not needed to find all optimal solutions to the error localisation problem. Furthermore, Winkler (1995) observes that if E^d replaces E^r in a generating set of edits, then any generated edit would necessarily dominate the edit that would have been obtained if E^r had been used. This implies that redundant edits need not be included in contributing sets of edits. Thus we can conclude that redundant edits can be deleted from the set of edits.

Garfinkel, Kunnathur and Liepins (1986) observe that if one contributing set is a proper subset of another contributing set and if generating on field j (using Lemma 4.1) yields essentially new implicit edits for both contributing sets, then the edit generated using the larger contributing set is redundant to the one using the smaller set. Because redundant edits are not useful for error localisation purposes, only prime contributing sets, i.e. contributing sets for which no proper subset exists that is also a contributing set of an essentially new implicit edit, have to be used in Lemma 4.1.

Example 4.10:
Suppose $m = 2$, $D_1 = \{1,2,3\}$, $D_2 = \{1,2,3\}$ and

$$P(E^1) = \{1,2\} \times \{1,2,3\} \tag{4.17}$$

$$P(E^2) = \{1,3\} \times \{1,3\} \tag{4.18}$$

$$P(E^3) = \{2,3\} \times \{2,3\} \tag{4.19}$$

$$P(E^4) = \{1\} \times \{2\} \tag{4.20}$$

E^4 is redundant, since it is dominated by E^1. Suppose that Lemma 4.1 is used to generate (essentially new) implicit edits on generating field 1. Consider the contributing sets $S_1 = \{E^1, E^3\}$ and $S_2 = \{E^1, E^2, E^3\}$. Note that S_1 is a proper subset of S_2. Generating on field 1 using contributing set S_1 yields the essentially new implicit edit E^5, with $P(E^5) = \{1,2,3\} \times \{2,3\}$. The essentially new implicit edit E^6, with $P(E^6) = \{1,2,3\} \times \{3\}$ is obtained by generating on field 1 using contributing set S_2. Obviously, E^5 dominates E^6. ■

4.5. The Fellegi-Holt method for numerical and mixed data

In the previous sections we have explained how the Fellegi-Holt method works for categorical data. For numerical data the method works similarly. The only difference is the way in which the essentially new implicit edits are generated. To generate implicit edits for categorical data we can apply Lemma 4.1. It is clear that for numerical data, we cannot apply this lemma. In this section we start by explaining how implicit edits are generated in numerical data. Subsequently, we briefly discuss the Fellegi-Holt method for mixed data.

For purely numerical data, the edits are given either by inequalities or by equalities. To generate an implicit edit we first select a generating field. Next, we basically apply Fourier-Motzkin elimination (see Chvátal, 1983; Korte and Vygen, 2000; see also Chapter 8) to eliminate the generating variable from the set of edits.

We consider all edits involving the selected variable x_r pair-wise. Suppose edit *s* is given by

$$a_{1s}x_1 + \ldots + a_{ns}x_n + b_s = 0 \tag{4.21a}$$

or

$$a_{1s}x_1 + \ldots + a_{ns}x_n + b_s \geq 0\,, \tag{4.21b}$$

and edit *t* by

$$a_{1t}x_1 + \ldots + a_{nt}x_n + b_t = 0\,, \tag{4.22a}$$

or

$$a_{1t}x_1 + ... + a_{nt}x_n + b_t \geq 0 . \tag{4.22b}$$

We now construct an implicit edit. If (4.21) is an equality, we use the equality

$$x_r = -b_s - \frac{1}{a_{rs}} \left(\sum_{\substack{i=1 \\ i \neq r}}^{n} a_{is} x_i \right) \tag{4.23}$$

to eliminate x_r from (4.22).

If (4.21) is an inequality and (4.22) is an equality, then similarly we can use the equality in (4.22) to eliminate x_r. If (4.21) and (4.22) are inequalities, we check whether the coefficients of x_r in those inequalities have opposite signs. That is, we check whether $a_{rs} \times a_{rt} < 0$. If that is the case, we generate the following implicit edit not involving x_r:

$$\widetilde{a}_1 x_1 + ... + \widetilde{a}_n x_n + \widetilde{b} \geq 0 , \tag{4.24}$$

where

$$\widetilde{a}_i = | a_{rs} | a_{it} + | a_{rt} | a_{is} \quad \text{for all } i=1,\ldots,m \tag{4.25}$$

and

$$\widetilde{b} = | a_{rs} | b_t + | a_{rt} | b_s . \tag{4.26}$$

Apart from the different way in which implicit edits are generated, the Fellegi-Holt method for numerical data is the same as the Fellegi-Holt method for categorical data. Again, the complete set of edits is generated by repeatedly selecting a generating field. Subsequently, all pairs of edits (explicit or implicit) are considered, and it is checked whether new (essentially new) implicit edits can be obtained by eliminating the selected generating field from these pairs of edits. This process continues until no new (essentially new) implicit edits can be generating anymore, whatever generating field is selected. The complete set of edits has then been determined.

To illustrate, we give the example below. This example is basically an example provided by Fellegi and Holt (1976), except for the fact that in their paper the edits indicate a condition of edit failure (i.e. if a numerical condition holds true, the edit is violated), whereas here the edits indicate the opposite condition of edit consistency (i.e. if a numerical condition holds true, the edit is satisfied).

Example 4.11:
Suppose we have four numerical variables x_i (i=1,…,4). The explicit edits are given by:

$$x_1 - x_2 + x_3 + x_4 \geq 0 \tag{4.27}$$

and

$$-x_1 + 2x_2 - 3x_3 \geq 0\,. \tag{4.28}$$

The implicit edits are given by:

$$x_2 - 2x_3 + x_4 \geq 0\,, \tag{4.29}$$

$$x_1 - x_3 + 2x_4 \geq 0 \tag{4.30}$$

and

$$2x_1 - x_2 + 3x_4 \geq 0\,. \tag{4.31}$$

The above five edits form the complete set of edits as no new (essentially new) implicit edit can be generated anymore.

Now, suppose we are editing a record with values (3, 4, 6, 1), and suppose that the reliability weights are all equal to 1. Examining the explicit edits we see that the first edit is satisfied, whereas the second edit is violated. From the explicit edits it is not clear which of the fields should be changed. If we also examine the implicit edits, however, we see that edits (4.28), (4.29) and (4.30) fail. Variable x_3 occurs in all three violated edits. So, we can satisfy all edits by changing the value of x_3, for example x_3 could be made equal to 1. Changing the value of x_3 is the only optimal solution to this error localisation problem. ■

Fellegi and Holt do not give a proof that their method works for numerical data. In particular, they do not prove that a solution to the set-covering problem based on all explicit and essentially new implicit edits is also a solution to the error localisation problem. Fortunately, a proof is easy to provide by using a standard result for Fourier-Motzkin elimination (see e.g. Chvátal, 1983). Suppose a solution to the set-covering problem based on all explicit and essentially new implicit edits has been obtained. We then eliminate all variables involved in this solution from the explicit edits. The obtained set of (implicit) edits is satisfied, if the original values are filled in for all variables not involved in the solution to the set-covering problem. This follows from the construction of the set-covering problem, where violated (explicit and implicit) edits correspond to constraints for the set-covering problem. A standard result for Fourier-Motzkin elimination now says that the (explicit) edits can be satisfied by filling in appropriate values for the variables involved in the solution to the set-covering problem. A solution to the set-covering problem is therefore also a solution to error localisation problem.

Instead of applying the standard result for Fourier-Motzkin one could also apply the results of Chapter 8 in this book. Those results are more general than the mentioned standard result for Fourier-Motzkin elimination as they include categorical data.

An important practical problem of the Fellegi-Holt method for numerical data is that the number of implicit edits may be very high. We have already mentioned that this is a serious problem of the Fellegi-Holt method for categorical data. The reasons for the potentially large number of implicit edits for numerical data are again that an enormous amount of subsets of the set of variables have to be eliminated, and that the number of edits may increase whenever a variable (or subset of variables) is eliminated. As for

categorical data, in most practical situations the first reason is mainly responsible for the large size of the complete set of edits.

In fact, the size of the complete set of edits appears to be an even more serious problem for numerical data than for categorical data. For most real-life problems the number of implicit edits becomes extremely high even for a small to moderate number of explicit edits, as some early experiments at Statistics Netherlands quickly confirmed. Due to the practical and theoretical hopelessness of the situation, this path based on the Fellegi-Holt method was quickly abandoned for numerical data.

An exception to the rule that the number of implicit edits becomes extremely high is formed by ratio-edits, i.e. ratios of two numerical variables that are bounded by constant lower and upper bounds. For ratio-edits the number of implicit edits is low, and the Fellegi-Holt method can be applied in practice (see Winkler and Draper, 1997).

For mixed data, we can – in principle – also use the Fellegi-Holt method to solve the error localisation problem. However, the logic to generate the implicit edits becomes quite complex. This logic becomes so complex because in order to generate the complete set of edits categorical variables frequently have to be eliminated from edits in which still some numerical variables are involved. The resulting edits are quite complicated ones. In Section 8.5 we return to this point, be it in a somewhat different context. For this reason, and because the number of implicit edits can become very high just like for purely numerical data, we do not consider the Fellegi-Holt method for mixed data in further detail in this book.

Remark: We would like to end this section with a remark on the history of Fourier-Motzkin elimination. In the early 19th century Fourier became interested in systems of linear inequalities. In particular, Fourier was interested in the problem whether a feasible solution to a specified set of linear inequalities exists. Fourier developed an interesting method for solving this problem based on successively eliminating variables from the system of inequalities. This method was later called Fourier-Motzkin elimination. It was published for the first time in 1826 (see Fourier, 1826, and Kohler, 1973). Fourier himself was not too impressed by his discovery. In his book on determinate equations, "*Analyse des equations déterminées*", that was published after his death in 1831 this method for eliminating variables was omitted.

Fourier's method eliminates variables from a system of linear inequalities and equalities. The number of linear inequalities and equalities may change while eliminating variables. This may be a problem for large systems. For large systems the number of linear (in)equalities often becomes so high that Fourier-Motzkin elimination becomes impractical, i.e. too much computing time and computer memory is required to apply Fourier-Motzkin elimination.

There is a dual version of Fourier-Motzkin elimination. Here linear (in)equalities are eliminated rather than variables. The number of variables may vary while eliminating constraints. The dual version of Fourier-Motzkin elimination suffers from the same problem as the primal version: for large systems the number of variables often becomes so high that the method is impractical.

For many years Fourier's method was forgotten. In the 20th century it was rediscovered several times. For instance, Motzkin (1936) rediscovered the method. For this reason the method is usually referred to as Fourier-Motzkin elimination. Other people who have rediscovered the method are Dines (1927) and Chernikova (1964, 1965). Chernikova in fact rediscovered the dual version of Fourier-Motzkin elimination (see also Chapter 5 of this book).

In 1976 Fellegi and Holt again rediscovered Fourier-Motzkin elimination. Fellegi and Holt, however, not only rediscovered Fourier-Motzkin elimination for numerical data, they also extended the method to categorical variables. This extension is similar to the resolution technique known from machine learning and mathematical logic (see for example Robinson, 1965 and 1968; Russell and Norvig, 1995; Williams and Brailsford, 1996; Marriott and Stuckey, 1998; Warners, 1999; Ben-Ari, 2001).

In the present book Fourier-Motzkin elimination and its extension to categorical data play a major role. The dual version of Fourier-Motzkin elimination is used in Chapter 5. The primal version of Fourier-Motzkin elimination and its extension to categorical data are used in Chapters 8 and 10. Finally, an extension of Fourier-Motzkin elimination to integer data is described and applied in Chapter 9.

4.6. Discussion

In this chapter we have described the Fellegi-Holt method for solving the error localisation problem. Hereby we have focussed on the method for categorical data. The reason for this is that the Fellegi-Holt method seems to be more suitable for categorical data than for numerical data. In practice, the number of implicit edits for numerical data appears to be too high for the Fellegi-Holt method to be applicable.

For mixed data the situation is even worse. Not only can the number of implicit edits become extremely high, but also the logic to generate implicit edits becomes complicated. Due to this complexity it is unlikely that implicit edits can be generated quickly. We can conclude that for mixed data the Fellegi-Holt method offers little promise at the moment. Therefore, in subsequent chapters of this book we will not consider the Fellegi-Holt method; instead we will examine several other methods for solving the error localisation problem in mixed data.

We end this chapter with the remark that although we do not consider the Fellegi-Holt method a promising method for solving the error localisation problem in mixed data, we do consider several ideas on which this method is based to be very important ones. In Chapter 8 we will again use the concept of implicit edits. There we will not generate the complete set of edits, but will restrict ourselves to (small) subsets. In Chapter 9 we will also use the lifting principle introduced in the present chapter extensively.

5. Vertex Generation Methods

5.1. Introduction

A well-known method for solving the error localisation problem for numerical (continuous) data is based on vertex generation techniques, in particular the algorithm of Chernikova. In this chapter we examine how vertex generation methods can be extended to solve the error localisation problem in a combination of categorical (discrete) and numerical data. The material contained in this chapter is for a substantial part based on many excellent papers, such as Chernikova (1964 and 1965), Duffin (1974), Rubin (1975 and 1977), Sande (1978a), Schiopu-Kratina and Kovar (1989), and Fillion and Schiopu-Kratina (1993).

Only the last three of these papers focus on the error localisation problem. The other papers describe general algorithms. Chernikova (1964 and 1965) describes an algorithm to determine the edges of a convex polyhedral cone in the nonnegative orthant with a vertex at the origin. Rubin (1975 and 1977) extends this algorithm to related problems. Duffin (1974) gives a thorough description of Fourier-Motzkin elimination. Chernikova's algorithm is in fact the dual of Fourier-Motzkin elimination.

Sande (1978a) discusses the error localisation problem for numerical data, and briefly also the error localisation problems for categorical data and mixed data. He proposes methods based on vertex generation in order to solve these problems. He does not provide details of the method for categorical and numerical data, however. Schiopu-Kratina and Kovar (1989) and Fillion and Schiopu-Kratina (1993) propose a number of improvements on Sande's method for solving the error localisation problem for numerical data. They, however, do not consider the error localisation problems for categorical or mixed data.

The main aim of this chapter is not to present new results, but rather to combine the ideas of the above-mentioned papers in order to give a "complete", self-contained description of the use of vertex generation methods in order to solve the error localisation problem in mixed data. In particular, we describe how the method based on vertex generation can be applied to identify errors in a mix of categorical and numerical data. In doing so, we give a detailed account of the method for the first time in literature. Furthermore, we show that many results for numerical data also hold true for a mix of categorical and numerical data. We will especially describe how improvements suggested by Schiopu-Kratina and Kovar (1989) and Fillion and Schiopu-Kratina (1993) on Sande's method for solving the error localisation problem for continuous data can also be used for mixed data. This had not yet been pointed out in literature before. Finally, we mention that vertex generation methods other than Chernikova's algorithm may be used to solve the error localisation problem. As far as we know this had not yet been noted in existing literature on the error localisation problem.

The remainder of the present chapter is organised as follows. Section 5.2 shows that the error localisation problem can be solved by generating the vertices of a certain polyhedron.

Unfortunately, the number of vertices of this polyhedron is often too high for this approach to be applicable in practice. Instead, one should generate a suitable subset of the vertices only. There are a number of vertex generation algorithms that efficiently generate such a suitable subset of the vertices of a polyhedron. A vertex generation algorithm that has been proposed by Chernikova (1964, 1965) has been used in several computer systems for automatic edit and imputation of numerical data, for example GEIS (Kovar and Whitridge, 1990; Pierzchala, 1995), CherryPi (De Waal, 1996; De Waal and Van de Pol, 1997), AGGIES (Todaro, 1999) and a SAS program developed by the Central Statistical Office of Ireland (see Central Statistical Office, 2000).

As noted above, the original algorithm of Chernikova was designed for calculating the edges of a convex polyhedral cone. When applied to solve the error localisation problem, this general algorithm turns out to be rather slow, as could be expected. The algorithm has been accelerated by various modifications (see Rubin, 1975 and 1977; Sande, 1978a; Schiopu-Kratina and Kovar, 1989; Fillion and Schiopu-Kratina, 1993). We describe Chernikova's algorithm and the above-mentioned modifications in Sections 5.3 to 5.5.

An alternative to Chernikova's algorithm is an algorithm described by Duffin (1974). That algorithm is very similar to Chernikova's algorithm. The two algorithms differ only with respect to a few technical details (see e.g. Houbiers, 1999b). We describe Duffin's algorithm and the differences with Chernikova's algorithm in Section 5.6 to 5.8.

Chernikova's algorithm and Duffin's algorithm have been designed to handle inequalities. In practice, however, many edits involve equalities rather than inequalities. Of course, we could represent each equality by two inequalities, and then use the algorithms proposed by Chernikova or Duffin to solve the error localisation problem. That would be a rather inefficient approach, however. A better approach is to treat equalities in a slightly different way than inequalities. In Section 5.9 we describe how Chernikova's algorithm and Duffin's algorithm can be modified in order to deal with equalities in an efficient manner.

Section 5.10 concludes the chapter with a brief discussion. Before we proceed we would like to remark that for some theorems mentioned in this chapter - short ones that provide an insight into the method - we give the proof. For theorems for which the proof is rather long and technical, we omit the proof, as such a proof would not give further insight into the method. The interested reader is referred to the literature for those proofs.

This chapter is based on De Waal (2000b). Part of this material is planned to be published in De Waal (2003).

5.2. Vertex generation methods and error localisation for mixed data

In this section we explain why vertex generation methods can be used to solve the error localisation problem in mixed data. An optimal solution to the error localisation problem corresponds to a minimum of

$$\sum_{i=1}^{m} w_i \left(\sum_k e_{ik}^N \right) + \sum_{i=1}^{n} w_{m+i} \left(\delta(z_i^P) + \delta(z_i^N) \right), \tag{5.1}$$

where w_i is the reliability weight of variable *i*, $\delta(x)=1$ if and only if $x \neq 0$ and $\delta(x)=0$ otherwise, is minimised subject to the constraints:

$$e_{ik}^P, e_{ik}^N \in \{0,1\}, \qquad (i=1,...,m) \qquad (5.2)$$

$$e_{ik}^N = 0 \text{ if } \gamma_{ik}^0 = 0, \qquad (i=1,...,m) \qquad (5.3)$$

$$z_i^P, z_i^N \geq 0, \qquad (i=1,...,n) \qquad (5.4)$$

$$\sum_k e_{ik}^P \leq 1, \qquad (i=1,...,m) \qquad (5.5)$$

$$e_{ik}^P + e_{ik}^N \leq 1 \qquad (i=1,...,m) \qquad (5.6)$$

$$\sum_k (\gamma_{ik}^0 + e_{ik}^P - e_{ik}^N) = 1, \qquad (i=1,...,m) \qquad (5.7)$$

$$\alpha_i \leq x_i^0 + z_i^P - z_i^N \leq \beta_i \qquad (i=1,...,n) \qquad (5.8)$$

and

$$\sum_{i=1}^n a_{ij}(x_i^0 + z_i^P - z_i^N) + b_j \geq M\left(\sum_{i=1}^m \left(\sum_{c_{ik} \in F_i^j} (\gamma_{ik}^0 + e_{ik}^P - e_{ik}^N) - 1\right)\right) \qquad (5.9)$$

for all edits *j*=1,...,*K* (see Section 3.4). Suppose a minimum of (5.1) subject to (5.2) to (5.9) is attained in a point given by:

1. $e_{ik}^N = 0$ for $(i,k) \in I_e^N$, $e_{ik}^N = 1$ otherwise,
2. $e_{ik}^P = 0$ for $(i,k) \in I_e^P$, $e_{ik}^P = 1$ otherwise,
3. $z_i^P = 0$ for $i \in I_z^P$, $z_i^P \neq 0$ otherwise,

and

4. $z_i^N = 0$ for $i \in I_z^N$, and $z_i^N \neq 0$ otherwise,

where I_e^N, I_e^P, I_z^P and I_z^N are certain index sets.

We now consider the problem of minimising the linear function

$$\sum_{(i,k) \in I_e^N} e_{ik}^N + \sum_{(i,k) \notin I_e^N} (1 - e_{ik}^N) + \sum_{(i,k) \in I_e^P} e_{ik}^P + \sum_{(i,k) \notin I_e^P} (1 - e_{ik}^P) + \sum_{i \in I_z^P} z_i^P + \sum_{i \in I_z^N} z_i^N \qquad (5.10)$$

subject to (5.3) to (5.9) and

$$0 \leq e_{ik}^N, e_{ik}^P \leq 1. \qquad (5.11)$$

Note that all variables are non-negative by (5.4) and (5.11), and that the e_{ik}^{P} and e_{ik}^{N} are at most equal to one by (5.11). Hence, function (5.10) is non-negative. Its value equals zero only for a point satisfying 1 to 4 above.

It is well-known that a linear function subject to a set of linear constraints attains its minimum, if such a minimum exists, in a vertex of the feasible polyhedron described by the set of linear constraints (see e.g. Chvátal, 1983; Hillier and Lieberman, 1995). So, the minimum of (5.10), zero, is attained in a vertex of the feasible polyhedron described by (5.2) to (5.9) and (5.11). We can conclude that a point satisfying by 1 to 4 above, i.e. an optimum of (5.1) subject to (5.2) to (5.9), is a vertex of the polyhedron defined by (5.3) to (5.9) and (5.11).

The above observation implies that the minimum of (5.1) subject to (5.2) to (5.9) can be found by generating all vertices of the feasible polyhedron given by (5.3) to (5.9) and (5.11). From these vertices we select the vertices that satisfy (5.2). From those vertices we subsequently select the vertices for which the value of objective function (5.1) is minimal. These vertices correspond to the optimal solutions to the error localisation problem.

5.3. Chernikova's algorithm

We start this section by describing Chernikova's algorithm. Subsequently, we explain how it can be used to determine the vertices of a polyhedron. Chernikova's algorithm is in fact designed to generate the edges of a system of linear inequalities given by

$$\mathbf{Cx} \geq \mathbf{0} \tag{5.12}$$

and

$$\mathbf{x} \geq \mathbf{0}\,, \tag{5.13}$$

where **C** is a constant $n_r \times n_c$-matrix and **x** an n_c-dimensional vector of unknowns.

Rubin's formulation (Rubin, 1975 and 1977) of Chernikova's algorithm is as follows:

1. Construct the $(n_r + n_c) \times n_c$-matrix $\mathbf{Y^0} = \begin{pmatrix} \mathbf{U^0} \\ \mathbf{L^0} \end{pmatrix}$, where $\mathbf{U^0} = \mathbf{C}$ and $\mathbf{L^0} = \mathbf{I_{n_c}}$: the $n_c \times n_c$-identity matrix.
2. $k := 0$
3. If any row of $\mathbf{U^k}$ has all components negative, set k equal to n_r. In this case $\mathbf{x} = \mathbf{0}$ is the only point satisfying (5.12) and (5.13).
4. If all the elements of $\mathbf{U^k}$ are non-negative, then set k equal to n_r. The columns of $\mathbf{L^{n_r}}$ are the edges of the cone described by (5.12) and (5.13).
5. If neither 3 nor 4 holds: choose a row of $\mathbf{U^k}$, say row r, with at least one negative entry.

6. Let $R = \{j \mid y_{rj}^k \geq 0\}$. Let v be the number of elements in R. Then the first v columns of the new matrix $\mathbf{Y^{k+1}}$ are all the columns y_{*j}^k of $\mathbf{Y^k}$ for $j \in R$, where y_{*j}^k denotes the j-th column of $\mathbf{Y^k}$.

7. Examine the matrix $\mathbf{Y^k}$.

   a. If $\mathbf{Y^k}$ has only two columns and $y_{r1}^k \times y_{r2}^k < 0$, then choose $\mu_1, \mu_2 > 0$ such that $\mu_1 y_{r1}^k + \mu_2 y_{r2}^k = 0$. Adjoin the column $\mu_1 y_{*1}^k + \mu_2 y_{*2}^k$ to Y^{k+1}. Go to Step 9.

   b. If $\mathbf{Y^k}$ has more than two columns then let $S = \{(s,t) \mid y_{rs}^k \times y_{rt}^k < 0 \text{ and } t > s\}$, i.e. let S be the set of all pairs of columns of $\mathbf{Y^k}$ whose elements in row r have opposite signs. Let I_0 be the index set of all non-negative rows of $\mathbf{Y^k}$, i.e. all rows of $\mathbf{Y^k}$ with only non-negative entries. For each $(s,t) \in S$, find all $i \in I_0$ such that $y_{is}^k = y_{it}^k = 0$. Call this set $I_1(s,t)$.

      - If $I_1(s,t) = \varnothing$, then y_{*s}^k and y_{*t}^k do not contribute another column to the new matrix.
      - If $I_1(s,t) \neq \varnothing$, check to see if there a v not equal to s or t such that $y_{iv}^k = 0$ for all $i \in I_1(s,t)$. If such a v exists, then y_{*s}^k and y_{*t}^k do not contribute a column to the new matrix. If no such v exists, then choose $\mu_1, \mu_2 > 0$ such that $\mu_1 y_{rs}^k + \mu_2 y_{rt}^k = 0$. Adjoin the column $\mu_1 y_{*s}^k + \mu_2 y_{*t}^k$ to $\mathbf{Y^{k+1}}$.

8. When all pairs in S have been examined, and the additional columns (if any) have been added, we say that row r has been processed. We then define matrices $\mathbf{U^{k+1}}$ and $\mathbf{L^{k+1}}$ by $\mathbf{Y^{k+1}} = \begin{pmatrix} \mathbf{U^{k+1}} \\ \mathbf{L^{k+1}} \end{pmatrix}$, where $\mathbf{U^{k+1}}$ is a matrix with n_r rows and $\mathbf{L^{k+1}}$ a matrix with n_c rows.

9. $k := k + 1$, and go to Step 3.

In Chernikova's original papers the transpose of the matrix $\mathbf{Y^k}$ is used, and the processing is done column-wise rather than row-wise as in Rubin's formulation. The formulation given here is used throughout this chapter, because this formulation is used most frequently in literature (see Rubin, 1975; Sande, 1978a; Schiopu-Kratina, 1989; Fillion and Schiopu-Kratina, 1993; De Waal, 2000b).

Chernikova's algorithm can be used to find the vertices of a system of linear inequalities because of the following lemma (see Rubin, 1975 and 1977).

Lemma 5.1: The vector $\mathbf{x^0}$ is a vertex of the system of linear inequalities

$$\mathbf{A}\mathbf{x} \le \mathbf{b} \tag{5.14}$$

and

$$\mathbf{x} \ge \mathbf{0} \tag{5.15}$$

if and only if $\left\{(\lambda \mathbf{x^0} \mid \lambda)^T, \lambda \ge 0\right\}$ is an edge of the cone described by

$$(-\mathbf{A} \mid \mathbf{b})\begin{pmatrix}\mathbf{x}\\ \xi\end{pmatrix} \ge \mathbf{0} \tag{5.16}$$

and

$$\begin{pmatrix}\mathbf{x}\\ \xi\end{pmatrix} \ge \mathbf{0}\,. \tag{5.17}$$

Here $\mathbf{A}$ is an $n_r \times n_v$-matrix, $\mathbf{b}$ an n_r-vector, $\mathbf{x}$ an n_v-vector, and ξ and λ scalar variables.

For notational convenience we write

$$n_c = n_v + 1 \tag{5.18}$$

throughout this chapter. The matrix in (5.16) is then an $n_r \times n_c$-matrix just like in (5.12), so we can use the same notation as in Rubin's formulation of Chernikova's algorithm.

If Chernikova's algorithm is used to determine the edges of (5.16) and (5.17), then after termination of the algorithm the vertices of (5.14) and (5.15) correspond to those columns j of $\mathbf{L^{n_r}}$ for which $l^{n_r}_{n_c,j} \ne 0$. The entries of such a vertex $\mathbf{x'}$ are given by

$$x'_i = l^{n_r}_{ij} \Big/ l^{n_r}_{n_c,j} \qquad \text{for } i=1,\ldots,n_v\,. \tag{5.19}$$

Example 5.1:
Suppose we want to use Chernikova's algorithm to determine the vertices of the following system:

$$\begin{aligned} x_1 + x_2 - 2x_3 &\le 1\\ 5x_1 - 2x_2 + 2x_3 &\le 3\\ 3x_1 + x_2 - 4x_3 &\le 2\\ -x_1 - x_2 + 2x_3 &\le -1 \end{aligned} \tag{5.20}$$

where the x_i are non-negative.

The matrix $\mathbf{Y^0}$ is then given by

$$\mathbf{Y^0} = \left(\begin{array}{cccc} -1 & -1 & 2 & 1 \\ -5 & 2 & -2 & 3 \\ -3 & -1 & 4 & 2 \\ 1 & 1 & -2 & -1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right). \tag{5.21}$$

The horizontal line separates the upper matrix $\mathbf{U^0}$ from the lower matrix $\mathbf{L^0}$.

We process the first row, and obtain the following result.

$$\mathbf{Y^1} = \left(\begin{array}{cccccc} 2 & 1 & 0 & 0 & 0 & 0 \\ -2 & 3 & -12 & -2 & 2 & 5 \\ 4 & 2 & -2 & -1 & 2 & 1 \\ -2 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{array}\right). \tag{5.22}$$

For instance, the first two columns of $\mathbf{Y^1}$ are obtained by copying the last two columns of $\mathbf{Y^0}$. The third column of $\mathbf{Y^1}$ is obtained by adding the third column of $\mathbf{Y^0}$ to two times the first column of $\mathbf{Y^0}$. In a similar way the other columns of $\mathbf{Y^1}$ are obtained.

We now process the fourth row. The result is:

$$\mathbf{Y^2} = \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ -12 & -2 & 2 & 5 \\ -2 & -1 & 2 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 2 & 1 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{array}\right). \tag{5.23}$$

Processing the third row yields:

$$\mathbf{Y^3} = \left(\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 2 & 5 & -10 & 3 \\ 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 2 & 1 \\ 2 & 1 & 2 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{array}\right). \tag{5.24}$$

Note that columns 1 and 4 of $\mathbf{Y^2}$ are not combined into a column of $\mathbf{Y^3}$ (see Step 7b of Chernikova's algorithm). Likewise, columns 2 and 3 of $\mathbf{Y^2}$ are not combined into a column of $\mathbf{Y^3}$.

Finally, we process the second row and obtain the final matrix

$$\mathbf{Y^4} = \left(\begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 3 & 0 & 0 \\ 2 & 1 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 2 & 16 \\ 2 & 1 & 1 & 12 & 16 \\ 1 & 0 & 0 & 7 & 6 \\ 0 & 1 & 2 & 0 & 20 \end{array}\right). \tag{5.25}$$

Note that columns 2 and 3 of $\mathbf{Y^3}$ are not combined into a column of $\mathbf{Y^4}$.

In matrix $\mathbf{Y^4}$ we can see that the vertices of system (5.20) are given by (0, 1, 0), (0.5, 0.5, 0) and (0.8, 0.8, 0.3) (see (5.19)). ■

5.4. Chernikova's algorithm and the error localisation problem

In the previous section we have described Chernikova's algorithm. In this section we explain how this algorithm can be used to solve the error localisation problem in mixed data.

The set of constraints (5.3) to (5.9) and (5.11) can be written in the form (5.14) and (5.15). We can find the vertices of the polyhedron corresponding to this set of constraints by applying Chernikova's algorithm to (5.16) and (5.17). Vertices of the polyhedron defined by (5.3) to (5.9) and (5.11) are given by columns $y_{*s}^{n_r}$ for which $u_{is}^{n_r} \geq 0$ for all i and $l_{n_c,s}^{n_r} > 0$, where n_c is the number of rows of the final matrix $\mathbf{L^{n_r}}$. In our case, n_c equals the total number of variables z_i^P, z_i^N, e_{ik}^P and e_{ik}^N plus one (corresponding to ξ in (5.16)

and (5.17)), i.e. $n_c = 2n + 2G + 1$, where $G = \sum_i g_i$. The values of the variables z_i^P, z_i^N, e_{ik}^P and e_{ik}^N in such a vertex are given by the corresponding values $l_{js}^{n_r} \Big/ l_{n_c,s}^{n_r}$.

In this way all vertices of the polyhedron defined by (5.3) to (5.9) and (5.11) can, in principle, be determined. However, because there may be extremely many vertices, determining all vertices of the feasible polyhedron is not an efficient way for solving the error localisation problem. McMullen (1970) gives the following, tight upper bound on the number of vertices for a set of *s* linear inequalities involving *t* unrestricted variables:

$$f(s,t) = \binom{s - \left\lfloor \frac{t+1}{2} \right\rfloor}{s-t} + \binom{s - \left\lfloor \frac{t+2}{2} \right\rfloor}{s-t}, \tag{5.26}$$

where $\lfloor x \rfloor$ stands for the largest integer smaller than x.

Even more important than the number of vertices in the worst case, is the number of vertices for a "typical" polyhedron. Prékopa (1972) shows that the expected number of vertices for a set of *s* linear inequalities involving *t* variables, assuming a simple chance mechanism for generating the *s* linear inequalities, is given by

$$g(s,t) = \binom{s}{t} 2^{t-s}. \tag{5.27}$$

Both the number of vertices in the worst case and the number of vertices of a typical polyhedron can be very high.

Because generating all vertices is likely to be far too demanding from a computational point of view, we aim to limit the number of vertices that are generated as much as possible. This can be done in the following way. If a (possibly suboptimal) solution to the error localisation problem has been found for which the value of the objective function equals η, then we look only for vertices corresponding to solutions that are at least as good, i.e. solutions for which the value of the objective function is at most equal to η.

A minor technical problem is that we cannot use objective function (5.1) directly when applying Chernikova's algorithm, because each e_{ik}^N, each z_i^P and each z_i^N has been replaced by several variables. Therefore, we need to introduce a new objective function. This objective function associates a value to each column of the matrix $\mathbf{Y^k}$. Assume that the first *G* entries of a column l_{*s}^k of $\mathbf{L^k}$ correspond to the e_{ik}^N-variables, the next *G* entries to the e_{ik}^P-variables, the next *n* entries to the z_i^N-variables, and the following *n* entries to the z_i^P-variables. We define the following objective function

$$\sum_{i=1}^{m} w_i \left(\sum_{k=1}^{g_i} \delta(l_{j,s}^k) \right) + \sum_{i=1}^{n} w_{m+i} \left(\delta(l_{2G+i,s}^k) + \delta(l_{2G+n+i,s}^k) \right), \tag{5.28}$$

where $j = \sum_{l=1}^{i-1} g_l + r$ for each combination of i and r (i=1,…,m; r=1,…, g_i). If column y_{*s}^k of $\mathbf{Y^k}$ corresponds to a solution to the error localisation problem, then the value of the objective function (5.28) for y_{*s}^k equals the value of the objective function (5.1) for this solution to the error localisation problem. This implies that we can use the objective function (5.28) to update the value of η.

Now we are ready to explain how Chernikova's algorithm can be used to solve the error localisation problem for mixed data. Two technical problems must be overcome. First, the algorithm must be sufficiently fast. Second, the solution found must be *feasible* for the error localisation problem for mixed data, i.e. the values of the variables e_{ik}^P and e_{ik}^N must be either 0 or 1. Both problems can be overcome by removing certain "undesirable" columns from the current matrix $\mathbf{Y^k}$, i.e. by deleting columns that cannot yield an optimal solution to the error localisation problem. That such undesirable columns may indeed be removed from the current matrix $\mathbf{Y^k}$ is essentially demonstrated by Rubin (1975 and 1977). We state this result as Theorem 5.1.

Theorem 5.1: Columns that cannot yield an optimal solution to the error localisation problem because they contain too many non-zero entries may be removed from an intermediate matrix.

Proof. Suppose we conclude that a certain column c in an intermediate matrix cannot yield an optimal solution. We cannot immediately conclude that we can remove this column, because this column might later be used in Step 7 of Chernikova's algorithm to prevent the generation of other columns. We note, however, that those columns have non-zero entries in the same positions as c, and perhaps more (see Step 7 of Chernikova's algorithm). This implies that those columns cannot yield optimal solutions either, just like c itself. That is, any column that might be generated because column c has been removed, may itself be removed because it cannot generate an optimal solution to the error localisation problem. ■

The computing time of Chernikova's algorithm can obviously be reduced by removing all columns for which the value of the objective function (5.28) is larger than η from the current matrix $\mathbf{Y^k}$ each time Step 9 has been executed. These columns can never generate a solution to the error localisation problem that is at least as good as the best one found so far, because the value of the objective function (5.28) of a combination of two columns (see Step 7) is at least equal to the maximum of the values of the objective function (5.28) for these columns separately.

Initially, the value of η can be set to a certain number η_0. All records for which the optimal value of (5.1) exceeds η_0 will then not be edited automatically. Those records are considered too erroneous for automatic correction, and have to be edited interactively or have to be discarded completely.

The computing time of Chernikova's algorithm can also be reduced by noting that in an optimal solution to the error localisation problem either $z_i^P = 0$ or $z_i^N = 0$ (or both). This implies that if in Step 7 of Chernikova's algorithm the entry of y_{*s}^k corresponding to z_i^P differs from zero and the entry of y_{*t}^k corresponding to z_i^N differs from zero as well, then columns y_{*s}^k and y_{*t}^k need not be combined to generate a new column. Fillion and Schiopu-Kratina (1993) report a reduction in computing time of approximately 45% when this 'complementary condition' is used. The experience at Statistics Netherlands shows a similar reduction in computing time.

We now consider the problem of constructing a feasible solution to the error localisation problem for mixed data. We have to ensure that for each variable *i* (*i*=1,...,*m*) at most one e_{ik}^P differs from zero, and that the e_{ik}^N and e_{ik}^P equal either zero or one after the termination of the algorithm. We can ensure that for each *i* at most one e_{ik}^P differs from zero in the following way. If in Step 7 of Chernikova's algorithm the entry of y_{*s}^k corresponding to $e_{ik_1}^P$ differs from zero and the entry of y_{*t}^k corresponding to $e_{ik_2}^P$ $(k_2 \neq k_1)$ differs from zero as well, then columns y_{*s}^k and y_{*t}^k are not combined to generate a new column.

We can also ensure that the e_{ik}^N equal either zero or one after the termination of the algorithm. For each *i* this is a problem only for the unique $e_{ik_0}^N$ for which $\gamma_{ik_0}^0 = 1$, namely the e_{ik}^N for which $k \neq k_0$ equal zero because of relation (5.3). We introduce binary variables $\tilde{e}_i$. These variables have to satisfy

$$e_{ik_0}^N + \tilde{e}_i = 1 . \qquad (5.29)$$

Relation (5.29) is treated as a constraint for the values of binary variables $e_{ik_0}^N$ and $\tilde{e}_i$. Because the values of $e_{ik_0}^N$ and $\tilde{e}_i$ have to be zero or one, we have to demand that either $e_{ik_0}^N = 0$ or $\tilde{e}_i = 0$. This can be ensured in the same manner as for the z_i^P and the z_i^N.

Finally, we have to ensure that the e_{ik}^P equal either zero or one after the termination of the algorithm. This is automatically the case if for each *i* only one e_{ik}^P differs from zero, at most one e_{ik}^N equals one, and the rest of the e_{ik}^N equal zero, because relation (5.7) has to hold true. We have already ensured that the above conditions are satisfied, so all e_{ik}^P equal zero or one after the termination of the algorithm. With the adaptations described above Chernikova's algorithm can be applied to solve the error localisation problem in mixed data.

Examining the algorithm described so far one might think that in a computer program the binary variables that are used for the categories of the discrete variables should be processed as bits rather than as variables that allow arithmetic operations. Sande (1978a),

however, disagrees and points out the advantage of treating the binary variables in the latter way. Using arithmetic operations on binary variables is sometimes called pseudo-Boolean programming to distinguish it from Boolean programming that uses logical operations. Pseudo-Boolean programming allows the combination of several Boolean edits into a single edit. For instance, edits expressing that at most one person in a household can be married to the head of the household can readily be expressed in a single pseudo-Boolean edit. In the Boolean format, edits for all pairs of persons would have to be constructed.

5.5. Adapting Chernikova's algorithm to the error localisation problem

In this section we consider more advanced adaptations of Chernikova's algorithm in order to make the algorithm better suited for solving the error localisation problem. Sande (1978a) notes that when two columns in the initial matrix $\mathbf{Y^0}$ have exactly the same entries in the upper matrix $\mathbf{U^0}$, they will be treated exactly the same in the algorithm. The two columns are always combined with the same other columns, and never with each other. Keeping both columns in the matrix only makes the problem unnecessarily bigger. One of the columns may therefore be temporarily deleted. After the termination of the algorithm, the combinations with the temporarily deleted column can easily be generated.

Suppose, for example, that for columns *s* and *t* we have $u^0_{js} = u^0_{jt}$ for all *j*. We can then temporarily delete one of these columns, say column *t*. Thus, we obtain a reduced matrix $\widetilde{\mathbf{Y}}^{\mathbf{0}}$ with upper matrix $\widetilde{\mathbf{U}}^{\mathbf{0}}$ and lower matrix $\widetilde{\mathbf{L}}^{\mathbf{0}}$. If Chernikova's algorithm applied to the reduced matrix finds a vertex given by

$$\sum_i \lambda_i \widetilde{l}^0_{*i} \tag{5.30}$$

then the vertices corresponding to the original matrix are given by

$$\sum_i \lambda_i l^0_{*i} \tag{5.31}$$

and if $\lambda_s \neq 0$ also by

$$\sum_{i \neq s} \lambda_i l^0_{*i} + \lambda_s l^0_{*t} \,. \tag{5.32}$$

Fillion and Schiopu-Kratina (1993) define a number of important concepts. These concepts allow several of their improvements to be explained easily. Suppose again that the upper matrix $\mathbf{U^k}$ and lower matrix $\mathbf{L^k}$ of a matrix $\mathbf{Y^k}$ have n_r and n_c rows, respectively. A *correction pattern* associated with column y^k_{*s} in an intermediate matrix $\mathbf{Y^k}$ is defined as the n_c-dimensional vector with entries $\delta(y^k_{js})$ for $n_r < j \leq n_r + n_c$, $\delta(y^k_{js})$ equals one if $y^k_{js} \neq 0$ and $\delta(y^k_{js})$ equals zero if $y^k_{js} = 0$. We have already noted that in the error localisation problem it suffices to find all correction patterns that correspond to vertices for

which (5.28), or equivalently (5.1), is minimal. That is, not all vertices for which (5.28) is minimal have to be found.

Sande (1978a) notes that Theorem 5.1 implies that once a vertex has been found, all columns with correction patterns with ones on the same places as in the correction pattern of this vertex can be removed.

The concept of correction patterns has been extended by Fillion and Schiopu-Kratina, who note that it is not important how the value of a variable is changed, but only whether the value of a variable is changed or not. A *generalised correction pattern* associated with column $y^k_{*_s}$ in an intermediate matrix $\mathbf{Y^k}$ is defined as the $(m + n)$-dimensional vector of which the j-th entry equals one if and only if the column $y^k_{*_s}$ indicates that the value of the j-th variable should be changed, and zero otherwise. Here m denotes the number of categorical variables and n the number of numerical variables. Again Theorem 5.1 implies that once a vertex has been found, all columns with generalised correction patterns with ones on the same places as in the generalised correction pattern of this vertex can be deleted.

Fillion and Schiopu-Kratina call a column of which the last entry is non-zero a *generator*. This is quite an understandable definition. In Chernikova's algorithm vertices and extremal rays are constructed by combining columns in intermediate matrices. To generate a vertex at least one generator has to be combined with other columns. It is impossible to generate a vertex by combining only columns that are non-generators.

Fillion and Schiopu-Kratina define a *failed row* as a row that contains at least one negative entry placed on a generator. They note that in order to solve the error localisation problem we can already terminate Chernikova's algorithm as soon as all failed rows have been processed. This is Theorem 5.2.

Theorem 5.2: If an intermediate matrix contains no failed rows, then all (generalised) patterns corresponding to vertices for which (5.28) is minimal have been found.

Proof. Suppose the theorem does not hold true, and there is a vertex for which (5.28) is minimal and with a new optimal generalised pattern that has not yet been generated. This vertex can only be generated by combining a generator c that does not correspond to a vertex (because if this generator were a vertex either the value of (5.28) would increase or the same generalised pattern would be repeated) with other columns. This implies, however, that c contains a negative entry. Hence the intermediate matrix contains a failed row, contradicting the assumption. ■

Fillion and Schiopu-Kratina (1993) report a reduction in computing time by a factor 60! The results at Statistics Netherlands were in the same order of magnitude, but slightly less spectacular. At Statistics Netherlands the reduction in computing time was approximately a factor 10 to 15. The disparity with the results of Fillion and Schiopu-Kratina may be explained by differences in the implementation details of the programs at Statistics Canada and Statistics Netherlands.

In Step 5 of Chernikova's algorithm a failed row should be selected, but we have not yet specified which failed row. In principle, the failed row that will be processed may be chosen arbitrarily. The efficiency of the algorithm depends strongly on the order in which the rows are processed, however. Rubin (1975) and Schiopu-Kratina and Kovar (1989) propose a simple rule for selecting the row that will be processed. Suppose that a failed row has z entries equal to zero, p positive entries and q negative entries, then processing this row may at worst lead to a matrix with $z + p + pq$ columns. The proposed rule says that a failed row should be selected to be processed for which the maximum number of resulting columns is as low as possible. If there are several failed rows that may be selected according to this rule, one of them is selected randomly.

The final adaptation of Fillion and Schiopu-Kratina to Chernikova's algorithm in order to solve the error localisation problem more efficiently is a method to speed-up the algorithm for records with missing values. Suppose the error localisation problem has to be solved for a record with missing values. In the standard algorithm described so far these missing values are replaced by values that are not allowed. Subsequently, the standard algorithm is applied. In the adapted algorithm proposed by Fillion and Schiopu-Kratina the variables with missing values are identified before Chernikova's algorithm starts. These variables will be part of the solution to the error localisation problem anyway. For each numerical variable an arbitrary value, say zero, is substituted. Next, a slightly adapted version of the algorithm described so far is applied. This slightly adapted version of the algorithm only differs from the standard algorithm with respect to the way in which the value of objective function (5.28) of a column is determined. In the standard algorithm all entries of a column are taken into account when calculating the value of function (5.28), i.e. also the entries that correspond to variables of which the value is missing. In the adapted version of the algorithm only the entries corresponding to variables with non-missing values are taken into account when calculating the value of function (5.28) of a column. The solution to the error localisation problem is given by the variables corresponding to the optimal generalised correction patterns that are determined plus the variables with missing values. It is clear that the adapted algorithm leads to correct solutions to the error localisation problem.

The advantage of using the adapted algorithm over using the original algorithm is that in the initial stages of the algorithm not much time is spent examining generalised correction patterns according to which many variables with non-missing values should be changed, which may happen in the standard algorithm. The values of function (5.28) of the corresponding columns later become too high when the variables with missing values are also identified as part of the optimal solutions to the error localisation problem.

Fillion and Schiopu-Kratina (1993) give computational results for two applications. The reduction in computing time due to the above treatment of missing data varies between a mere 43% to a factor 13! The applications at Statistics Netherlands show a reduction in computing time that is comparable to the 43% reduction.

5.6. Duffin's algorithm and Fourier-Motzkin elimination

Duffin (1974) considers parametric linear programming problems and uses Fourier-Motzkin elimination to analyse these problems. To make Fourier's method more efficient

Duffin proposes some improvements. Therefore, we will refer to the algorithm described by Duffin as Duffin's algorithm. The algorithm proposed by Fourier and later modified by Duffin was, in contrast to Chernikova's algorithm, not intended for determining the extremal points or extremal rays of a polyhedron. Nevertheless, Duffin's algorithm can be used for generating the extremal points of a polyhedron, as we will show.

In Duffin's paper the processing is done column-wise. Because in the present chapter all processing is done row-wise instead of column-wise, we have reformulated Duffin's algorithm to conform to the notation of the present chapter.

Given is a system of linear inequalities of the following form:

$$(z_1,...,z_{n_r})\mathbf{C} \geq (\lambda_1,...,\lambda_{n_c}), \tag{5.33}$$

where $\mathbf{C}$ is an $n_r \times n_c$-matrix, the z_i are unknown variables, and the λ_j are parameters. We say that column *j* of matrix $\mathbf{C}$ corresponds to the *j*-th entry of the parameter vector.

To examine whether such a system has a solution, we apply Fourier-Motzkin elimination to eliminate the variables. To eliminate variable z_r from (5.33) by means of Fourier-Motzkin elimination we first determine all c_{rj} that are equal to zero, and copy those columns to a new matrix $\mathbf{C}'$. The entry λ'_* of the parameter vector corresponding to such a copied column *h* equals λ_h.

Next, we consider all pairs (*s*,*t*) such that $c_{rs} \times c_{rt} < 0$. For each such pair we choose $\mu_1, \mu_2 > 0$ such that $\mu_1 c_{rs} + \mu_2 c_{rt} = 0$, and add column $\mu_1 c_{*s} + \mu_2 c_{*t}$ to $\mathbf{C}'$. The corresponding entry λ'_* of the parameter vector is given by $\mu_1\lambda_s + \mu_2\lambda_t$. After all pairs (*s*,*t*) such that $c_{rs} \times c_{rt} < 0$ have been considered row *r* of matrix $\mathbf{C}'$ is deleted. We have now obtained a system given by

$$(z_1,...,z_{r-1},z_{r+1},...,z_{n_r})\mathbf{C}' \geq (\lambda'_1,...,\lambda'_k), \tag{5.34}$$

where the λ'_j are certain linear combinations of the parameters λ_j, and *k* equals the number of columns of $\mathbf{C}'$.

Continuing in this fashion we can eliminate all variables from (5.33). In each step the entries of the parameter vector can be expressed as linear combinations of the (original) parameters λ_j. The resulting system is referred to as a *solvent system* and is of the form

$$0 \geq (\lambda_1,...,\lambda_{n_c})\mathbf{B},$$

where **B** is the so-called solvent matrix. The following theorem is easy to show (see e.g. Duffin, 1974).

Theorem 5.3. A system of linear inequalities has a feasible solution if and only if the solvent system is satisfied.

Example 5.2:
Consider the following very simple system of linear inequalities.

$$(z_1, z_2)\begin{pmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{pmatrix} \geq (\lambda_1, \lambda_2, \lambda_3)\,. \tag{5.35}$$

Eliminating z_1 from (5.35) yields

$$z_2(1 \quad -1) \geq (\lambda_1 + \lambda_2, \lambda_3)\,. \tag{5.36}$$

Eliminating z_2 from (5.36) gives us

$$0 \geq \lambda_1 + \lambda_2 + \lambda_3\,. \tag{5.37}$$

So, the solvent matrix is given by $\mathbf{B} = (1 \quad 1 \quad 1)^T$. System (5.35) has a feasible solution if an only if (5.37) is satisfied. ■

If there is at least one pair (*s*,*t*) such that $c_{rs} \times c_{rt} < 0$ when we are eliminating variable z_r, variable z_r is said to be *actively* eliminated. Otherwise, it is said to be *passively* eliminated. Duffin proves the following theorem.

Theorem 5.4. Theorem 5.3 remains valid if after *t* variables have been actively eliminated, any inequality involving *t*+2 or more parametric terms is deleted.

Theorem 5.4 can be used to remove some redundant columns. In general, however, not all redundant columns will be removed when we apply Theorem 5.3. To remove all redundant columns we introduce the concept of dominated columns.

Without loss of generality we suppose that the variables z_1 to z_{n_r} are eliminated in order. When *d* variables have been eliminated from a system of linear inequalities *S*, we obtain an eliminant system *S*(*d*) consisting of linear inequalities involving only z_{d+1} to z_{n_r}. Variable z_{d+1} may already have dropped out of *S*(*d*). In that case, *S*(*d*+1)=*S*(*d*).

In an eliminant system *S*(*d*) a column c_{*s}, corresponding to an entry of the parameter vector given by $\sum_{i=1}^{n_r} \alpha_i \lambda_i$, is *dominated* by another column c_{*t}, corresponding to an entry of the parameter vector given by $\sum_{i=1}^{n_r} \beta_i \lambda_i$, if $\alpha_i = 0$ implies $\beta_i = 0$. Duffin proves the following theorems.

Theorem 5.5. Theorem 5.3 remains valid if redundant columns in an eliminant system $S(d)$ are deleted in the iterative process to determine the solvent matrix.

Theorem 5.6. A column in an eliminant system $S(d)$ is redundant if and only if it is a non-negative combination of the other inequalities. Moreover, a redundant column is dominated.

5.7. Duffin's algorithm and the error localisation problem

5.7.1. Dines' problem

Closely related to Duffin's algorithm is Dines' problem (see Dines, 1927; Duffin, 1974; Houbiers, 1999b). Dines' problem is to find all solutions of the system

$$\mathbf{Cx} = \mathbf{0} \tag{5.38}$$

and

$$\mathbf{x} \geq \mathbf{0}\,, \tag{5.39}$$

where $\mathbf{C}$ is a constant $n_r \times n_c$-matrix and $\mathbf{x}$ an n_c-dimensional vector of unknowns.

The problem can be solved by elimination of equations rather than elimination of variables. Dines' problem can be considered as a dual to Fourier's problem (see e.g. Duffin, 1974, and Williams, 1986).

To solve Dines' problem we can use Fourier's method, and hence Duffin's algorithm (see Duffin, 1974). Namely, the general solution to Dines' problem is given by

$$\mathbf{x}^T = (\mu_1,...,\mu_{n_c})\mathbf{B} \tag{5.40}$$

where $\mu_i \geq 0$ and $\mathbf{B}$ is the solvent matrix of the corresponding parametric linear programming problem given by

$$(z_1,...,z_{n_r})\mathbf{C} \geq (\lambda_1,...,\lambda_{n_c}) \tag{5.41}$$

Solutions to the system given by (5.38) and (5.39) are linear combinations of the columns of the solvent matrix $\mathbf{B}$.

5.7.2. Applying Duffin's algorithm to the error localisation problem

We now describe how Duffin's algorithm can be applied to the error localisation problem. We have already noted before that the error localisation problem can be solved by finding vertices of the polyhedron given by a system of the form (5.14) and (5.15). It remains to be described how the vertices of this polyhedron can be found by Duffin's algorithm. To this

end Houbiers (1999b) introduces additional non-negative variables ϕ and u_1 to u_{n_r}, and rewrites (5.14) as a set of n_r equalities

$$(-\mathbf{A} \quad \mathbf{b} \quad -\mathbf{I}_{\mathbf{n_r}}) \begin{pmatrix} \mathbf{x} \\ \phi \\ \mathbf{u} \end{pmatrix} = \mathbf{0}, \tag{5.42}$$

where $\mathbf{I}_{\mathbf{n_r}}$ is the $n_r \times n_r$-identity matrix. Finding all feasible solutions to the constraints given by (5.42) together with the non-negativity constraints

$$\begin{pmatrix} \mathbf{x} \\ \phi \\ \mathbf{u} \end{pmatrix} \geq \mathbf{0} \tag{5.43}$$

is precisely Dines' problem.

So, the solutions to (5.42) and (5.43) can be found by applying Duffin's algorithm to the system

$$(z_1,...,z_{n_r})(\mathbf{A} \quad -\mathbf{b} \quad -\mathbf{I}_{\mathbf{n_r}}) \geq (\lambda_1,...,\lambda_{n_r+n_c}), \tag{5.44}$$

where the λ_i are again parameters. These solutions are given by the columns of

$$(\mu_1,...,\mu_{n_r+n_c})\mathbf{B}', \tag{5.45}$$

where $\mu_i \geq 0$, and $\mathbf{B}'$ is the solvent matrix of (5.44).

Applying Lemma 5.1 and Theorems 5.3, 5.5 and 5.6, we see that the vertices of the system (5.14) and (5.15) correspond to non-dominated columns of $\mathbf{B}'$ for which the entry corresponding to ϕ, i.e. the n_c-th entry, is larger than zero. The vertex corresponding to such a column *j* is given by

$$v_{i*} = b'_{ij} / b'_{n_c,j} \qquad \text{for } i=1,\ldots,n_v. \tag{5.46}$$

5.7.3. A formulation of Duffin's algorithm for finding vertices

If we apply Duffin's algorithm to find the vertices of the system given by (5.14) and (5.15), the algorithm can be formulated in almost the same way as Chernikova's algorithm (see Section 5.3). In this section we first give a formulation of the former algorithm, and then illustrate it by means of an example.

Like Chernikova's algorithm, Duffin's algorithm for finding the vertices of the system given by (5.14) and (5.15) consists of 9 steps. Steps 1 to 6 and Steps 8 and 9 are exactly the same. Only Step 7 has to be formulated differently.

To explain why only Step 7 has to be formulated differently, we make two remarks. First, the matrix $\mathbf{L^k}$ that is used in Chernikova's algorithm can also be used in Duffin's algorithm. It is then used to keep track of the parameters λ_i. See also Example 5.3 below.

Second, in Duffin's algorithm as described in Section 5.6 the entries of a row that is being processed all have to become zero, whereas in Chernikova's algorithm it is sufficient that the entries in a row all have to become non-negative. However, when Duffin's algorithm is used to find the vertices of the system given by (5.14) and (5.15) a slightly different matrix as in Chernikova's algorithm is used. This matrix includes $-\mathbf{I_{n_r}}$, the $n_r \times n_r$-identity matrix (see Section 5.7.2). We can avoid the use of this $n_r \times n_r$-identity matrix $-\mathbf{I_{n_r}}$, however. Like in Chernikova's algorithm, it is then sufficient that all entries in a row that is being processed have to become non-negative. Again, see also Example 5.3 below.

Step 7 in Duffin's algorithm for finding the vertices of the system (5.14) and (5.15) consists of two parts:

- For each pair (*s*,*t*) for which $y^k_{rs} \times y^k_{rt} < 0$ we choose $\mu_1, \mu_2 > 0$ such that $\mu_1 y^k_{rs} + \mu_2 y^k_{rt} = 0$ and adjoin the column $\mu_1 y^k_{*s} + \mu_2 y^k_{*t}$ to $\mathbf{Y^{k+1}}$.
- Delete (some of) the redundant columns of $\mathbf{Y^{k+1}}$.

Duffin (1974) gives two rules to delete redundant columns of $\mathbf{Y^{k+1}}$. The first rule is weaker than the rule that is used in Step 7 of Chernikova's algorithm. This rule is called the *refined elimination* rule, and is based on Theorem 5.4. It can be formulated as follows:

Refined elimination rule: When *t* variables have been actively eliminated, delete any columns that have been generated by combining *t*+2 or more original columns.

This first rule allows the generation of redundant columns. The second rule makes sure that no redundant columns are generated. We call this rule the *dominance* rule. This rule is based on Theorems 5.5 and 5.6.

Dominance rule: Delete any column y^k_{*u} in $\mathbf{Y^k}$ that is dominated by some other column y^k_{*v}.

Example 5.3:
Suppose we want to use Duffin's algorithm with the refined elimination rule to determine the vertices of the set of constraints of Example 5.1 of Section 5.3, i.e. the constraints given by (5.20). We process the rows in the same order as in Example 5.1.

If we write the system in form (5.42), then we have to apply Duffin's algorithm to the system

$$(z_1,...,z_4)\left(\begin{array}{cccc|cccc} -1 & -1 & 2 & 1 & -1 & 0 & 0 & 0 \\ -5 & 2 & -2 & 3 & 0 & -1 & 0 & 0 \\ -3 & -1 & 4 & 2 & 0 & 0 & -1 & 0 \\ 1 & 1 & -2 & -1 & 0 & 0 & 0 & -1 \end{array}\right) \geq (\lambda_1,...,\lambda_8) \qquad (5.47)$$

according to (5.44).

Like we mentioned above, we prefer to use the matrix $\mathbf{Y^0}$ given by (5.21). The *i*-th column of the lower matrix $\mathbf{L^0}$ then corresponds to λ_i (for *i*=1,…,4). In $\mathbf{Y^0}$ the last four columns of the matrix above do not occur. As a result, we do not have to make sure that all entries in all rows become zero (like Duffin's algorithm requires), but rather that all entries in all rows become non-negative.

When we process row one, we obtain matrix $\mathbf{Y^1}$. Note that a column in matrix $\mathbf{L^1}$ corresponds to a linear combination of the λ_i. Similarly, a column in a matrix $\mathbf{L^k}$ will also correspond to a linear combination of the λ_i.

Processing row four, we obtain matrix $\mathbf{Y^2}$. Subsequently, processing row three, we obtain matrix $\mathbf{Y^3}$. The combination of columns 1 and 4 of matrix $\mathbf{Y^2}$ is deleted because of the refined elimination rule. Likewise, the combination of columns 2 and 3 of matrix $\mathbf{Y^2}$ is also deleted because of the refined elimination rule.

Finally, processing row two, we obtain the matrix below.

$$\hat{\mathbf{Y}}^{\mathbf{4}} = \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 5 & 3 & 0 & 0 & 0 \\ 2 & 1 & 0 & 10 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 2 & 2 & 16 \\ 2 & 1 & 1 & 12 & 4 & 16 \\ 1 & 0 & 0 & 7 & 2 & 6 \\ 0 & 1 & 2 & 0 & 2 & 20 \end{array}\right). \qquad (5.48)$$

In comparison to matrix $\mathbf{Y^4}$ of Example 5.1, this matrix has an additional column, namely the fifth column. If we apply the dominance rule, we see that this column is redundant and may be deleted.

The vertices that Duffin's algorithm determines are, of course, the same as the vertices determined by Chernikova's algorithm. ■

5.8. Comparing Chernikova's algorithm to Duffin's algorithm

If Duffin's algorithm for finding vertices is applied with the dominance rule, the resulting matrix does not contain any redundant columns. One might think that Duffin's algorithm is

better in this respect than Chernikova's algorithm. This is not the case: Chernikova's algorithm does not generate any redundant columns either.

The above statement is easy to prove by induction. First note that the initial matrix $\mathbf{Y^0}$ does not contain any redundant columns. Now, suppose that a matrix $\mathbf{Y^k}$ does not contain any redundant columns, but Chernikova's algorithm generates a redundant column *c* in the next matrix $\mathbf{Y^{k+1}}$. According to Theorem 5.6 there is a column *v* that dominates *c*. There are two possibilities: either *v* was copied from $\mathbf{Y^k}$, or *v* is a combination of two columns of $\mathbf{Y^k}$. In both cases, *v* (if *v* was copied from $\mathbf{Y^k}$) or the constituent columns of *v* (if *v* is a combination of two columns of $\mathbf{Y^k}$) already dominated *c* (if *c* was copied from $\mathbf{Y^k}$) or the constituent columns of *c* (if *c* is a combination of two columns of $\mathbf{Y^k}$). So, both cases would contradict the assumption that $\mathbf{Y^k}$ does not contain any redundant columns.

The problem with Chernikova's algorithm is not that it generates redundant columns, but that it is a rather slow algorithm. This is for a substantial part caused by its Step 7. In Step 7 of Chernikova's algorithm a time-consuming check has to be performed to prevent the generation of redundant columns. Duffin manages to identify a rule, the refined elimination rule, that prevents the generation of a number of redundant columns. This rule is much faster to apply than Step 7 of Chernikova's algorithm, but allows the generation of redundant columns.

All the modifications to Chernikova's algorithm by Rubin, Sande, Schiopu-Kratina and Kovar, and Fillion and Schiopu-Kratina described in this chapter to make it more efficient for solving the error localisation problem also apply to Duffin's algorithm. One could therefore consider using the refined elimination rule during most iterations and only every once in a while, for instance when the number of columns becomes too high to be handled efficiently by a computer, resort to the dominance rule. After all failed rows have been processed the dominance rule has to be applied to remove redundant columns in the solvent matrix $\mathbf{B}'$ of (5.44). One may hope that this leads to an algorithm that is faster than Chernikova's algorithm, but this remains to be tested.

5.9. Modifications for equalities

In Chernikova's algorithm, and equivalently Duffin's algorithm, equalities can be dealt with by specifying each equality as two inequalities. This is, however, rather inefficient, and equalities can be dealt with more efficiently by treating them in a special way. First, a row corresponding to an equality is considered failed if at least one non-zero entry is placed on a generator (see Section 5.5 for a definition of a failed row corresponding to an inequality).

Second, when all entries on a row corresponding to an equality are positive, or all entries on such a row are negative, then the only feasibly solution to (5.12) and (5.13) is $\mathbf{x} = \mathbf{0}$.

Third, when a row representing an equality is processed, only the columns that contain zeroes on that row will be copied in Step 6 of Chernikova's or Duffin's algorithm rather than all columns with non-negative entries on that row.

Fourth, when checking whether two columns should be combined, the rows corresponding to processed equalities need not be considered as they contain only zeroes.

Fifth, if a failed row corresponds to an equality, processing this row may at worst lead to a matrix with $z + pq$ columns, where the failed row has z entries equal to zero, p positive entries and q negative entries. Thus, the criterion given in Section 5.5 to decide which row should be processed next is replaced by the following criterion:

- Calculate for each failed row the value $N_{\max}$ given by $N_{\max} = z + p + pq$ if the row corresponds to an inequality, and by $N_{\max} = z + pq$ if the row corresponds to an equality. Here, z is the number of entries in the row equal to zero, p the number of positive entries in the row and q the number of negative entries in the row.
- Process the failed row with the lowest value of $N_{\max}$. If there are several failed rows for which the value of $N_{\max}$ is minimal, select one of them randomly for processing.

5.10. Discussion

Chernikova's algorithm has been implemented in several software packages for error localisation, namely GEIS (Kovar and Whitridge, 1990), CherryPi (De Waal, 1996; De Waal and Van de Pol, 1997), AGGIES (Todaro, 1999), and a program developed by the Central Statistical Office of Ireland (see Central Statistical Office, 2000). All these systems can handle only numerical data. Sande has developed a computer program for error localisation of mixed data based on Chernikova's algorithm.

In GEIS none of the improvements proposed by Fillion and Schiopu-Kratina have been implemented. The computing power of GEIS is limited to approximately 25 edits involving 25 variables. Larger surveys have to be split up. AGGIES and the program developed by the Central Statistical Office of Ireland have both been developed in SAS. As far as the author of this book is aware the improvements proposed by Fillion and Schiopu-Kratina have been implemented in both systems. Like GEIS the computing power is limited to approximately 25 edits involving 25 variables. Again, larger sets of variables or edits have to be split up into smaller subsets. CherryPi has been developed in Delphi 3. All improvements proposed by Fillion and Schiopu-Kratina have been incorporated into CherryPi. The program can handle a bit more than 100 edits involving a bit more than 100 variables. This is sufficient to handle most business surveys at Statistics Netherlands. For larger surveys the program becomes very slow, i.e. several minutes per record on a 500 MHz PC. For some practical results using CherryPi we refer to Section 11.4. The computer program developed by Sande can handle the same number of edits and variables as CherryPi, or may be even a bit more. Unfortunately, Sande has not published any details on his system.

The method described in this chapter can be improved upon in several ways. First of all, it should be possible to develop a better way to select the row to be processed. In this chapter we only consider one criterion, namely the maximum number of columns that will be generated in the next intermediate matrix. Other criteria, for example a criterion that aims to generate a good feasible solution to the error localisation problem quickly, should be developed and tested.

Second, missing values might be handled better than in the way that is proposed in this chapter. Here we simply fill in an arbitrary value, say zero, for a value that is missing, and then run the vertex generation method to determine the optimal solutions to the error localisation problem. Instead of filling in an arbitrary value, one could consider filling in a more appropriate value. For instance, if the total of some variables is missing, but the values of the constituent variables are not missing, it is probably a good idea to fill in the sum of these values for the missing total. It remains to be examined when appropriate values can be determined quickly, and whether filling these values in really leads to an improved performance of the algorithm.

Third, in this chapter we only consider the vertex generation methods due to Chernikova and Duffin. One could also consider using other vertex generation methods. Sande (2000a) considers a vertex generation algorithm proposed by Bremner, Fukuda and Marzetta (1998) to be a promising one.

At first sight it seems very useful to remove redundant edits that were specified by the subject-matter specialists (see e.g. Houbiers, 1999c). Because the number of rows is reduced if redundant edits are removed, fewer rows have to be processed by the method presented in this chapter, so one could expect this method to terminate sooner. Sande (2000a), however, draws attention to another phenomenon, namely that redundant constraints may help the method in its search for feasible solutions to the error localisation problem. Redundant edits may therefore help the algorithm find a good feasible solution rather quickly. According to Sande (2000a) having redundant edits can often result in shorter computing time.

In Chapter 11 we compare computational results of CherryPi to computational results of various others programs for automatic error localisation, such as a program based on a commercial solver for mixed integer programming problems (see also Section 3.5), and program based on a non-standard branch-and-bound algorithm (see Chapter 8).

We end this chapter with the remark that a modified version of Chernikova's can be used to solve general 0-1 integer programming problems (see Rubin, 1977). In turn, an adapted version of that modification can be used to detect balance edits in a data set of which the edits are as yet unknown (see De Waal, 1997b).

6. The Error Localisation Problem as a Disjunctive-Facet Problem

6.1. Introduction

The disjunctive-facet problem was proposed in 1974 by Glover, Klingman and Stutz. In their words, the motivation for studying this problem is that "it provides a 'direct' formulation for certain classes of mixed integer programs in which one desires to select an optimal production schedule, rental policy, distribution pattern, etc., from a variety of alternatives (disjunctive sets), each of which has an associated set of constraints that must all be satisfied if a particular alternative is chosen". To illustrate the usefulness of the disjunctive-facet problem we briefly describe in Section 6.2 a motivational example given by Glover, Klingman and Stutz. From an abstract point of view, the disjunctive-facet problem is mathematically equivalent to the mixed integer programming problem. The difference is that the disjunctive-facet problem models certain discrete alternatives by means of so-called facet conditions rather than by means of integer variables. In many cases modelling an optimisation problem as a disjunctive-facet problem appears to be more natural than modelling it as a standard mixed integer programming problem. At first sight this seems to be also true for the error localisation problem (see the end of Section 6.2). This was one reason why we became interested in formulating the error localisation problem as a disjunctive-facet problem.

Glover, Klingman and Stutz (1974) also propose a method for solving the disjunctive-facet problem. This method is an extension of an earlier method by Glover and Klingman (1973) for solving the so-called generalised lattice-point problem. The method exploits the structure of the disjunctive-facet problem by means of suitably defined convexity cuts. It is based on repeatedly solving standard linear programming (LP) problems. If the solution to such an LP-problem also constitutes a solution to the disjunctive-facet problem, we are done. Otherwise, we generate a cut and add it to the current LP-problem. The fact that the disjunctive-facet problem can be solved by solving a sequence of standard LP-problems was another reason for our interest in formulating the error localisation problem as a disjunctive-facet problem.

Apart from the article by Glover, Klingman and Stutz, we were unable to find more information on the disjunctive-facet problem in literature. The problem and the solution method by Glover, Klingman and Stutz apparently never became popular. Despite this unpopularity, the two above-mentioned positive aspects of the disjunctive-facet problem, the rather natural formulation for certain classes of optimisation problems and the fact that disjunctive-facet problems can be solved by repeatedly solving LP-problems, offered enough potential in our opinion to study both the problem formulation and the solution method in detail.

In Section 6.3 we describe the disjunctive-facet problem, and in Section 6.4 the algorithm proposed by Glover, Klingman and Stutz for solving it. Section 6.5 illustrates the algorithm by means of an example. The material in these sections is based on the article by Glover, Klingman and Stutz.

In Section 6.6 we show that the algorithm proposed by Glover, Klingman and Stutz is not necessarily finite. This had not yet been noted in literature before. In Section 6.7 we try to formulate the error localisation problem for continuous data as a disjunctive-facet problem. This turns out to be a non-trivial task. In fact, we need to adapt the disjunctive-facet problem from a 'static' problem where all requirements are given beforehand to a 'dynamic' problem where some additional requirements are determined while solving the problem. In Section 6.7 we also modify the algorithm due to Glover, Klingman and Stutz in order to solve the error localisation problem for continuous data. Our algorithm is illustrated in Section 6.8 by means of an example. We prove the finiteness of the proposed algorithm in Section 6.9 for some simple instances. In Section 6.9 we also propose modifications to the algorithm of Section 6.7 so that finiteness is guaranteed in the general case. The material in Sections 6.6 to 6.9 is original work.

Section 6.10 defines some cuts proposed by Cabot (1975) that are in some cases stronger than the cuts used by Glover, Klingman and Stutz. In Section 6.11 we formulate the error localisation problem in a mix of continuous and categorical data as a dynamic disjunctive-facet problem. In the same section we also propose algorithms for solving this problem, and discuss the finiteness of these algorithms. Finally, Section 6.12 concludes this chapter with a brief discussion. Sections 6.11 and 6.12 are again original work.

This chapter is based on De Waal (1997c and 1998a).

6.2. Motivational example

Glover, Klingman and Stutz (1974) give the following motivational example for the disjunctive-facet problem. A group of investors has 20 pounds of a raw material from which they can make either of two finished products. For the first product three pounds of the raw material per pound of the finished product is required, and for product 2 five pounds of the raw material per pound of the finished product. A buyer has offered to buy all of each product the investors can produce in one week at 100 Euro per pound of product 1 and 150 Euro per pound of product 2. The products can be produced at two shops (A and B). The investors can buy 17 man-hours of work from shop A for 450 Euro or 17 man-hours from shop B for 350 Euro. To produce a pound of product 1 shop A takes four man-hours and shop B 6 man-hours. To produce a pound of product 2 takes shop A 7 man-hours and shop B 4 man-hours. The decisions the investors need to make is whether to use shop A or shop B and how much of each product to produce in order to maximise their profit.

Let x_i (i=1,2) be the pounds of product i produced. Let x_3 be 1 if shop A is used and 0 if it is not, and let x_4 be 1 if shop B is used and 0 if it is not. The objective function that is to be maximised is then given by

$$100x_1 + 150x_2 - 450x_3 - 350x_4 . \tag{6.1}$$

The objective function (6.1) has to be maximised subject to

$$3x_1 + 5x_2 \leq 20 , \tag{6.2}$$

$$x_3 + x_4 \geq 1 , \tag{6.3}$$

$$x_3 \leq 1, \tag{6.4}$$

$$x_4 \leq 1, \tag{6.5}$$

$$\mathbf{x} \geq \mathbf{0} \tag{6.6}$$

and

$$\mathbf{x} \in L_A \text{ or } \mathbf{x} \in L_B, \tag{6.7}$$

where

$$L_A = \{\mathbf{x} \mid 4x_1 + 7x_2 \leq 17, x_3 = 1\} \tag{6.8}$$

and

$$L_B = \{\mathbf{x} \mid 6x_1 + 4x_2 \leq 17, x_4 = 1\}. \tag{6.9}$$

Constraint (6.2) is a resource constraint. Constraint (6.3) expresses that at least one shop is used. Constraint (6.7) says that either the constraint set that corresponds to shop A or the constraint set that corresponds to shop B applies. By introducing slack variables u_1, u_2, u_3, and u_4 given by

$$u_1 = 17 - 6x_1 - 4x_2, \tag{6.10}$$

$$u_2 = 17 - 4x_1 - 7x_2, \tag{6.11}$$

$$u_3 = 1 - x_4, \tag{6.12}$$

$$u_4 = 1 - x_3, \tag{6.13}$$

L_A and L_B can be re-written as

$$L_A = \{\mathbf{x} \mid u_4 = 0, u_2 \geq 0\} \tag{6.14}$$

and

$$L_B = \{\mathbf{x} \mid u_3 = 0, u_1 \geq 0\}. \tag{6.15}$$

The requirement that $\mathbf{x}$ be an element of L_A or L_B is an example of a so-called L-set requirement (see next section). Each u_i occurring in the definition of an L_k must satisfy either $u_i = 0$ or $u_i \geq 0$. We will therefore refer to these conditions as facet conditions (see next section).

In the error localisation problem, one may hope that the structure of edits we consider in this book, i.e. edits of type (3.1), allows one to formulate this problem in a natural manner as a disjunctive-facet problem. After all, the numerical conditions of systems of edits of type (3.1) can be split up into disjunctive subsystems that each have to hold true if the corresponding categorical variables values have certain values. At first sight, it appears to be possible to let each such disjunctive subsystem correspond with an L-set requirement.

Unfortunately, as we will see in the course of this chapter, formulating the error localisation problem as a disjunctive-facet problem is a non-trivial exercise.

6.3. The disjunctive-facet problem

The disjunctive-facet problem is an LP problem with an extra requirement: the *L-set requirement*. In the disjunctive-facet problem two polyhedral regions are defined: one polyhedral region corresponding to the LP problem, the other one corresponding to the *L*-set requirement. Loosely speaking, sets L_k are defined that identify (or contain) a number of facets of the convex polyhedral region corresponding to the *L*-set requirement. A set L_k is said to be *satisfied* by a vector **x** when **x** is an element of L_k. Now the *L*-set requirement is that at least *q* sets L_k are satisfied. Below we give a stricter mathematical formulation of the disjunctive-facet problem. This formulation closely follows the formulation given by Glover, Klingman and Stutz.

In the disjunctive-facet problem a set of constraints

$$\mathbf{u} = \mathbf{d} - \mathbf{D}\mathbf{x}, \tag{6.16}$$

$$u_i \geq 0 \quad \text{for } i \geq r, \tag{6.17}$$

(for certain *r*) is defined. Each component u_i of **u** is *involved* in one or more sets L_k ($k = 1,...,s$). For a given set L_k, each u_i that is involved in L_k has a *facet condition* associated with it. This facet condition is denoted as $u_i \in C_{ik}$. There are two types of facet conditions: C_{ik} may be either $\{0\}$ or the set containing all non-negative real numbers. That is, a facet condition for a u_i that is involved in L_k may either be $u_i = 0$ or $u_i \geq 0$. A u_i may have different facet conditions in different sets L_k, but can have only one facet condition in a given set (if u_i is involved in that set), because the combination of the two facet conditions $u_i = 0$ and $u_i \geq 0$ is equivalent to $u_i = 0$. A set L_k is said to be *satisfied* if $u_i \in C_{ik}$ for all u_i that are involved in L_k. For convenience we sometimes write $i \in L_k$ instead of "u_i is involved in L_k".

Using the above terminology, the disjunctive-facet problem may be defined as finding a solution to the problem of minimising

$$x_0 = \mathbf{c}\mathbf{x}, \tag{6.18}$$

i.e. the inner product of a constant vector **c** and a vector of unknowns **x**, subject to the constraints (6.16) and (6.17),

$$\mathbf{v} = \mathbf{b} - \mathbf{A}\mathbf{x}, \tag{6.19}$$

$$\mathbf{v} \geq \mathbf{0} \tag{6.20}$$

and the *L*-set requirement, i.e. the constraint that at least *q* of the sets L_k (k=1,...,s) are satisfied. The restrictions (6.19) and (6.20) include $\mathbf{x} \geq \mathbf{0}$.

The Error Localisation Problem as a Disjunctive-Facet Problem

To illustrate the concepts introduced above we repeat an example of a disjunctive-facet problem given by Glover, Klingman and Stutz (1974) for the case q = 1, i.e. for the case where only one L-set has to be satisfied.

Example 6.1:
The objective function, which is to be minimised, is defined as

$$x_0 = -\tfrac{1}{7}x_1 - x_2 . \tag{6.21}$$

The v-variables, i.e. the slacks corresponding to the ordinary LP constraints, have to satisfy

$$v_1 = -2 + \tfrac{2}{3}x_1 + x_2 , \tag{6.22}$$

$$v_2 = 6 + \tfrac{1}{7}x_1 - x_2 , \tag{6.23}$$

$$v_3 = 12 - x_1 - x_2 , \tag{6.24}$$

$$v_4 = x_1 , \tag{6.25}$$

$$v_5 = x_2 \tag{6.26}$$

and

$$v_i \geq 0 \quad i=1,\ldots,5. \tag{6.27}$$

The u-variables, i.e. the variables that are involved in the facet conditions, have to satisfy

$$u_1 = -3 - \tfrac{1}{6}x_1 + x_2 , \tag{6.28}$$

$$u_2 = 4 + x_1 + x_2 , \tag{6.29}$$

$$u_3 = -4 + \tfrac{1}{3}x_1 + x_2 , \tag{6.30}$$

$$u_4 = 6 - \tfrac{3}{2}x_1 + x_2 , \tag{6.31}$$

$$u_5 = 4 + \tfrac{3}{4}x_1 - x_2 , \tag{6.32}$$

$$u_6 = -\tfrac{2}{7}x_1 + x_2 \tag{6.33}$$

and

$$u_i \geq 0 \quad \text{for } i \geq 4 . \tag{6.34}$$

The L-sets L_1, L_2 and L_3 are defined as follows: $L_1 = \{u_1 \geq 0, u_3 \geq 0, u_6 = 0\}$, $L_2 = \{u_2 \geq 0, u_5 = 0\}$ and $L_3 = \{u_1 \geq 0, u_2 \geq 0, u_4 = 0\}$. Each variable u_i is involved in at least one set L_k, namely u_1 is involved in L_1 and L_3, u_2 in L_2 and L_3, u_3 in L_1, u_4 in L_3, u_5 in L_2, and u_6 in L_1. A set L_k is satisfied when the vector $\begin{pmatrix}\mathbf{u}\\ \mathbf{v}\end{pmatrix}$ is an element of L_k. At least one set L_k (k = 1,...,3) should be satisfied.

6.4. The algorithm of Glover, Klingman and Stutz

Glover, Klingman and Stutz (1974) distinguish between two cases: $q = 1$ and $q > 1$. The algorithm for the second case being a simple extension of the algorithm for the case $q = 1$. In the present book we only discuss the algorithm for the case $q = 1$.

The general idea of the algorithm as follows. A set of constraints, cuts, is defined while executing the algorithm. These cuts are added to the set of constraints defined by (6.16), (6.17), (6.19) and (6.20). Initially, the set of cuts is empty. We repeat the following two steps until a solution that satisfies the *L*-set requirement has been found.

1. Use the simplex method (see e.g. Hadley, 1962, and Chvátal, 1983) to solve the LP problem of minimising objective function (6.18) subject to the constraints defined by (6.16), (6.17), (6.19) and (6.20), where (6.19) consists of the original constraints and the current set of cuts.

2. Add an additional cut to the set of cuts when the solution obtained does not satisfy the *L*-set requirement.

It is clear that the performance of the algorithm depends on the cuts that are generated during the algorithm. The cuts that are generated have to be *valid*, i.e. they have to be violated by the current solution obtained by the simplex method in Step 1 but have to be satisfied by an optimal solution (if a feasible solution exists) to the disjunctive-facet problem. This guarantees that when an optimal solution to an LP problem is found during the algorithm that satisfies the *L*-set requirement, this solution to the LP problem is also an optimal solution to the disjunctive-facet problem. The cuts preferably should be as deep as possible, i.e. they should cut off a region of the feasible region of the current LP problem that is as large as possible. When the cuts are deep, the algorithm generally terminates after relatively few iterations.

We now describe the cuts that are proposed by Glover, Klingman and Stutz. Their cuts are constructed by considering the simplex tableau after an iteration of the simplex method. First the vector $\mathbf{y}$ is defined by

$$\mathbf{y} = \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix}. \tag{6.35}$$

We construct the simplex tableau for the following LP problem: maximise

$$x_0 = -\mathbf{cx} \tag{6.36}$$

subject to the constraints

$$\mathbf{y} = \overline{\mathbf{B}}_{\mathbf{0}} - \overline{\mathbf{B}}\mathbf{x}\,, \tag{6.37}$$

and

$$\mathbf{v} \geq 0 \text{ and } y_i \geq 0 \text{ (for } i \geq r), \tag{6.38}$$

where, initially,

$$\overline{\mathbf{B}}_{\mathbf{0}} = \begin{pmatrix} \mathbf{d} \\ \mathbf{b} \end{pmatrix} \text{ and } \overline{\mathbf{B}} = \begin{pmatrix} \mathbf{D} \\ \mathbf{A} \end{pmatrix}. \tag{6.39}$$

The above format of the simplex tableau enables us to keep track of the transformations of the *u*-constraints due to pivoting. We keep in mind, however, that it is not the above LP problem that needs to be solved. Besides satisfying (6.37) and (6.38) at least one of the sets L_k has to be satisfied.

Using the above simplex tableau as the initial tableau, the general current tableau representation of the problem may be given as: maximise

$$x_0 = c_0 - \mathbf{ct} \tag{6.40}$$

subject to the constraints

$$\mathbf{y} = \mathbf{b_0} - \mathbf{Bt} \tag{6.41}$$

and (6.38), where **y** is the same as above, **t** denotes any set of current non-basic variables, and c_0, **c**, $\mathbf{b_0}$ and **B** depend on the composition of **t**. After the termination of the simplex algorithm the current optimal solution is denoted as

$$\mathbf{y}' = \begin{pmatrix} \mathbf{u}' \\ \mathbf{v}' \end{pmatrix} = \mathbf{b_0}, \tag{6.42}$$

and the *j*-th column of **B** as $\mathbf{b_j}$.

In order to describe the cuts of Glover, Klingman and Stutz conveniently we need to introduce some notation for the distinct values of a non-basic variable occurring in (6.40) and (6.41), say t_j, corresponding to the sequence in which the u_i-variables become zero as t_j is increased from zero. The sequence determines values t_j^h, h=1,...,p (where p depends on j) such that $0 = t_j^0 < t_j^1 < t_j^2 < \ldots < t_j^p = \infty$, where at least one u_i that was not equal to 0 for $t_j = 0$ becomes 0 for $t_j = t_j^h$, $0 < h < p$ (we define $p = 1$ if no such u_i exists), and where no u_i is 0 unless $t_j = t_j^h$ for $0 < h < p$.

We also define I_j^h by $I_j^h = \{i \mid u_i = 0 \text{ for } t_j = t_j^h\}$, i.e. I_j^h is the set of indices of the u_i that become 0 when $t_j = t_j^h$, holding the remaining non-basic variables at 0.

To initiate their algorithm Glover, Klingman and Stutz define a "current" set L_k^* for each L_k that consists of the indices of the u_i involved in L_k such that $u_i' \notin C_{ik}$.

Having introduced all this notation we are finally ready to state the algorithm of Glover, Klingman and Stutz:

1. Find the optimal solution to the current LP problem by means of the simplex algorithm.

2. Select a non-basic variable t_j (not previously selected) and examine the values t_j^h in sequence, beginning with $h = 1$. If $t_j^1 = \infty$, then let $t_j^* = \infty$ and select another t_j.

3. If $L_k^* \subset I_j^h$ for any k such that t_j is not involved in L_k, then let $t_j^* = t_j^h$ and return to Step 2 to select another t_j variable.

4. Otherwise, for each k such that t_j is not involved in L_k, update L_k^* to be $L_k^* - I_j^h$, and increase h by 1. If now $t_j^h = \infty$, set $t_j^* = \infty$ and return to Step 2. Otherwise, return to Step 3.

5. When all t_j have been selected: construct the cut

$$v_* \geq 0, \tag{6.43}$$

where

$$v_* = \sum_j (1/t_j^*) t_j - 1, \tag{6.44}$$

and add it to the set of constraints. Return to Step 1.

Note that the number of entries of the vectors **v** and **y** is increased by one each time Step 5 is executed.

Glover, Klingman and Stutz (1974) give an algebraic proof that the cuts generated in the above way are indeed valid ones. We restrict ourselves to giving geometric arguments for the validity of the cuts. First note that L_k^* consists of the indices of those u_i involved in L_k for which $u_i' \notin C_{ik}$ and such that the *associated hyperplanes*, i.e. $u_i = 0$, do not intersect any of the line segments

$$\mathbf{b_0} - \mathbf{b_j}\lambda \quad \text{for } 0 < \lambda < t_j^* \tag{6.45}$$

corresponding to the t_j that have already been selected, nor the line segment

$$\mathbf{b_0} - \mathbf{b_j}\mu \quad \text{for } 0 < \mu < t_j^h \tag{6.46}$$

corresponding to the t_j that is currently being dealt with, where t_j^h is the value that is currently being examined in Step 3. Namely, any i in the initial current set L_k^*, i.e. the index of any u_i involved in L_k for which $u_i' \notin C_{ik}$, for which the associated hyperplane, $u_i = 0$, intersects either one of the line segments (6.45) or the line segment (6.46) is removed from L_k^*.

When a current set L_k^* is contained in a set I_j^h in Step 3, then all hyperplanes associated to the u_i involved in L_k intersect at least one of the line segments (6.45), or the line segment

$$\mathbf{b_0} - \mathbf{b_j}\mu \quad \text{for } 0 < \mu \leq t_j^h \tag{6.47}$$

corresponding to the t_j that is currently being dealt with, where t_j^h is the value that is currently being examined. At least one hyperplane associated to a u_i involved in L_k intersects the line segment (6.47) in the point

$$\mathbf{b_0} - \mathbf{b_j}t_j^h . \tag{6.48}$$

After all t_j have been dealt with, for each t_j a value t_j^* has been determined. The cutting plane that is constructed in Step 5 is the plane through those points given by

$$\mathbf{b_0} - \mathbf{b_j}t_j^* \quad \text{for which } t_j^* < \infty \tag{6.49}$$

and parallel to those vectors $\mathbf{b_j}$ for which $t_j^* = \infty$.

If t_j is involved in L_k then L_k^* does not have to be updated in Step 4. Namely, suppose that *i* is an element of the I_j^h for the t_j involved in L_k ($j \in J$) only, then no point in the region that is cut off can satisfy $u_i = 0$ and $t_j = 0$ for all $j \in J$.

It is clear that the cut generated in Step 5 cuts off the optimal solution to the current LP problem. Now suppose that a cut generated by the algorithm of Glover, Klingman and Stutz would not be valid, i.e. suppose that the new feasible region would not contain the optimal solution to the disjunctive-facet problem (assuming that such an optimal solution exists). In other words, suppose that an optimal solution to the disjunctive-facet problem is contained in the region that is cut off. Denote this optimal solution by

$$\mathbf{y}'' = \begin{pmatrix} \mathbf{u}'' \\ \mathbf{v}'' \end{pmatrix}. \tag{6.50}$$

This optimal solution satisfies the *L*-set requirement, say it satisfies at least L_k. The set L_k can be written as

$$L_k = \{\mathbf{y} \mid u_i = 0 \text{ for } i \in J_1, u_j \geq 0 \text{ for } j \in J_2\} \tag{6.51}$$

for certain index sets J_1 and J_2 ($J_1 \cap J_2 = \varnothing$). The initial current set L_k^* can be expressed in terms of J_1 and J_2 as

$$L_k^* = \{i \in J_1 \mid u_i' \neq 0\} \cup \{i \in J_2 \mid u_i' < 0\} . \tag{6.52}$$

Now suppose i is an element of the above initial current set L_k^*. We consider two cases.

1. If $i \in J_1$, then $u_i' \neq 0$ in the optimal solution to the current LP problem and $u_i'' = 0$ in an optimal solution to the disjunctive-facet problem, which we assume to lie in the region that is cut off. This implies that the hyperplane $u_i = 0$ intersects at least one of the line segments given by (6.45). This latter observation is clear from a geometric point of view. Glover (1973) calls this the *first cut-search lemma*, and proves it algebraically.

2. If $i \in J_2$, then $u_i' < 0$ in the optimal solution to the current LP problem and $u_i'' \geq 0$ in an optimal solution to the disjunctive-facet problem, which we assume to lie in the region that is cut off. This implies that there is a point in the region that is cut off for which $u_i = 0$. This in turn implies again that the hyperplane $u_i = 0$ intersects at least one of the line segments given by (6.45).

We conclude that if the region that is cut off contained an optimal solution to the disjunctive-facet problem (assuming that such an optimal solution exists) then all hyperplanes $u_i = 0$ for i in the initial current set L_k^*, where L_k is any L-set that is satisfied by this optimal solution to the disjunctive-facet problem, would intersect a line segment of the form (6.45). This is a contradiction, because by construction of the algorithm there exists at least one i in the initial current set L_k^* such that the hyperplane $u_i = 0$ intersects the half line

$$\mathbf{b_0} - \mathbf{b_j}\lambda \quad (\lambda > 0) \tag{6.53}$$

in a point of the form (6.49).

Above we have given geometric arguments for the validity of the cuts generated by the algorithm of Glover, Klingman and Stutz. In principle, it is possible that the cuts can be made somewhat deeper. The algorithm guarantees only that an optimal solution to the disjunctive-facet problem does not lie in the region that is cut off, but it does not guarantee that an optimal solution lies on the boundary of the cut constructed in Step 5. In Section 6.10 alternative cuts are described that are sometimes stronger than the cuts proposed by Glover, Klingman and Stutz.

6.5. An example of the algorithm of Glover, Klingman and Stutz

In this section we apply the algorithm of Glover, Klingman and Stutz to the example of Section 6.3. We only construct one cut. We use the dual simplex algorithm to find the optimal LP solution to the problem of minimising (6.21) subject to (6.22) to (6.34). The optimal LP solution is attained at the point for which both $v_1 = 0$ and $u_6 = 0$. This point is infeasible for the disjunctive-facet problem. The simplex tableau (in "extended Tucker form") is given in Table 6.1.

Table 6.1: Simplex tableau

		$-v_1$	$-u_6$
x_0	-9/10	9/20	11/20
u_1	-11/4	-1/8	-7/8
u_2	-13/10	-27/20	7/20
u_3	-27/10	-13/20	-7/20
u_4	69/20	51/40	-91/40
u_5	199/40	-39/80	119/80
u_6	0	0	-1
v_1	0	-1	0
v_2	57/10	3/20	17/20
v_3	93/10	27/20	-7/20
v_4	21/10	-21/20	21/20
v_5	3/5	-6/20	-7/10

According to the current simplex tableau given in Table 6.1 we can express the basic u-variables in the following way in terms of the non-basic variables:

$$u_1 = -11/4 + 1/8\, v_1 + 7/8\, u_6 , \tag{6.54}$$

$$u_2 = -13/10 + 27/20\, v_1 - 7/20\, u_6 , \tag{6.55}$$

$$u_3 = -27/10 + 13/20\, v_1 + 7/20\, u_6 , \tag{6.56}$$

$$u_4 = 69/20 - 51/40\, v_1 + 91/40\, u_6 , \tag{6.57}$$

and

$$u_5 = 199/40 + 39/80\, v_1 - 119/80\, u_6 . \tag{6.58}$$

Increasing v_1 we find:

$$t_1^1 = 26/27 = 0.96 \text{ and } I_1^1 = \{2\} \text{ (i.e. } u_2 = 0 \text{ for } v_1 = 26/27 \text{ and } u_6 = 0),$$

$$t_1^2 = 46/17 = 2.71 \text{ and } I_1^2 = \{4\} ,$$

$$t_1^3 = {}^{54}\!/_{13} = 4.15 \text{ and } I_1^3 = \{3\}$$

and

$$t_1^4 = 22 \text{ and } I_1^4 = \{1\}.$$

Similarly, increasing u_6 we find:

$$t_2^1 = {}^{22}\!/_{7} = 3.14 \text{ and } I_2^1 = \{1\},$$

$$t_2^2 = {}^{398}\!/_{119} = 3.34 \text{ and } I_2^2 = \{5\}$$

and

$$t_2^3 = {}^{54}\!/_{7} = 7.71 \text{ and } I_2^3 = \{3\}.$$

The initial current sets L_k^* (k=1,...,3) are given by $L_1^* = \{1,3\}$, $L_2^* = \{2, 5\}$ and $L_3^* = \{1, 2, 4\}$.

We first consider v_1. Since $L_k^* \not\subset I_1^1$ ($k = 1,...,3$), the current sets L_k^* are updated to become: $L_1^* = \{1, 3\}$, $L_2^* = \{5\}$ and $L_3^* = \{1, 4\}$. Since $L_k^* \not\subset I_1^2$ ($k = 1,...,3$), the current sets L_k^* are again updated to become: $L_1^* = \{1, 3\}$, $L_2^* = \{5\}$ and $L_3^* = \{1\}$. Since $L_k^* \not\subset I_1^3$ (k=1,...,3), the current sets L_k^* are once again updated to become: $L_1^* = \{1\}$, $L_2^* = \{5\}$ and $L_3^* = \{1\}$. Since $L_k^* \subset I_1^4$ for k=1, 3, we set $t_1^* = t_1^4 = 22$.

Next we consider u_6. Since $L_k^* \subset I_2^1$ for k=1,…,3, we set $t_2^* = t_2^1 = {}^{22}\!/_{7}$.

We construct the cut

$$ {}^{1}\!/_{22} v_1 + {}^{7}\!/_{22} u_6 \geq 1, \tag{6.59}$$

i.e. the cut

$$v_6 = -1 + {}^{1}\!/_{22} v_1 + {}^{7}\!/_{22} u_6 \geq 0. \tag{6.60}$$

We add the constraint

$$v_6 \geq 0 \tag{6.61}$$

to the set of constraints. We solve the new LP problem, i.e. the problem of minimising (6.21) subject to (6.22) to (6.34), (6.60) and (6.61). If necessary, we construct a new cut, etc, until the disjunctive-facet problem has been solved. In fact, it turns out that in this example in total two cuts are necessary (see Glover, Klingman and Stutz, 1974). After these two cuts have been added to the set of constraints the disjunctive-facet problem can solved to optimality by solving the associated LP problem.

6.6. Finiteness of the algorithm of Glover, Klingman and Stutz

Glover, Klingman and Stutz (1974) do not discuss the finiteness of their algorithm. In this section we demonstrate that their algorithm for the disjunctive-facet problem does not always terminate after a finite number of iterations. The counterexample we consider is given below. This example is taken from De Waal (1997c).

The objective function to be minimised is given by

$$x_0 = 0 , \tag{6.62}$$

i.e. we are in fact only trying to find a feasible vector.

The v-variables are given by

$$v_1 = x_1 , \tag{6.63}$$

$$v_2 = x_2 , \tag{6.64}$$

$$v_3 = -\sqrt{3}x_1 - x_2 + 2\sqrt{3} , \tag{6.65}$$

and

$$v_4 = \sqrt{3}x_1 - x_2 - \sqrt{3} . \tag{6.66}$$

These v-variables have to satisfy

$$v_i \geq 0 \quad \text{for } i=1,\ldots,4. \tag{6.67}$$

Note that the constraint $v_1 \geq 0$ is redundant, i.e. it is satisfied if the other three constraints are satisfied.

The u-variables are given by

$$u_1 = x_1 - \tfrac{3}{2} , \tag{6.68}$$

$$u_2 = \sqrt{3}x_1 - 3x_2 - \sqrt{3} , \tag{6.69}$$

and

$$u_3 = \sqrt{3}x_1 + 3x_2 - 2\sqrt{3} . \tag{6.70}$$

The L-sets L_1, L_2 and L_3 are defined by

$$L_1 = \{u_2 = 0, u_3 = 0\} , \tag{6.71}$$

$$L_2 = \{u_1 = 0, u_3 = 0\} \tag{6.72}$$

and

$$L_3 = \{u_1 = 0, u_2 = 0\} . \tag{6.73}$$

At least one of the sets L_k (k=1,...3) has to be satisfied.

The disjunctive-facet problem has only one feasible solution, namely $x_1 = \frac{3}{2}$ and $x_2 = \frac{1}{6}\sqrt{3}$. In that case all sets L_k (k=1,...3) happen to be satisfied.

In (x_1, x_2)-space, the polyhedron defined by (6.63) to (6.70) has three vertices: vertex A_0 given by $x_1 = 1$ and $x_2 = 0$ (corresponding to $v_2 = 0$ and $v_4 = 0$), vertex B_0 given by $x_1 = 2$ and $x_2 = 0$ (corresponding to $v_2 = 0$ and $v_3 = 0$), and vertex C_0 given by $x_1 = \frac{3}{2}$ and $x_2 = \frac{1}{2}\sqrt{3}$ (corresponding to $v_3 = 0$ and $v_4 = 0$).

Suppose that during the first iteration vertex A_0 is found as solution to the current LP problem. The cut generated by the algorithm of Glover, Klingman and Stutz is given by

$$v_5 = \sqrt{3}x_1 + x_2 - \tfrac{3}{2}\sqrt{3} \geq 0. \qquad (6.74)$$

Vertex A_0 is cut off by this cut, whereas two new vertices are generated in (x_1, x_2)-space (a vertex given by $x_1 = \frac{3}{2}$ and $x_2 = 0$, and a vertex given by $x_1 = \frac{5}{4}$ and $x_2 = \frac{1}{4}\sqrt{3}$). Name one of these new vertices A_1.

Suppose that during the second iteration vertex B_0 is found as solution to the current LP problem. The cut generated by the algorithm of Glover, Klingman and Stutz is given by

$$v_6 = -\sqrt{3}x_1 + x_2 + \tfrac{3}{2}\sqrt{3} \geq 0. \qquad (6.75)$$

Vertex B_0 is cut off by this cut. There are two vertices of the new polyhedron for which $v_6 = 0$, namely a vertex given by $x_1 = \frac{3}{2}$ and $x_2 = 0$, and a vertex given by $x_1 = \frac{7}{4}$ and $x_2 = \frac{1}{4}\sqrt{3}$. At least one of the vertices for which $v_6 = 0$ differs from vertex A_1. Name one of these vertices B_1.

Suppose that during the third iteration vertex C_0 is found as solution to the current LP problem. The cut generated by the algorithm of Glover, Klingman and Stutz is given by

$$v_7 = -x_2 + \tfrac{1}{4}\sqrt{3} \geq 0. \qquad (6.76)$$

Vertex C_0 is cut off by this cut. There are two vertices of the new polyhedron for which $v_7 = 0$, namely a vertex given by $x_1 = \frac{5}{4}$ and $x_2 = \frac{1}{4}\sqrt{3}$, and a vertex given by $x_1 = \frac{7}{4}$ and $x_2 = \frac{1}{4}\sqrt{3}$. Either one of these vertices has been named A_1, or the other one has been named B_1. Name the vertex that has not received a name yet C_1.

In (x_1, x_2)-space, the new feasible region, consisting of convex combinations of A_1, B_1 and C_1, is in some sense similar to the original feasible region, consisting of convex combinations of A_0, B_0 and C_0. The new feasible region can be constructed from the original feasible region by applying the transformation

$$\varphi : (x_1, x_2) \mapsto (\tfrac{9}{4} - \tfrac{1}{2}x_1, \tfrac{1}{4}\sqrt{3} - \tfrac{1}{2}x_2) \tag{6.77}$$

to each point (x_1, x_2) in the original feasible region in (x_1, x_2)-space. The transformation φ maps the point $(\tfrac{3}{2}, \tfrac{1}{6}\sqrt{3})$ onto itself. Another property of φ is that it leaves $u_1 = 0$, $u_2 = 0$ and $u_3 = 0$ invariant. That is, φ maps points on $u_1 = 0$ to points on $u_1 = 0$, points on $u_2 = 0$ to points on $u_2 = 0$, and points on $u_3 = 0$ to points on $u_3 = 0$. The area of the new feasible region is a factor four smaller than the area of the original feasible region.

In the same manner we can construct a sequence of vertices A_i, B_i and C_i ($i \geq 1$) such that the area of the region given by convex combinations of A_i, B_i and C_i is a factor four smaller than the area of the region given by convex combinations of A_{i-1}, B_{i-1} and C_{i-1}. The transformation φ transforms each region given by convex combinations of A_{i-1}, B_{i-1} and C_{i-1} into the region given by convex combinations of A_i, B_i and C_i. Each of the regions given by convex combinations of A_i, B_i and C_i contains the only feasible point to the disjunctive-facet problem in its interior. We can conclude that irrespective of the number of cuts generated in the above way we will never obtain the optimal solution to the disjunctive-facet problem. This concludes our counterexample to the finiteness of the algorithm of Glover, Klingman and Stutz.

6.7. Error localisation for continuous data as a dynamic disjunctive-facet problem

The error localisation problem described in Chapter 3 can be considered as a *dynamic* disjunctive-facet problem. That is, the *L*-sets in the *L*-set requirement are not fixed beforehand, but are generated during the algorithm. After a, generally non-optimal, solution to the error localisation problem has been found the *L*-sets are updated. The new *L*-sets express the wish to find a solution to the error localisation problem that is better than the previous best one, i.e. they express the wish to find a solution to the error localisation problem for which the value of the objective function is smaller than the smallest value found so far. This process goes on until an *L*-set requirement cannot be satisfied anymore. The best solution to the error localisation problem that has been found is then an optimal one.

Whenever we refer to the error localisation problem in the present section, or in Section 6.8, 6.9, or 6.10, we in fact mean the error localisation problem for continuous data. Below we first reformulate the error localisation problem for continuous data in terms of *u* and *v* variables.

Edit checks for numerical data are written as a set of linear inequalities that have to hold simultaneously, i.e. they are written as

$$\mathbf{Az} \leq \mathbf{b}\,, \tag{6.78}$$

where $\mathbf{A}$ is a constant matrix and $\mathbf{b}$ is a constant vector. When a record $\mathbf{z^0}$ fails the edit checks, we want to find a modification vector $\Delta\mathbf{z}$ such that $\mathbf{z^0} + \Delta\mathbf{z}$ satisfies the edit

checks. We introduce two non-negative vectors $\mathbf{z}^{\mathbf{P}}$ and $\mathbf{z}^{\mathbf{N}}$. These vectors satisfy the following relations

$$\Delta\mathbf{z} = \mathbf{z}^{\mathbf{P}} - \mathbf{z}^{\mathbf{N}} \ . \tag{6.79}$$

We also introduce a vector **v** defined by

$$\mathbf{v} = \mathbf{b}' - \mathbf{A}'\mathbf{x} \,, \tag{6.80}$$

where

$$\mathbf{b}' = \mathbf{b} - \mathbf{A}\mathbf{z}^{\mathbf{0}} \,, \tag{6.81}$$

$$\mathbf{A}' = (\mathbf{A} \mid \mathbf{-A}) \tag{6.82}$$

and

$$\mathbf{x} = \begin{pmatrix} \mathbf{z}^{\mathbf{P}} \\ \mathbf{z}^{\mathbf{N}} \end{pmatrix}. \tag{6.83}$$

We also introduce *u*-variables given by

$$u_i = x_i \ \text{ for } i=1,\ldots,2n. \tag{6.84}$$

In terms of the *u*- and *v*-variables the error localisation problem for continuous data can be written as: minimise the objective function

$$\sum_{i=1}^{2n} w_i \delta(u_i) \tag{6.85}$$

subject to

$$\mathbf{v} \geq \mathbf{0} \,, \tag{6.86}$$

where n is the number of continuous variables involved in the error localisation problem, and

$$w_{n+i} = w_i \ \text{ for } i=1,\ldots,n. \tag{6.87}$$

Suppose that at least one, generally non-optimal, solution **x** to the error localisation problem has already been determined. For each of the already determined solutions we evaluate the value of the objective function (6.85). We let an L-set correspond to all solutions to the error localisation problem better than the best one found so far. If the smallest value of the objective function found so far equals T, then these L-sets are given by

$$\sum_{i \notin L_k} w_i < T \,, \tag{6.88}$$

i.e. by

$$L_k = \{x_i = 0 \mid i \in J_k\} \,, \tag{6.89}$$

where J_k is any set of indices such that

$$\sum_{i \notin J_k} w_i < T \ . \tag{6.90}$$

There may be extremely many of such *L*-sets. In fact, there may be too many *L*-sets to apply the algorithm of Glover, Klingman and Stutz efficiently to the error localisation problem. Therefore, the *L*-sets are not specified explicitly, but instead are incorporated implicitly in our algorithm.

Define I^0 to be $\{i \mid t_j = x_i \text{ for some } j\}$. Define the initial current set I_i^* (for $i \in I^0$) to be the empty set. Also define the initial current set Q^* to be the empty set. Finally, define the initial value of T to be ∞. We are now ready to formulate our first algorithm, which we call Algorithm 6.1.

1. Find a feasible solution to the current LP problem. The algorithm terminates if no feasible solution to the current LP problem exists. In that case the best solution to the error localisation problem that has been found so far is the optimal one.

2. Evaluate the value V of the objective function (6.85) for the solution found in Step 1. If $V < T$, then update T to be V.

3. Select a non-basic variable t_j (not previously selected). If $t_j = x_i$ (for some *i*), then $J^* = I^0 - \{i\}$, else $J^* = I^0$. Examine the values t_j^h in sequence, beginning with $h = 1$. If $t_j^1 = \infty$, then let $t_j^* = \infty$ and select another t_j.

4. Determine the minimum *N* of $\sum_{i \notin \Omega} w_i$, where $\Omega = \left(\bigcup_{i \in J^*} R_i \right) \cup Q^* \cup I_j^h$ and R_i is either $\{i\}$ or I_i^* (for $i \in J^*$). If $N < T$ then let $t_j^* = t_j^h$ and return to Step 3 to select another t_j variable.

5. If $N \geq T$ and $t_j \neq x_i$ for all *i* then update Q^* to be $Q^* \cup I_j^h$. If $N \geq T$ and $t_j = x_i$ for some *i* then update I_i^* to be $I_i^* \cup I_j^h$. Increase *h* by 1. If now $t_j^h = \infty$, set $t_j^* = \infty$ and return to Step 3. Otherwise, return to Step 4.

6. When all t_j have been selected: check whether one (or more) of the vectors $\mathbf{b_0} - \mathbf{b_j} t_j^*$ ($t_j^* < \infty$) for which the value of the objective function is smaller than *T* (if such a vector exists), is a feasible solution to the current LP problem. If none of the vectors

$\mathbf{b_0} - \mathbf{b_j} t_j^*$ satisfies these conditions, then go to Step 7. Otherwise, select the best of these vectors, i.e. the vector for which the value of the objective function (6.85) is the smallest, and pivot to this feasible solution. Return to Step 2.

7. Construct the following cut

$$v_* \geq 0\,, \tag{6.91}$$

where

$$v_* = \sum_j (1/t_j^*) t_j - 1 \tag{6.92}$$

and add it to the set of constraints. Return to Step 1. ■

Theorem 6.1: The cuts generated by Algorithm 6.1 (without Step 6) are the same as the cuts generated by the algorithm of Glover, Klingman and Stutz for the case where the *L*-sets are all sets L_k such that $\sum_{i \notin L_k} w_i < T$ (provided the *t*-variables are selected in the same order in both algorithms).

Proof. Suppose that $L_k^* \subset I_j^h$ (for any *k* such that t_j is not involved in L_k) in Step 3 of the algorithm of Glover, Klingman and Stutz. In that case

$$\{i \mid x_i \text{ is involved in } L_k\} \subset \left(\bigcup_{i \in J^*} R_i^0 \right) \cup Q^* \cup I_j^h\,,$$

for certain sets R_i^0 for which either $R_i^0 = \{i\}$ or $R_i^0 = I_i^*$. This implies that $N < T$ in Step 4 of Algorithm 6.1, because *N* is the minimum of all sets $\Omega = \left(\bigcup_{i \in J^*} R_i \right) \cup Q^* \cup I_j^h$ where R_i is either $\{i\}$ or I_i^*. Note that because t_j is not involved in L_k, we can take $J^* = I^0 - \{m\}$ if $t_j = x_m$ for some *m*. If $t_j \neq x_m$ for all *m* we take $J^* = I^0$.

On the other hand, suppose that $\sum_{i \notin \Omega_0} w_i < T$ in Step 4 of Algorithm 6.1, where $\Omega_0 = \left(\bigcup_{i \in J^*} R_i^0 \right) \cup Q^* \cup I_j^h$ then define $L_k = \{x_i = 0 \text{ for } i \in \Omega_0\}$. For this set L_k we have $L_k^* \subset I_j^h$ and $\sum_{i \notin L_k} w_i = \sum_{i \notin \Omega_0} w_i < T$. Finally, note that t_j is not involved in L_k, because if $t_j = x_m$ (for some *m*) then $m \notin \Omega_0$. ■

Unfortunately, Algorithm 6.1 seems inefficient in practice, because in Step 4 a time consuming optimisation problem must be solved. We can avoid solving this optimisation problem if we construct cuts that are less deep than the cuts of Glover, Klingman and Stutz. Denote the sum of all reliability weights by *W*. Initially, we set $w_i' = w_i$ (for all *i*). We replace Step 4 of Algorithm 6.1 by

4. If $W - \sum_{i \in \Gamma} w_i' < T$, where $\Gamma = J^* \cup Q^* \cup I_j^h$ then let $t_j^* = t_j^h$. If in addition $t_j = x_i$ then update w_i' to be the maximum of w_i and $\sum_{i \in I_j^*} w_i$. Return to Step 3 to select another t_j variable.

This algorithm is called Algorithm 6.2.

Theorem 6.2: The cuts generated by Algorithm 6.2 are valid.

Proof. The term $W - \sum_{i \in \Gamma} w_i'$, where $\Gamma = J^* \cup Q^* \cup I_j^h$, which is evaluated in Step 4 of Algorithm 6.2 is a lower bound on *N*, which has to be determined in Step 4 of Algorithm 6.1. So, the cuts generated by Algorithm 6.2 are at most as deep as the cuts generated by Algorithm 6.1. Because the cuts generated by Algorithm 6.1 are valid, the cuts generated by algorithm 6.2 are valid as well. ∎

Theorem 6.3: If the current LP solution is the best solution found so far to the error localisation problem for continuous data then the cuts made by Algorithms 6.2 and 6.1 (and hence by the algorithm of Glover, Klingman and Stutz) are the same (provided the *t*-variables are selected in the same order).

Proof. If the current LP solution is the best solution found so far to the error localisation problem then for $t_j = x_i$ we have $w_i > \sum_{k \in I_j^*} w_k$. So, the minimum *N* of $\sum_{i \notin \Omega} w_i$ determined in Step 4 of Algorithm 6.1 equals the lower bound on *N*, $W - \sum_{i \in \Gamma} w_i'$, determined in Step 4 of Algorithm 6.2. ∎

We conclude this section with some remarks regarding the algorithms. First of all note that in the formulation of the error localisation problem as a dynamic disjunctive-facet problem we do not minimise a linear objective function. This implies that in each iteration we only have to find a feasible solution to an LP problem in Step 1.

Step 6 is included with the aim of accelerating the algorithms. Any vector $\mathbf{b_0} - \mathbf{b_j} t_j^*$ ($t_j^* < \infty$) that is a feasible solution to the error localisation problem is a vertex neighbouring the solution to the current LP problem. If one of the vectors $\mathbf{b_0} - \mathbf{b_j} t_j^*$ ($t_j^* < \infty$) is a feasible solution to the error localisation problem and is better than any previously found solution, then we pivot to that vector. Not only do we find a better solution to the error localisation problem, but also we update the value T. The subsequent cuts will be deeper than if we would not have pivoted to that solution. However, Step 6 requires some computing time. Further research is necessary to determine whether execution of Step 6 really reduces the computing time.

The first step when applying Algorithm 6.1 or Algorithm 6.2 is the determination of a, generally non-optimal, solution $\mathbf{z^0} + \Delta\mathbf{z}$ to the error localisation problem. In principle, any solution to the error localisation problem suffices. In practice, however, it is a good idea to start with a relatively good solution, i.e. a solution with only few non-zero entries. If the modification vector $\Delta\mathbf{z}$ has many non-zero entries then the original data vector $\mathbf{z^0}$ apparently contains many errors. Consequently, the data vector $\mathbf{z^0}$ and its modified version $\mathbf{z^0} + \Delta\mathbf{z}$ are of little statistical value, and may as well be discarded. One can therefore specify an upper bound, say $T_{\max}$, on the number of non-zero entries of the modification vector $\Delta\mathbf{z}$. When the number of non-zero entries of $\Delta\mathbf{z}$ exceeds $T_{\max}$ then $\mathbf{z^0}$ is discarded for automatic error localisation. The use of an upper bound $T_{\max}$ for the number of non-zero entries of $\Delta\mathbf{z}$ generally leads faster to the optimal solution of the error localisation problem than when such an upper bound would not be used. So, our first step to solve the error localisation problem optimally is to find a solution to the error localisation problem with at least n-$T_{\max}$ entries equal to zero, where n denotes the number of variables involved in the error localisation problem. This first step can be solved by applying, for instance, Algorithm 6.2, where the objective function is given by the number of non-zero entries and where initially the value of T equals $T_{\max}$.

6.8. An example

In this we apply Algorithm 6.2 proposed in Section 6.7 to generate a cut for a simple example. In this example we assume that the edits are given by

$$T = P + C, \tag{6.93}$$

$$0.5T \leq C \tag{6.94}$$

and

$$C \leq 1.1T. \tag{6.95}$$

where T denotes the turnover of an enterprise, P its profit and C its costs. T and C may only assume non-negative values, whereas P may assume any value (if P is negative the enterprise actually made a loss). Edit (6.93) expresses that the turnover of an enterprise equals the sum of the profit and the costs. This is an edit check that can be logically derived, and holds true for every enterprise. Edits (6.94) and (6.95) give bounds for the costs of an enterprise in terms of its turnover. These edits cannot be logically derived, but

have to be determined by a subject matter specialist by means of data analysis. It will hold true for certain classes of enterprises only.

We assume that the original values of the variables in a certain faulty record are given by $T = 100$, $C = 60{,}000$, and $P = 40{,}000$. The reliability weights are assumed to be $w_T = 3/2$, $w_C = 1$ and $w_p = 1$. The optimal solution to the error localisation is changing the value of T to be 100,000. The optimum value of the objective function is $3/2$.

For each variable we introduce a positive change, denoted by subscript 1, and a negative change, denoted by subscript 2. We replace each variable by its original value plus the positive change minus the negative change. For instance, we replace T by $100 + T_1 - T_2$. The problem of adapting the values of T, C and P in such a way that the sum of reliability weights is minimised while the new values of T, C and P satisfy (6.93), (6.94) and (6.95) can then be written as

$$\text{minimise } \sum_{i=1}^{6} w_i \delta(u_i), \tag{6.96}$$

where

$$w_1 = w_4 = w_T, w_2 = w_5 = w_C \text{ and } w_3 = w_6 = w_P, \tag{6.97}$$

subject to the *v*-constraints

$$v_1 = T_1 - T_2 - C_1 + C_2 - P_1 + P_2 - 99{,}900 \geq 0, \tag{6.98}$$

$$v_2 = -T_1 + T_2 + C_1 - C_2 + P_1 - P_2 + 99{,}900 \geq 0, \tag{6.99}$$

$$v_3 = -1/2 T_1 + 1/2 T_2 + C_1 - C_2 + 59{,}950 \geq 0, \tag{6.100}$$

$$v_4 = 1.1T_1 - 1.1T_2 - C_1 + C_2 - 59{,}890 \geq 0, \tag{6.101}$$

$$v_5 = T_1 - T_2 + 100 \geq 0, \tag{6.102}$$

$$v_6 = C_1 - C_2 + 60{,}000 \geq 0, \tag{6.103}$$

$$v_7 = T_1 \geq 0, \tag{6.104}$$

$$v_8 = C_1 \geq 0, \tag{6.105}$$

$$v_9 = P_1 \geq 0, \tag{6.106}$$

$$v_{10} = T_2 \geq 0, \tag{6.107}$$

$$v_{11} = C_2 \geq 0, \tag{6.108}$$

$$v_{12} = P_2 \geq 0, \tag{6.109}$$

and the *u*-constraints

$$u_1 = T_1\,, \tag{6.110}$$

$$u_2 = C_1\,, \tag{6.111}$$

$$u_3 = P_1\,, \tag{6.112}$$

$$u_4 = T_2\,, \tag{6.113}$$

$$u_5 = C_2 \tag{6.114}$$

and

$$u_6 = P_2\,. \tag{6.115}$$

Constraints (6.98) and (6.99) correspond to (6.93), constraint (6.100) to (6.94), and constraint(6.101) to (6.95). Constraints (6.102) and (6.103) reflect that the turnover and the costs of an enterprise, respectively, are non-negative.

In the first step of the algorithm we have to determine a vector satisfying (6.98) to (6.109). Suppose the feasible solution $v_1 = v_2 = v_3 = v_7 = v_8 = v_9 = v_{10} = 0$, $v_4 = 60$, $v_5 = 100$, $v_6 = 50$, $v_{11} = 59{,}950$ and $v_{12} = 39{,}950$ is determined by the simplex algorithm. The value of the objective function (6.85) equals 2 for this solution.

The solution is degenerate. Suppose that variables v_2, v_3, v_7, v_8, v_9 and v_{10} are the non-basic variables. We can express the *u*-variables in terms of the non-basic variables:

$$u_1 = v_7\,, \tag{6.116}$$

$$u_2 = v_8\,, \tag{6.117}$$

$$u_3 = v_9\,, \tag{6.118}$$

$$u_4 = v_{10}\,, \tag{6.119}$$

$$u_5 = 59{,}950 - v_3 - \tfrac{1}{2} v_7 + v_8 + \tfrac{1}{2} v_{10} \tag{6.120}$$

and

$$u_6 = 39{,}950 - v_2 + v_3 - \tfrac{1}{2} v_7 + v_9 + \tfrac{1}{2} v_{10}\,. \tag{6.121}$$

The total sum of reliability weights W equals 7. The weights w_i' are initially given by $w_i' = w_i$ (i=1,...,6). The set I^0 is given by

$$I^0 = \{1, 2, 3, 4\}\,, \tag{6.122}$$

i.e. by the indices of the *u*-variables with value zero in the current solution to the LP problem.

We select a non-basic variable, say v_2. We have $t_1^1 = 39{,}950$ and $I_1^1 = \{6\}$. So, $\Gamma = \{1,2,3,4,6\}$. Because $W - (w_1' + w_2' + w_3' + w_4' + w_6') < 2$, we let $t_1^* = 39{,}950$.

We select another non-basic variable, say v_3. We have $t_2^1 = 59{,}950$ and $I_2^1 = \{5\}$. So, $\Gamma = \{1,2,3,4,5\}$. Because $W - (w_1' + w_2' + w_3' + w_4' + w_5') < 2$, we let $t_2^* = 59{,}950$.

We select yet another non-basic variable, say v_7. In this case $J^* = \{2, 3, 4\}$, because $u_1 = v_7$. We have $t_3^1 = 79{,}900$ and $I_3^1 = \{6\}$. So, $\Gamma = \{2,3,4,6\}$. Because $W - (w_2' + w_3' + w_4' + w_6') \geq 2$, we update I_1^* to be $\{6\}$. We consider $t_3^2 = 119{,}900$ and $I_3^2 = \{5\}$. Because $W - (w_2' + w_3' + w_4' + w_5' + w_6') < 2$, we let $t_3^* = 119{,}900$.

We select another non-basic variable, say v_8. We find that $t_4^* = \infty$. Similarly, we find for the remaining non-basic variables that the corresponding t^*-values equal ∞.

We obtain the following cut:

$$v_2/39{,}950 + v_3/59{,}950 + v_7/119{,}900 \geq 1. \tag{6.123}$$

Note that the solution to the current LP problem violates this constraint, whereas the optimal solution to the error localisation problem (for which $v_2 = 0$, $v_3 = 10{,}000$ and $v_7 = 99{,}900$) satisfies this constraint. This demonstrates that the generated cut is a valid one.

6.9. Finiteness of the algorithms

It is not clear whether the algorithms presented in Section 6.7 are finite. We have neither been able yet to demonstrate finiteness for the general case, nor to find a counterexample to finiteness. In contrast to the algorithm of Glover, Klingman and Stutz for the normal disjunctive-facet problem (see Section 6.6) the algorithms for the error localisation problem presented in Section 6.7 are finite for the two-dimensional case. This is caused by the special structure of the *u*-constraints.

Theorem 6.4: For the two-dimensional case the algorithms for the error localisation problem for continuous data presented in Section 6.7 terminate after a finite number of iterations.

Proof. In this proof we assume that the vector **x** given by (6.83) is two-dimensional. We divide the set of *v*-constraints into three classes: *edge defining* ones, *vertex defining* ones and *redundant* ones. For an edge defining *v*-constraint $v_i \geq 0$ we have that $v_i = 0$ on an entire edge of the feasible polyhedron. For a vertex defining *v*-constraint $v_i \geq 0$ we have that $v_i = 0$ in a vertex of the feasible polyhedron only. For a redundant *v*-constraint $v_i \geq 0$ we have that v_i is larger than zero for all points in the feasible polyhedron. Note that according to the above definition two *v*-constraints that are equal to zero on the same edge of the feasible polyhedron are both considered edge defining. A subset of the vertex defining *v*-constraints is formed by the *inner-vertex defining* *v*-constraints. An inner-vertex

defining *v*-constraint $v_i \geq 0$ is a vertex defining *v*-constraint for which $v_i = 0$ in a vertex where both $x_1 > 0$ and $x_2 > 0$. Denote the number of edge defining *v*-constraints by *E*, the number of inner-vertex defining *v*-constraints by *I* and the number of vertices with $x_1 = 0$ or $x_2 = 0$ (or both) by *Z*. By the construction of the cuts new vertices for which $x_1 = 0$ or $x_2 = 0$ are never created during the algorithm (either Algorithm 6.1 or Algorithm 6.2 of Section 6.7). So, *Z* is non-increasing during the algorithm.

Suppose v_i and v_j are the non-basic variables that are used in Steps 3 to 7 of the algorithm to generate the cut. Both these non-basic variables are either edge defining or vertex defining. When v_i and v_j intersect in a point for which either $x_1 = 0$ or $x_2 = 0$ then *Z* is decreased by one. The sum *E+I* may have been increased by one (the cut, an edge defining *v*-constraint, has been added to the set of constraints).

Suppose that v_i and v_j do not intersect in a point for which $x_1 = 0$ or $x_2 = 0$. We consider the generated cut. There are three possibilities. First, the cut may be given by "$0 \geq 1$", i.e. both $t_i^* = \infty$ and $t_j^* = \infty$. In this case the algorithm terminates immediately, because the new set of constraints is infeasible.

Second, suppose that the cut is given by

$$v_i / t_i^* + v_j / t_j^* \geq 1, \tag{6.124}$$

where $0 < t_i^*, t_j^* < \infty$. Then the only possible feasible point for which $v_i = 0$ is the point for which $v_j = t_j^*$ (if v_j became larger than t_j^* then either x_1 or x_2 would become smaller than zero). For this point we therefore have $x_1 = 0$ or $x_2 = 0$. This implies that $v_i \geq 0$ is neither an edge defining *v*-constraint nor an inner-vertex defining *v*-constraint. Similarly, we can prove that $v_j \geq 0$ is neither an edge defining *v*-constraint nor an inner-vertex defining *v*-constraint.

Third, suppose only one variable is involved in the cut, i.e. that, without loss of generality, the cut is given by

$$v_j / t_j^* \geq 1, \tag{6.125}$$

where $0 < t_j^* < \infty$. In this case it is clear that $v_j \geq 0$ has become redundant. We examine whether $v_i \geq 0$ can be edge defining or inner-vertex defining. For $v_j = t_j^*$ and $v_i = 0$ either $x_1 = 0$ or $x_2 = 0$ (or both). Without loss of generality we assume that $x_1 = 0$ for $v_j = t_j^*$ and $v_i = 0$. This implies that x_1 can be written as

$$x_1 = \alpha + \beta v_i - (\alpha / t_j^*) v_j, \tag{6.126}$$

where $\alpha \geq 0$ ($\alpha \geq 0$ because $v_i = v_j = 0$ defines a feasible point). So, $v_i = 0$ implies that x_1 and v_j are related by

$$x_1 = \alpha - (\alpha / t_j^*) v_j . \tag{6.127}$$

Combining (6.125) and (6.127), we find $x_1 \leq 0$ if $v_i = 0$. Because we also have to satisfy the constraint $x_1 \geq 0$, we have derived that $x_1 = 0$ (and $v_j = t_j^*$) if $v_i = 0$. That is, we have derived that $v_i \geq 0$ is neither an inner-vertex defining *v*-constraint nor an edge-defining *v*-constraint. So, after adding a cut involving only one *v*-variable both *v*-constraints $v_i \geq 0$ and $v_j \geq 0$ become neither edge defining nor inner-vertex defining.

Summarising, when a cut is added to the set of *v*-constraints the following may happen:

Either the trivial cut "$0 \geq 1$" is added, in which case the algorithm terminates immediately, or

1. An edge defining *v*-constraint, namely the cut, is added.
2. Some of the edge defining or inner-vertex defining *v*-constraints, different from $v_i \geq 0$ and $v_j \geq 0$, may become neither edge defining nor inner-vertex defining *v*-constraints.
3. Some of the edge defining *v*-constraints, different from $v_i \geq 0$ and $v_j \geq 0$, may become inner-vertex defining *v*-constraints.
4. If $v_i \geq 0$ and $v_j \geq 0$ are used to generate the cut, and $v_1 = 0$ and $v_2 = 0$ intersect in a point for which $x_1 = 0$ or $x_2 = 0$, then *Z* is decreased by at least one.
5. If $v_i \geq 0$ and $v_j \geq 0$ are used to generated the cut, and $v_1 = 0$ and $v_2 = 0$ do not intersect in a point for which $x_1 = 0$ or $x_2 = 0$, then both *v*-constraints become neither edge defining nor inner-vertex defining.

Whenever a non-trivial cut, i.e. a cut different from "$0 \geq 1$", is added to the set of constraints the value of $G = E + I + 2Z$ is decreased by at least one. Because initially the value of *G* is finite and the algorithm terminates when *G* becomes zero, the algorithm (either Algorithm 6.1 or Algorithm 6.2) is finite. ■

The two-dimensional case is not a very interesting one. In fact, this case arises when only one variable occurs in the error localisation problem (because to each variable in the error localisation problem two variables in the dynamic disjunctive-facet problem correspond; see (6.83)). The above theorem does show, however, that the infiniteness of the algorithm proposed by Glover, Klingman and Stutz does not imply that our algorithms for the error localisation problem are infinite. For higher-dimensional cases we do not know whether our algorithms are finite. However, we can prove finiteness when we modify the algorithms a bit.

In Step 1 of the algorithms of Section 6.7 no linear objective function is minimised, only a feasible solution to the current LP problem is determined. Although the simplex algorithm requires more iterations to determine the optimum of a linear function than to determine a feasible solution only, it may still be worthwhile minimising a linear function in Step 1. Not only can we prove finiteness when we minimise a particular linear function in Step 1, but this may possibly also accelerate the algorithm because better solutions to the error localisation problem (resulting in an update of the best value T of the objective function) may be found faster. We now describe the proposed modification of the algorithms.

Initially, let $\Phi_S = \varnothing$ and $\Psi_C = \varnothing$. Φ_S and Ψ_C are sets of sets. During the first iteration Step 1 of the modified algorithms is the same as Step 1 of the algorithms of Section 6.7. During later iterations Step 1 is replaced by:

1. Select a set Ω_0 from the set of sets Ψ_C. Minimise the linear objective function

$$\sum_{i \in \Omega_0} x_i \tag{6.128}$$

subject to the constraints (*v*-constraints and *u*-constraints) of the current LP problem. Replace Φ_S by $\Phi_S \cup \{\Omega_0\}$, and let $\Psi_C = \varnothing$.

Step 4 of Algorithm 6.1 is replaced by:

4. Determine the minimum N of $\sum_{i \notin \Omega} w_i$, where $\Omega = \left(\bigcup_{i \in J^*} R_i \right) \cup Q^* \cup I_j^h$ and R_i is either $\{i\}$ or I_i^* (for $i \in J^*$), and $\Omega \notin \Phi_S$. If $N < T$ then let $t_j^* = t_j^h$, and replace Ψ_C by $\Psi_C \cup \{\Omega\}$. Return to Step 3 to select another t_j variable.

Step 4 of Algorithm 6.2 is replaced by:

4. If $W - \sum_{i \in \Gamma} w_i' < T$, where $\Gamma = J^* \cup Q^* \cup I_j^h$ then add a set given by $\Omega = \left(\bigcup_{i \in J^*} R_i \right) \cup Q^* \cup I_j^h$, where R_i is either $\{i\}$ or I_i^* (for $i \in J^*$), such that $\Omega \notin \Phi_S$ to Ψ_C.

a) If such a set Ω exists then let $t_j^* = t_j^h$. If in addition $t_j = x_i$ (for some *i*) then update w_i' to be the maximum of w_i and $\sum_{i \in I_j^*} w_i$. Return to Step 3 to select another t_j variable.

b) If such a set Ω does not exist then go to Step 5.

Theorem 6.5: With the above modifications the algorithms for solving the error localisation problem for continuous data are finite.

Proof. First note that if $\Psi_C = \varnothing$ after all t_j have been dealt with then the best solution to the error localisation problem determined so far cannot be improved anymore. The algorithm terminates in this case. Therefore, we assume that $\Psi_C \neq \varnothing$ after all t_j have been dealt with. Suppose that a set $\Omega_0 \in \Psi_C$ is used to specify the linear function in Step 1. There are two possibilities:

1. If the optimal value of (6.128) is larger than zero, then there is no feasible solution to the error localisation problem for which $x_i = 0$ for all $i \in \Omega_0$.
2. If the optimal value of (6.128) equals zero, then the value of objective function (6.85) is at most $V_0 = \sum_{i \notin \Omega_0} w_i$. In later iterations we want to find a solution to the error localisation problem for which the value of the objective function (6.85) is less than $T \leq V_0$. This implies that in later iterations the set Ω_0 does not have to be used in Step 1 to specify the linear function.

So, a set of indices Ω_0 has to be used at most once to specify a linear function in Step 1. The set of sets Φ_S consists of those sets that have already been used to specify the linear function in Step 1. During each iteration the number of sets Ω_0 contained in Φ_S is increased by one. Because there are only finitely many different sets Ω_0 the algorithm terminates after a finite number of iterations. ∎

In the modified version of Algorithm 6.1 the set of sets Ψ_C keeps track of those sets that may correspond to a better solution to the error localisation problem than the best one previously found. If the optimal value of (6.128) equals zero in Step 1, then we have found a better solution to the error localisation problem.

In the modified version of Algorithm 6.2 the value $\sum_{i \notin \Omega} w_i$ corresponding to the set Ω added to Ψ_C in the modified Step 4 should preferably be as small as possible. If this value is less than *T* and the optimal value of (6.128) equals zero in Step 1, then we have found a better solution to the error localisation problem.

6.10. Cabot-cuts

In this section we describe another kind of cuts than those presented in Sections 6.4 to 6.9. These cuts have been proposed by Cabot (1975) for the generalised lattice-point problem. In some cases these cuts are stronger than the cuts proposed by Glover, Klingman and Stutz. Cabot (1975) proposed his cuts for a problem with fixed constraints. We describe how such cuts can be generated for the error localisation problem, where the constraints are determined dynamically.

We first observe that an optimal solution to the error localisation problem is attained in a vertex of the feasible polyhedron defined by (6.86) (see Section 5.2 for a simple proof). This observation implies the existence of a number ε such that if x_i is non-zero in an optimal solution to the error localisation problem then automatically $x_i \geq \varepsilon$. We assume that the feasible region is bounded, i.e. that $x_i \leq M$ (for all i) for a sufficiently large number M. This assumption is justified for the error localisation problem, because all variables refer to observable quantities, such as turnover, costs and profit of an enterprise. Observable quantities are bounded.

Suppose the best solution to the error localisation problem found so far is given by $x_i = 0$ for $i \in I$, otherwise $x_i \neq 0$, where I is a certain index set. The value of the objective function equals $\sum_{i \notin I} w_i = H_0$. We want to find a better solution to the error localisation problem. That is, we want to find any solution to the error localisation problem such that $x_j = 0$ if and only if $j \in J$ (otherwise $x_j \neq 0$) and $\sum_{j \notin J} w_j = H_1 < H_0$.

We consider the convex function

$$F(x_1,...,x_n) = \sum_i \frac{w_i}{p + x_i q}, \tag{6.129}$$

where p and q are non-negative constants.

In the current best solution to the error localisation problem the value of (6.129) is given by

$$\sum_{i \in I} \frac{w_i}{p} + \sum_{i \notin I} \frac{w_i}{p + x_i q}. \tag{6.130}$$

An upper bound on this value is given by

$$\sum_{i \in I} \frac{w_i}{p} + \sum_{i \notin I} \frac{w_i}{p + \varepsilon q} = \frac{(W - H_0)}{p} + \frac{H_0}{p + \varepsilon q}, \tag{6.131}$$

where W is the sum of all reliability weights.

The value of F in a better solution to the error localisation problem is given by

$$\sum_{j \in J} \frac{w_j}{p} + \sum_{j \notin J} \frac{w_j}{p + x_j q} . \tag{6.132}$$

A lower bound on this value is given by

$$\sum_{j \in J} \frac{w_j}{p} + \sum_{j \notin J} \frac{w_j}{p + Mq} = \frac{(W - H_1)}{p} + \frac{H_1}{p + Mq} = F_L . \tag{6.133}$$

Now, we want the value of F in the current best solution to the error localisation problem to be smaller than the value of F in any better solution to the error localisation problem. This is the case if

$$\frac{(W - H_1)}{p} + \frac{H_1}{p + Mq} > \frac{(W - H_0)}{p} + \frac{H_0}{p + \varepsilon q} . \tag{6.134}$$

This leads to

$$p < \frac{\varepsilon M q (H_0 - H_1)}{M H_1 - \varepsilon H_0} , \tag{6.135}$$

if M is sufficiently large so that $MH_1 > \varepsilon H_0$.

The value of F is smaller in the current best solution to the error localisation problem than in any better solution if p is chosen according to (6.135) and H_1 is chosen equal to the largest possible sum of w_j smaller than H_0. M is chosen in such a way that $x_i \leq M$ (for all i) for each feasible solution to the error localisation problem, and $M > \varepsilon H_0 / H_1$. The value of q can be chosen equal to an arbitrary positive number.

Given a non-basic variable v_i we may increase its value from zero (while keeping the values of the other non-basic variables equal to zero) until the value of (6.129) equals (6.133). This gives us the value t_i^*. After all t_i^* have been determined in this way, we generate the valid cut

$$\sum_i (1 / t_j^*) t_i \geq 1 . \tag{6.136}$$

This cut is added to the set of constraints. A feasible solution to the updated set of constraints is subsequently determined, and a new cut is constructed, etc. This process goes on until a set of constraints does not have a feasible solution. The best solution to the error localisation problem obtained so far is then the optimal one.

Example 6.2:

We use the simple example of Section 6.8 again to illustrate the cuts that are generated by the algorithm. We assume that the original values of the variables in a faulty record are given by $T = 100$, $C = 60{,}000$, and $P = 40{,}000$, like we also assumed in Section 6.8. The error localisation problem is then given by (6.96) to (6.115). Again we suppose that in the

first step of the algorithm the feasible solution given by $v_1 = v_2 = v_3 = v_7 = v_8 = v_9 = v_{10} = 0$, $v_4 = 60$, $v_5 = 100$, $v_6 = 50$, $v_{11} = 59{,}950$ and $v_{12} = 39{,}950$ is determined by the simplex algorithm. The value of the objective function (6.85) equals 2 for this solution. The solution is degenerate. Suppose that variables v_2, v_3, v_7, v_8, v_9 and v_{10} are the non-basic variables. We can express the *u*-variables in terms of the non-basic variables by (6.116) to (6.121).

We suppose that in this case M = 1,000,000 is sufficiently large to be an upper bound on the values of the variables T_i, C_i and P_i (i=1,2). Usually the numbers that have to be filled in by respondents as answers to questions posed in a questionnaire are rounded figures, i.e. they are integer. We assume this to be the case in the present situation too. We can therefore choose ε to be equal to 1. We also choose q to be equal to 1. The value of the objective function (6.96) of the best solution to the error localisation problem found so far is given by $H_0 = 2$. The largest sum of reliability weights smaller than H_0 is given by $H_1 = 3/2$. Relation (6.135) gives $p < 0.333$. We choose $p = 0.3$.

We have $W = 7$ and $F_L = 18.333$. We write the convex function (6.129) in terms of the non-basic variables.

$$F = \frac{3/2}{0.3+v_7} + \frac{1}{0.3+v_8} + \frac{1}{0.3+v_9} + \frac{3/2}{0.3+v_{10}} + \frac{1}{59950.3 - v_3 - 1/2\, v_7 + v_8 + 1/2\, v_{10}} + \frac{1}{39950.3 - v_2 + v_3 - 1/2\, v_7 + v_9 + 1/2\, v_{10}} \tag{6.137}$$

To find the value t_1^* corresponding to the first non-basic variable, say v_2, we set all other non-basic variables equal to zero in (6.137). We find the following function

$$F_{v_2}(v_2) = 16.666 + \frac{1}{39950.3 - v_2}. \tag{6.138}$$

The value t_1^* is now found by solving

$$F_{v_2}(t_1^*) = F_L, \tag{6.139}$$

i.e. by solving

$$16.666 + \frac{1}{39950.3 - t_1^*} = 18.333. \tag{6.140}$$

We find $t_1^* = 39{,}949.7$. The other values t_i^* can be found in a similar way. After all values t_i^* have been determined the cut (6.136) is added to the set of constraints. This concludes our example. ■

The Cabot-cuts described above can be used in combination with the cuts described in Section 6.7. If the Cabot-cuts are stronger than the cuts described in Section 6.7, the Cabot-cuts are added to the system of constraints, otherwise the cuts described in Section 6.7 are added to the system of constraints. The advantage of this approach is that the cuts that are added to the system of constraints sometimes are stronger than the cuts described in Section 6.7. The disadvantage is that generating cuts in this way can be very time-consuming. First, because in many cases the cuts generated by the algorithms in Section 6.7 are stronger than the Cabot-cuts. For instance, in the example the value t_1^* determined for v_2 is less when the above approach is used than when the value t_1^* is determined by means of Algorithm 6.2 of Section 6.7 (see also the example in Section 6.8). Second, it may be difficult to determine the largest possible sum of w_j smaller than H_0 efficiently. In the above example it is easy to see that H_1 is equal to $3/2$, but in general this is not so easy. However, when all reliability weights are equal, say equal to one, then the determination of the largest possible sum of reliability weights smaller than H_0 is trivial: it is given by $H_0 - 1$. Third, the computation of t_i^* may require relatively much computing time. Generally, to compute a t_i^* a non-linear equation in one variable must be solved. In the above example the calculation of t_i^* for any non-basic variable different from v_2 is a non-trivial matter.

6.11. Error localisation for mixed data as a dynamic disjunctive-facet problem

In this section we formulate the error localisation problem for mixed data, i.e. for a mix of categorical and continuous data, as a dynamic disjunctive-facet problem. In particular, we formulate the mixed integer programming (MIP) formulation for the error localisation problem given in Section 3.3 as a dynamic disjunctive-facet problem. The algorithms of Section 6.7 cannot be applied directly to solve this error localisation problem for mixed data. We therefore consider a relaxation of the error localisation problem for mixed data. This relaxation of the error localisation problem for mixed data is the problem of minimising a certain objective function, which is described later (see (6.154)), subject to constraints (3.24) to (3.30) and the constraint that the variables e_{ik}^P and e_{ik}^N lie between 0 and 1 (see Section 3.3). That is, the constraints of the relaxation of the error localisation problem for mixed data are the same as those of the error localisation problem for mixed data except that the variables e_{ik}^P and e_{ik}^N do not have to be integer; the variables e_{ik}^P and e_{ik}^N are treated as continuous non-negative numerical variables. In the relaxation of the error localisation problem for mixed data a large penalty is given in the case that a variable e_{ik}^P or e_{ik}^N is not integral. If this penalty is sufficiently high, an optimal solution to the relaxation of error localisation problem for mixed data is automatically also an optimal solution to the error localisation problem for mixed data.

The relaxation of the error localisation problem for mixed data can itself be considered to be an error localisation problem for *continuous* data. The relaxation can hence be solved by the iterative algorithms described in Section 6.7. Below we describe the relaxation of the error localisation problem for mixed data in detail. We especially focus on the *v*-constraints and the costs (weights) involved in the objective function. By an appropriate choice of these costs, the weights $\hat{w}_i$ (see (6.149) to (6.153) below), an optimal solution to the relaxation of the error localisation problem for mixed data is also an optimal solution to the error localisation problem for mixed data itself.

We denote the total number of categories of the categorical variables by G, i.e. $G = \sum_i g_i$, where g_i is the number of categories of the *i*-th categorical variable (*i*=1,…,*m*)

We introduce slack variables v_j by

$$v_j = \sum_{i=1}^{n} a_{ij}(x^0_{m+i} + z^P_{m+i} - z^N_{m+i}) + b_j - M\left(\sum_{i=1}^{m}\left(\sum_{c_{ik}\in F_i^j}(\gamma^0_{ik} + e^P_{ik} - e^N_{ik}) - 1\right)\right) \qquad (6.141)$$

for all edits *j*=1,...,*K*, and

$$v_j = e^P_{ik} \qquad (j = K + \sum_{l=1}^{i-1} g_l + k \; ; \; i=1,\ldots,m; \; k=1,\ldots, g_i), \qquad (6.142)$$

$$v_j = 1 - e^P_{ik} \qquad (j = K + G + \sum_{l=1}^{i-1} g_l + k \; ; \; i=1,\ldots,m; \; k=1,\ldots, g_i), \qquad (6.143)$$

$$v_j = e^N_{ik} \qquad (j = K + 2G + \sum_{l=1}^{i-1} g_l + k \; ; \; i=1,\ldots,m; \; k=1,\ldots, g_i), \qquad (6.144)$$

$$v_j = 1 - e^N_{ik} \qquad (j = K + 3G + \sum_{l=1}^{i-1} g_l + k \; ; i=1,\ldots,m; \; k=1,\ldots, g_i), \qquad (6.145)$$

$$v_j = z^P_{m+i} \qquad (j = K + 4G + i \; ; \; i=1,\ldots,n), \qquad (6.146)$$

$$v_j = z^N_{m+i} \qquad (k = K + 4G + n + i \; ; i=1,\ldots,n). \qquad (6.147)$$

A feasible solution to the relaxation of the error localisation problem for mixed data has to satisfy

$$\mathbf{v} \geq \mathbf{0}, \qquad (6.148)$$

where the entries of vector **v** are given by the v_j.

We associate the following weights to the slack variables:

$$\hat{w}_j = H \qquad (j = K+1,\ldots,K+2G), \qquad (6.149)$$

$$\hat{w}_j = H + w_i \qquad (j = K + 2G + \sum_{l=1}^{i-1} g_l + k \; ; \; i=1,\ldots,m; \; k=1,\ldots, g_i), \tag{6.150}$$

$$\hat{w}_j = H \qquad (j = K + 3G + \sum_{l=1}^{i-1} g_l + k \; ; i=1,\ldots,m; \; k=1,\ldots, g_i), \tag{6.151}$$

$$\hat{w}_j = w_{m+i} \qquad (j = K + 4G + i \text{ and } j = K + 4G + n + i \; ; \; i=1,\ldots,n) \tag{6.152}$$

and

$$\hat{w}_j = 0 \qquad \text{otherwise,} \tag{6.153}$$

where H is any number larger than $\sum_{j=1}^{m+n} w_j$. Equation (6.153) says that the weight $\hat{w}_j$ of any variable v_j corresponding to a cut that is created in Step 7 of the applied algorithm, either Algorithm 6.1, Algorithm 6.2, or a modified version of these algorithms (see Section 6.9 for these modifications), equals zero.

The objective function of the relaxation of the error localisation problem for mixed data is defined by

$$\sum_j \hat{w}_j \delta(v_j) \, . \tag{6.154}$$

The relaxation of the error localisation problem for mixed data is the problem of minimising (6.154) subject to (6.141) to (6.148).

Theorems 6.1 and 6.2 imply that the cuts generated by Algorithms 6.1 and 6.2 are valid for the relaxation of the error localisation problem for mixed data. In Theorem 6.6 below we demonstrate that these cuts are also valid for the error localisation problem for mixed data itself. That is, we demonstrate that the cuts cut off, i.e. are violated by, the solution to the current LP problem determined in Step 1, but are satisfied by any optimal solution to the error localisation problem for mixed data.

Theorem 6.6: Algorithms 6.1 and 6.2 generate valid cuts for the error localisation problem for mixed data.

Proof. First note that any feasible solution to the relaxation of the error localisation problem for mixed data for which $T < (2G+1)H$ is also a feasible solution to the error localisation problem for mixed data. Namely, if $T < (2G+1)H$ then e_{ik}^N and e_{ik}^P equal either zero or one for all k=1,…, g_i and i=1,…,m. For such a feasible solution to the relaxation of the error localisation problem for mixed data the value of the function (3.31) equals $T - 2GH$. Conversely, any feasible solution to the error localisation problem for mixed data is also a feasible solution to the relaxation of the error localisation problem for mixed data. The value of function (6.154) equals the value of function (3.31) plus $2GH$. This shows that an optimal solution to the relaxation of the error localisation problem for

mixed data is also an optimal solution to the error localisation problem for mixed data itself, and vice versa. Because the algorithms generate valid cuts for the relaxation of the error localisation problem for mixed data (see Theorems 6.1 and 6.2), they also generate valid cuts for the error localisation problem for mixed data itself. ■

As we mentioned before, it is not clear whether Algorithms 6.1 and 6.2 are guaranteed to be finite. However, according to Theorem 6.5, finiteness is ensured for the error localisation problem for continuous data if Algorithm 6.1 or Algorithm 6.2 is adapted according to the modifications described in Section 6.9. Theorem 6.7 below states that finiteness of the modified algorithms is also ensured for the error localisation problem for mixed data.

Theorem 6.7: With the modifications of Section 6.9 Algorithms 6.1 and 6.2 solve the error localisation problem for mixed data to optimality in a finite number of iterations.

Proof. With the modifications of Section 6.9, Algorithms 6.1 and 6.2 solve the relaxation of the error localisation problem for mixed data to optimality in a finite number of iterations (see Theorem 6.5). Because an optimum to the relaxation of the error localisation problem for mixed data is also an optimum to the error localisation problem for mixed data itself (see Theorem 6.6), the modified algorithms solve the error localisation problem for mixed data to optimality in a finite number of iterations. ■

6.12. Discussion

In this chapter we have shown how to adapt the algorithm of Glover, Klingman and Stutz for the disjunctive-facet problem in such a way that it is suited for the error localisation problem. We have demonstrated that the cuts generated by our algorithms are valid ones.

At the moment it is unknown whether the algorithms described in Section 6.7 are finite. More research is needed to answer this question. In case the algorithms turn out to be infinite, the algorithms should be adapted. A modification of the algorithms of Section 6.7 such that finiteness is ensured is given in Section 6.9. This modification complicates the procedure and possibly leads to an increase in computing time, however.

As we explained in the introduction to this chapter there are two aspects of the disjunctive-facet problem that attracted us to this problem and the solution method proposed by Glover, Klingman and Stutz (1974). One reason was that certain optimisation problems could be formulated more naturally as a disjunctive-facet problem than as a standard mixed integer programming problem. The other reason was the fact that the disjunctive-facet problem can be solved by solving a sequence of standard LP problems.

Unfortunately, when we tried to formulate the error localisation problem as a disjunctive-facet problem and use the solution method proposed by Glover, Klingman and Stutz to solve it, both the formulation of the problem and the solution method became less attractive. First, formulating the error localisation problem as a (dynamic) disjunctive-facet problem turned out to be less natural than we initially hoped. The edits of type (3.1) (see Chapter 3) rather naturally split up the linear numerical constraints into disjunctive

problems, each triggered by appropriate categorical values. We hoped to exploit this structure when formulating the error localisation problem for mixed data as a disjunctive-facet problem. However, in the error localisation problem for mixed data the categorical values may themselves be incorrect, which implies that the problem cannot be directly formulated as a disjunctive-facet problem. Besides, in the error localisation problem it is not clear beforehand which L-set requirement has to be satisfied. We need to work with a (very) large number of potential L-requirements. The final L-set requirement needs to be determined dynamically. Due to the fact that we need to work with a (very) large number of potential L-set requirements, even for the very simple and small instance of the error localisation problem of Section 6.8 the corresponding dynamic disjunctive-facet problem already contains relatively many variables and constraints. For real-life instances of the error localisation problem, the number of variables and constraints involved in the corresponding dynamic disjunctive-facet problem will be (very) high.

Second, to solve the dynamic disjunctive-facet problem for mixed data a (probably) large number of LP problems has to be solved. Each LP problem is obtained from the previous one by adding an additional constraint. It may be necessary to solve many LP problems to arrive at an optimal solution to the error localisation problem. So, the algorithm is likely to be slow. Moreover, after the termination of the algorithms proposed in this chapter we have obtained only one optimal solution to the error localisation problem. It is unclear how to find all solutions to the error localisation problem in an efficient manner once one optimal solution has been found.

Finally, the algorithms are quite complicated. It would be very difficult to implement the algorithms without making mistakes. A computer program based on these algorithms would be hard to maintain by software engineers. This basically rules out the possibility to apply the algorithms in the actual editing processes at Statistics Netherlands.

For the above reasons, we decided not to implement the proposed algorithms. Of course, we would feel grateful if anyone had the courage to implement the proposed algorithms and evaluate them by means of experiments on test data.

7. Pergamentsev's Algorithm

7.1. Introduction

In this chapter we describe a heuristic for the error localisation problem for categorical and continuous data. This heuristic has been developed by Pergamentsev, a post-graduate student at Eindhoven University of Technology who did an internship at Statistics Netherlands for nine months. At Eindhoven University of Technology Pergamentsev was supervised by Dr. Hurkens, and at Statistics Netherlands by the author of this book. Our own contribution to Pergamentsev's algorithm is limited to proposing several potential improvements.

The basic idea of the developed algorithm is to solve the error localisation problem for the numerical variables, given certain values for the categorical variables. The values for the categorical variables, and hence also whether the values of these variables need to be changed, are determined by a local search technique.

The simplicity of this approach, solving the problem for numerical data and trying to improve on the solution by local search, is its major appeal. The solutions of the algorithm will generally not be optimal ones, but we hope that its simplicity will compensate for this. The non-optimal solutions determined by Pergamentsev's algorithm could in any case serve as a benchmark for other algorithms.

The remainder of this chapter is organised as follows. Section 7.2 starts by describing the error localisation problem in numerical data as a mixed integer problem. Section 7.3 describes the basic form of the heuristic. Section 7.4 describes our potential improvements on Pergamentsev's original algorithm. Section 7.5 concludes with a brief discussion of the algorithm and the suggested improvements.

This chapter is based on De Waal (1998c).

7.2. The mixed integer programming problem for numerical variables

The error localisation problem in a mix of categorical and continuous data can be split into two parts: identifying the optimal changes in the categorical variables and subsequently, given these optimal changes in the categorical variables, identifying the optimal changes in the continuous variables. Of course, in practice it is very difficult to identify the optimal changes in the categorical variables without considering the numerical variables simultaneously. Nevertheless, the idea of splitting the error localisation problem into a categorical part and a numerical part is quite useful, and allows us to construct a heuristic.

To explain this heuristic we consider the error localisation problem for numerical data for given values of the categorical ones. This problem is generally much simpler than the problem for a mix of categorical and continuous data. It has the following structure:

$$\text{Minimise} \sum_{i=m+1}^{m+n} w_i z_i \,, \tag{7.1}$$

subject to

$$a_{1j}\Delta x_1 + \ldots + a_{nj}\Delta x_n \le \widetilde{b}_j \quad \text{for } j=1,\ldots,s, \tag{7.2}$$

$$\widetilde{L}_i z_i \le \Delta x_i \le \widetilde{U}_i z_i \qquad \text{for } i=m+1,\ldots,m+n, \tag{7.3}$$

and

$$z_i \in \{0,1\} \qquad \text{for } i=1,\ldots,m+n. \tag{7.4}$$

Here x_i^0 (i=1,…,n) denotes the original numerical values, and s the number of numerical constraints in all edits applicable to this record. Moreover, $\widetilde{b}_j = b_j - (a_{1j}x_1^0 + \ldots + a_{nj}x_n^0)$ (j=1,...,s), and $\Delta x_i = x_i - x_i^0$ is the change in the value of the i-th numerical variable. The numerical constraints are determined by the values of the categorical variables $v_1,\ldots,v_m$. The lower and upper bound on a single Δx_i are induced by the upper bound U_i and lower bound L_i on the corresponding numerical variable, namely $\widetilde{U}_i = U_i - x_i^0$ and $\widetilde{L}_i = L_i - x_i^0$ (i=1,...,n).

The above formulation allows the Δx_i to be non-zero only if the corresponding z_i are equal to 1. Thus, any change in a numerical variable is allowed only when we add the cost (reliability weight) of this variable to the objective function. Because $a_{1j}(x_1^0 + \Delta x_1) + \ldots + a_{nj}(x_n^0 + \Delta x_n) \le b_j$ for all j=1,...s, the modified values satisfy all applicable numerical constraints. The fields corresponding to those z_i that are equal to 1 should be modified. The values of the Δx_i provide us with a feasible solution, i.e. with a modified record that satisfies all edits. Having solved this mixed integer programming (MIP) problem we add the weights corresponding to changes in the categorical part of the record to obtain the real cost of the found solution. We aim the find a solution, or preferably all solutions, with the lowest possible cost.

7.3. Pergamentsev's algorithm for error localisation

Pergamentsev (1998) proposes an algorithm for solving the error localisation problem. This algorithm is presented below.

1. Fix the categorical variables to their current values (in the first iteration the current values are the original values) and solve the resulting MIP problem. Save the numerical part of the solution.
2. Check if it is possible to change any categorical variable to its original value such that the set of numerical constraints induced by the values of the categorical variables remains the same (in the first iteration this check can obviously be skipped). If so, then make this change because this obviously leads to an improvement of the cost function,

and if the number of iterations is less than a predefined number, say M_0, go to Step 1, else go to Step 3. If no categorical variable can be returned to its original value without changing the set of induced numerical constraints, go to Step 3.

3. Check if it is possible to change any categorical variable to its original value such that - although the set of numerical edits induced by the values of the categorical variables changes - the numerical part of one of the already found solutions satisfies all constraints. Whenever a solution is found in this way, the value of the cost function is evaluated and compared with the best solution found so far. If the new solution is better, then store it and try to improve it by returning one of the modified numerical values to its original value. To do this all but one of the numerical variables with values in the solution that differ from their original values are allowed to be modified, the other numerical variables are fixed on their original values. Then check if by modifying the allowed numerical variables a feasible solution to the system of linear numerical constraints can be found. If an improvement to the best solution up to date is found, this improvement is stored. If the number of iterations is less than M_0 then go to Step 1, else go to Step 4.

4. Select for every 1-step change in the categorical variables, i.e. a change in one categorical field, the violated numerical constraints and estimate the number of numerical variables that need to be modified to satisfy these constraints. We make the following observation: to satisfy a violated numerical constraint at least one numerical variable that occurs in this constraint needs to be modified. Based on this observation Pergamentsev (1998) suggests to estimate the number of numerical variables that need to be modified in a greedy way. First find the numerical variable that occurs most frequently in the violated constraints, then the one that occurs most frequently in the remainder of the violated constraints and so on. A number, say N_0, 1-step changes in categorical variables with the lowest estimated costs is selected. For each of the N_0 combinations of values of categorical variables the MIP problem from Section 7.2 is constructed, and the first improvement to the best solution so far is chosen. If no improvement is found, the best of the N_0 solutions is chosen. If the number of iterations is less than M_0 then go to Step 2, else stop.

In Step 4 of Pergamentsev's algorithm the number of numerical variables that need to be modified, given the values of the categorical variables, is estimated. In fact, this is not quite correct. Instead the sum of the reliability weights of the numerical variables that need to be modified, given the values of the categorical variables, should be estimated. We therefore propose to look not at the frequency of a numerical variable in the remaining violated constraints, but at that frequency divided by the variable's reliability weight. In each iteration of the greedy heuristic the numerical variable for which this ratio is the highest is selected in the estimated solution (see also Section 7.4.1).

We also remark that if a combination of values of categorical variables induces an impossible constraint, such as "$1 \leq 0$", the total cost associated to that combination is set to infinity. That is, in such a case this combination of values of categorical variables will never be among the N_0 best combinations of values of categorical variables.

After termination of Pergamentsev's algorithm a, probably good but not necessarily optimal, solution to the error localisation problem has been found.

7.4. Improvements on Pergamentsev's algorithm

7.4.1. Using better heuristics for solving set-covering problems

A very important part of Pergamentsev's algorithm is the estimation of the minimum sum of weights of the numerical variables that need to be changed to satisfy the edits, given the values of the categorical variables. This estimation is done in a rather crude manner. It is based on a simple greedy heuristic for solving set-covering problems. Instead of this simple heuristic more advanced hybrid heuristics proposed by Vasko and Wilson (1986) can be applied. Such hybrid heuristics have, for instance, been applied successfully in the field of statistical disclosure control (cf. Van Gelderen, 1995).

The general, or weighted, set-covering problem is given by

$$\text{Minimise} \qquad \sum_{i=1}^{k} w_i y_i \tag{7.5}$$

subject to

$$\sum_{i=1}^{k} a_{ij} y_i \geq 1, \qquad \text{for } j=1,...,t, \tag{7.6}$$

and

$$y_i \in \{0,1\}, \qquad \text{for } i=1,...,k, \tag{7.7}$$

where each a_{ij} is either 0 or 1, and the w_i are non-negative weights. For each y_i that equals 1 in a (suboptimal) solution to the set-covering problem we say that this variable *enters* the solution. In that case that all w_i equal the same positive number, say 1, the problem is known as the minimum cardinality set-covering problem.

It is clear that the observation in Step 4 of Pergamentsev's algorithm, i.e. the observation that to satisfy a violated numerical constraint it is necessary to modify at least one numerical variable that enters it, translates into a weighted set-covering problem.

Vasko and Wilson (1986) distinguish between two basic greedy heuristics for solving minimum cardinality set-covering problems. These two basic greedy heuristics are the main ingredients for their hybrid algorithms.

The first basic greedy heuristic is the same one as proposed by Pergamentsev. That is, in each iteration of the heuristic the variable that occurs most frequently in the remaining violated constraints is selected in the solution. The violated constraints in which the selected variable occurs are removed from the set of remaining violated constraints. This process goes on until the set of remaining violated edits is empty. This greedy heuristic is the most obvious one for solving set-covering problems.

Pergamentsev's Algorithm

The second basic greedy heuristic selects variables to enter the solution based on their ability to satisfy inflexible constraints, i.e. constraints with only a few non-zero a_{ij}. The rationale for this heuristic is that these inflexible constraints are the hardest ones to satisfy. If inflexible constraints are not satisfied until the end of the heuristic, it is likely that each of these constraints will require a separate variable to enter the solution. Therefore, it is a good idea to satisfy inflexible constraints as soon as possible.

Vasko and Wilson make the following two observations.

1. The first several variables to enter a solution should primarily be chosen on their ability to satisfy inflexible constraints, i.e. initially the second basic greedy heuristic should be used. However, not the entire solution should be determined by that heuristic. At some point variables should enter the solution based strictly on how many constraints they satisfy, i.e. at some point the first basic greedy heuristic should be used.
2. After a partial solution has been generated, it is likely that variables that entered the solution early in the selection process may no longer contribute as much to that solution, in terms of number of constraints satisfied, because of variables that have since entered the solution. It is therefore advantageous to determine if any variables in the solution can be replaced by a variable not in the solution, resulting in a larger number of satisfied constraints.

To construct a hybrid heuristic based on the two basic greedy ones, one should describe when to switch from considering the second basic greedy heuristic to the first one, and when to start checking if an exchange of two variables, one in the solution and one not, might be beneficial. Different choices for either when to switch from the second basic greedy heuristic to the first one, or when to start checking if an exchange might be beneficial, result in different hybrid algorithms. In Vasko and Wilson (1986) and Van Gelderen (1995) computational results for several hybrid algorithms are presented. Generally these hybrid algorithms perform better than the basic greedy algorithms.

To apply the hybrid algorithms proposed by Vasko and Wilson (1986) for the minimum cardinality set-covering problem to the weighted set-covering problem, we suggest replacing the frequency of occurrence of a variable in the remaining violated constraints by this frequency divided by the variable's weight. With this modification the hybrid algorithms proposed by Vasko and Wilson (1986) can be applied in Step 4 of Pergamentsev's algorithm. It remains to be examined which hybrid algorithms perform best for this particular estimation problem.

7.4.2. Updating the constraints of the set-covering problems

To estimate the number of numerical variables that need to be modified in Step 4 of his algorithm Pergamentsev only considers the specified edits. However, more information becomes available during the execution of the algorithm. Each time the estimated number of numerical variables that need to be modified is less than the actual number of numerical variables that need to be modified, it is apparently impossible to satisfy all edits by modifying only the numerical variables in the estimated solution. We may therefore specify that, for this particular combination of values of the categorical variables, an additional constraint should be satisfied, namely the constraint that at least one of the

numerical variables not in the estimated solution should be modified. Taking this additional constraint into account during later iterations of the algorithm may result in better estimates for the number of numerical variables that need to be modified.

7.4.3. Solving a MIP instead of checking the feasibility

In Step 3 of Pergamentsev's algorithm it is tried to change any categorical variable to its original value such that - although the set of numerical edits induced by the values of the categorical variables changes - the numerical part of one of the already found solutions satisfies all constraints. If such a solution is found, and is better than the best solution found so far, it is subsequently tried to return one of the modified numerical variables to its original value. This is done by checking the feasibility of a number of systems of linear numerical constraints. Instead one may construct a single MIP problem. In this MIP problem each numerical variable is given its original value, the values of the categorical variables are equal to their values in the current solution, and only the numerical variables in the current solution are allowed to change. The constraint is that the values of the numerical variables should satisfy all edits corresponding to the values of the categorical variables. The goal is to minimise the sum of the reliability weights of the numerical variables that need to be modified. Because generally one a few variables are allowed to be modified in this MIP problem, it can be solved relatively fast.

Solving a MIP problem instead of checking the feasibility of a number of systems of linear constraints has the advantage that the solution of the MIP problem gives the variables with the maximum sum of reliability weights that may be returned to their original values. Checking the feasibility of a number of systems of linear constraints gives only one variable that may be returned to its original value. A disadvantage is that solving a MIP problem may possibly require more time than checking a number of systems of linear constraints. However, because only a few variables are involved in the MIP problem, not much time (if any) will be gained by checking a number of systems of linear constraints instead. In any case, the time required to solve this small MIP problem will generally be negligible in comparison with the time required to solve the large MIP problems in Step 1 and Step 4 of Pergamentsev's algorithm.

7.4.4. Generating several optimal solutions

An important feature of the present versions of CherryPi, the computer program for solving the error localisation problem in numerical data based on Chernikova's algorithm (see Chapter 5) developed by Statistics Netherlands (see e.g. De Waal, 1996 and 1998b), and Leo, a prototype computer program for solving the error localisation problem in a mix of categorical and continuous data based on a non-standard branch-and-bound algorithm (see Chapter 8) also developed by Statistics Netherlands, is that they are able to generate several solutions to the error localisation problem with the same minimal weighted sum of variables that should be modified. The best, in some sense (a sense differing from the generalised Fellegi-Holt paradigm), of these solutions is subsequently selected for imputation. The algorithm of Pergamentsev, however, does not generate several solutions to the error localisation problem, but only one.

Pergamentsev's Algorithm

If one wants to apply Pergamentsev's algorithm, but nevertheless wants to generate several solutions to the error localisation problem, one can follow a number of simple approaches. First, Pergamentsev's algorithm can be applied to obtain a solution to the error localisation problem in mixed data. Assuming that in all solutions to the error localisation problem the categorical variables should have the same values, one can fix the categorical variables to their values in the obtained solution, and subsequently use, for example, the present version of CherryPi or Leo to arrive at several optimal solutions for the numerical part.

Second, the constraint can be added that at least one of the variables not in the solution should be modified. That is, if the index set S describes all variables that should be changed, the following constraint should be added

$$\sum_{\substack{i=1 \\ i \notin S}}^{m+n} z_i \geq 1 . \tag{7.8}$$

Applying Pergamentsev's algorithm again will generally result in a solution with cost at least equal to the cost of the original solution. If we find another equally good solution, we may add another constraint, namely that at least one variable not in the new solution should be modified. We can continue this process until we find a solution that is less good than the original solution. Note that, because Pergamentsev's algorithm is a heuristic, we may sometimes find a better solution if we add an additional constraint. In that case we should proceed with the best solution, and consider this as the original solution.

In principle, the second approach is better, because in the first approach only the values of the numerical variables are allowed to differ from the original solution obtained by Pergamentsev's algorithm. The drawback of the second approach is, however, that Pergamentsev's algorithm may have to be applied many times to obtain all equally good solutions. This may be very time-consuming. The algorithm implemented in the present version of CherryPi or Leo has to be applied only once to arrive at all equally good numerical parts.

Both approaches can be combined. After Pergamentsev's algorithm has found a solution we can use the present version of CherryPi or Leo to arrive at all optimal numerical parts corresponding to the categorical part of the found solution. To examine whether there exist other categorical parts that also yield the same value for the objective function as the original solution, we add the constraint that at least one of the categorical variables not in the solution should changed. That is, if S is again the index set of the variables that should be changed, the following constraint should be added

$$\sum_{\substack{i=1 \\ i \notin S}}^{m} z_i \geq 1 . \tag{7.9}$$

If the solution to this problem is equally good as the solution to the original problem, we can again apply the present version of CherryPi or Leo to arrive at all optimal numerical parts corresponding to the categorical part of this solution. This process goes on until Pergamentsev's algorithm finds a solution that is worse than the original solution.

7.4.5. Determining the quality of the solution

The number of optimal solutions to the error localisation is a measure for the quality of error localisation. We feel that in case the quality of error localisation is estimated to be low, the corresponding record should not be corrected automatically, but by other means. Many optimal solutions to the error localisation problem suggest that the quality of the final imputed record will be low, because the Fellegi-Holt paradigm is apparently not powerful enough to distinguish between these solutions and additional assumptions are necessary. Few optimal solutions suggest the quality may be high. Algorithms that determine all optimal solutions to the error localisation problem, such as the ones we propose in other chapters of this book, by definition provide this quality measure. However, as we have seen in Section 7.4.4 generating all optimal solutions by means of Pergamentsev's algorithm - possibly in combination with, for example, the present version of CherryPi or Leo - may be (too) time-consuming. So, we want a measure for the quality of error localisation that can be evaluated faster.

Such a measure can be based on the estimated sums of weights of the categorical and numerical variables that need to be modified. In fact, an estimated sum of weights of the categorical and numerical variables that need to be modified consists of the corresponding estimate for the numerical part (see Step 4 of Pergamentsev's algorithm) plus the sum of the weights of the categorical variables for which the current value differs from the original value. To use these estimates to measure the quality of error localisation a number of aspects is important. Examples of such aspects are: the number of estimates close to the sum of weights of the best solution, the differences between the estimates and the sum of weights of the best solution, and the quality of the estimates themselves.

The quality of error localisation is probably low if several of the estimates are almost equal to the sum of weights of the best solution, or if the differences between the estimates closest to the sum of weights of the best solution and this sum is small. In both cases the Fellegi-Holt paradigm is not powerful enough to select a clearly best solution. The quality of error localisation is probably high if the sum of weights of the best solution is clearly less than the estimates, and the distances between estimates closest to the sum of weights of the best solution and this sum is large. In this case the Fellegi-Holt paradigm succeeds in selecting a clearly best solution.

If the quality of the estimates is low, which can be evaluated by comparing the estimates for the numerical part to the solutions of the MIP problems in Step 4, the conclusion drawn about the quality of error localisation is not very trustworthy. Even if the conclusion would be that the quality of error localisation is fairly high, one may decide not to edit the record under consideration automatically in case the quality of the estimates for the numerical part in Step 4 is low. This is an extra precaution to ensure that the quality of the carried out error localisation will be sufficiently high.

7.4.6. Using other local search strategies

The best changes in the categorical variables in Pergamentsev's algorithm are found by a simple heuristic. Instead other heuristics, for example simulated annealing and tabu search, may be used to find the best changes. Below we briefly describe simulated annealing and tabu search.

Simulated annealing is based on an analogy between cooling of solids to reach a low level energy ground state and solving a combinatorial optimisation problem. At each evaluation of a neighbouring solution, we always accept an improvement of the current solution. However, we may also accept worse solutions. A worse solution is accepted with a certain probability.

The rationale of accepting worse solutions is to avoid getting trapped in a local minimum. When only improvements would be accepted, the process stops as soon as a local minimum, not necessarily a global minimum, has been found.

The probability of accepting a worse solution depends on both the change in cost of the candidate solution and the current temperature of the system. The temperature is analogous to the temperature in physical annealing. The temperature is lowered throughout the annealing process. At high temperatures almost any candidate solution is accepted, while at lower temperatures candidate solutions worse than the current solution are only rarely accepted. The idea behind reducing temperature is that near the end of the annealing process only the neighbourhood of a locally optimal solution is explored.

The temperature is reduced by an appropriate cooling schedule. One of the most applied cooling schedules is to start at some initial temperature *T*, and after a certain number of iterations reduce *T* by a factor *K*, where *K* is less than 1. The number of repetitions at each temperature may be varied depending on the size of the neighbourhood and the temperature. The performance of a simulated annealing algorithm often strongly depends on the cooling schedule employed.

At Statistics Netherlands simulated annealing has been applied to round tables in a controlled manner (cf. Bakker, 1997). For more information on simulated annealing we refer to Van Laarhoven and Aarts (1987).

Tabu search is a recent heuristic technique used for solving combinatorial optimisation problems. Tabu search is based on using information gained earlier in the search process to guide the search. When the neighbourhood of a solution is evaluated some potential solutions are considered tabu based on the history of the search process. In each iteration we go to the best neighbouring solution that is not tabu, even if this solution is worse than the current one. The rationale of this approach is again that we want to avoid being trapped in a local minimum.

A simple method is to record the recent steps and classify these steps, or the corresponding reverse steps, as tabu. For example, in the error localisation problem we may say that when a 1-step change in the categorical variables is made, the corresponding reverse 1-step change becomes tabu for a certain period of time.

7.5. Discussion

From a theoretical point of view Pergamentsev's algorithm offers many possibilities for future research. First, in Pergamentsev's algorithm two parameters, the number of iterations (M_0) and the number of 1-step changes in the categorical variables for which a MIP problem is solved (N_0), should be specified. At the moment it is unclear what appropriate values for M_0 and N_0 are.

Second, it remains to be examined which hybrid heuristic is best-suited for estimating the weighted number of numerical variables that need to be modified in Step 4 (see Section 7.4.1). In fact, it is not clear that such a hybrid heuristic should be used, other algorithms may be more appropriate for estimating this value.

Third, in Section 7.4.2 a simple method is described to update the constraints of the set-covering problems. Other, more advanced, methods might be developed that give better results. Moreover, it remains to be investigated whether updating the constraints really improves the quality of the estimates.

Fourth, it remains to be investigated if it is beneficial to solve a MIP problem instead of checking the feasibility of several systems of linear constraints (see Section 7.4.3). Solving a MIP problem may possibly increase the computing time in each iteration, but on the other hand it may also lead to a better solution after the completion of the iteration.

Fifth, to generate several optimal solutions to the error localisation problem in mixed data three simple approaches have been described. One approach aims to find only the optimal solutions that share the same categorical part as the original solution determined by Pergamentsev's algorithm, the other two approaches aim to find all optimal solutions. The later two approaches are more time-consuming than the former one. It remains to compare these approaches in terms of both computing time and quality. Besides, other approaches may be developed to generate several optimal solutions.

Sixth, to determine the quality of error localisation three constituents for a quality measure are mentioned in Section 7.4.5. It is an open question how these constituents should be combined to yield a good quality measure. It is quite likely that a good quality measure should also incorporate other ingredients. Such other ingredients remain to be found.

Seventh, instead of the search technique applied in Pergamentsev's algorithm one can apply other search techniques. Two general techniques, simulated annealing and tabu search, have been briefly described in Section 7.4.6. It has yet to be determined which local search technique is best for the error localisation problem in mixed data.

From a practical point of view, however, Pergamentsev's algorithm offers less perspective. As we mentioned in Section 7.1 the major appeal of the algorithm was its apparent simplicity. Unfortunately, implementation of the algorithm in a computer program turned out to be much more complex than was anticipated. In fact, although two persons worked on it for several months, we never succeeded in obtaining a fully operational version of the intended computer program.

Besides the unexpected complexity, a computer program based on Pergamentsev's algorithm would have several other disadvantages. A practical advantage would be that the error localisation problem for numerical data would need to be solved by means of another algorithm. This implies that two different algorithms need to be maintained by the statistical office.

Another disadvantage is that Pergamentsev's algorithm does not necessarily determine all optimal solutions to the error localisation problem. Because the solution returned by Pergamentsev's algorithm need not be optimal, the quality of the record that is edited and the quality of the determined solution are unclear. Additional steps must be taken in order to get some idea about the quality of the record or the determined solution. To obtain more

than one solution the algorithm needs to be extended slightly, thereby making it even more complex.

For the above reasons Pergamentsev's algorithm has never been fully implemented in a computer program. Therefore the algorithm will obviously not be evaluated in the chapter on computational results, Chapter 11, of this book. We do not consider this to be a serious omission as the evaluation experiments in Chapter 11 are done on purely numerical data only, and Pergamentsev's algorithm needs another algorithm in order to solve the error localisation problem for purely numerical data anyway.

8. A Branch-and-Bound Algorithm

8.1. Introduction

In 1999 Quere, a post-graduate student at Eindhoven University of Technology, did an internship at Statistics Netherlands. During his internship Quere worked on the error localisation problem for purely numerical data. During this period he was supervised by Dr. Hurkens from Eindhoven University of Technology and the author of this book. Quere succeeded in developing an algorithm that seemed so promising that after his internship had ended Statistics Netherlands hired Quere for a short period, during which he was again supervised by the author of this book, in order to jointly extend the developed algorithm to a mix of categorical and continuous data.

In this chapter the result of this work, a branch-and-bound algorithm for solving the error localisation problem for a mix of categorical and continuous data, is described. The algorithm is based on constructing a binary tree, and subsequently searching this tree for optimal solutions to the error localisation problem. In a standard branch-and-bound tree in each node of the tree a variable is selected. Subsequently, branches are constructed by fixing the selected variable to (some of) its possible values. In the branch-and-bound algorithm by Quere and De Waal, however, two branches are constructed by fixing the selected variable to its original value in one branch and by eliminating the selected variable in the other branch. To eliminate a variable, the algorithm by Quere and De Waal uses implicit edits, just like the Fellegi-Holt method. We give a mathematical description of the branching part of this algorithm in Section 8.2. An example illustrating the algorithm is given in Section 8.3. A proof that the proposed algorithm indeed generates all optimal solutions to the error localisation problem is given in Section 8.4. In Section 8.5 we examine what happens if we allow categorical variables to be selected before all numerical variables have been selected. In Section 8.6 computational aspects of the algorithm are briefly considered. The material in that section is partly based on work carried out by Daalmans when he was doing an internship at Statistics Netherlands as a student at Tilburg University. At Tilburg University Daalmans was supervised by Prof. Dr. Magnus, and at Statistics Netherlands by the author of this book. In Section 8.6 we also describe the bounding part of the algorithm. Finally, Section 8.7 concludes this chapter with a short discussion.

This section is based on Quere and De Waal (2000), De Waal (2000a), and Daalmans (2000). Part of this material is planned to be published in De Waal and Quere (2003).

8.2. A branching algorithm

We first assume that no values are missing. The basic idea of the algorithm is then that a binary tree is constructed. In each node of this tree a variable is selected that has not yet been selected in any predecessor node. If all variables have already been selected in a predecessor node, we have reached a terminal node of the tree.

After selection of a variable two branches are constructed: in one branch the selected variable is fixed to its original value, in the other branch the selected variable is eliminated from the set of current edits. In each branch the current set of edits is updated. A variable that has either been fixed or eliminated is said to have been treated. The algorithm we propose is based on a depth-first search: the current branch is searched for solutions to the error localisation problem until its terminal nodes are reached before another branch is searched.

For simplicity we assume in this section that all numerical variables are selected before any categorical variable is selected. In Section 8.5 we briefly consider the more general, and more complicated, case where categorical variables may be selected before all numerical variables have been selected. In particular, we show that the resulting implied edits are more difficult than when all numerical variables are selected before any categorical variable.

Fixing a variable to its original value corresponds to assuming that this original value is correct, eliminating a variable from the set of current edits corresponds to assuming that the original value of this variable is incorrect and has to be modified.

Updating the set of current edits is the most important step in the algorithm. How the set of edits has to be updated depends on whether the selected variable was fixed or eliminated, and also on whether this variable was categorical or continuous.

Fixing a variable, either numerical or categorical, to its value is easy. We simply substitute this value in all current edits. Note that, given that we fix this variable to its original value, the new set of current edits is a set of implicit edits for the remaining variables in the tree, i.e. the remaining variables have to satisfy the new set of edits. As a result of fixing the selected variable to its value some edits may become always satisfied, for example when a categorical variable is fixed to a value such that the IF-condition of an edit can never become true anymore. These edits may be discarded from the new set of edits. Conversely, some edits may become violated. In such a case this branch of the binary tree can never result in a solution to the error localisation problem.

Eliminating a variable is a relatively complicated process. It amounts to generating a set of implicit edits that do not involve this variable. That set of implicit edits becomes the current set of edits corresponding to the current branch of the tree.

If a numerical variable is to be eliminated, we basically apply Fourier-Motzkin elimination (see Duffin, 1974; Chvátal, 1983; Imbert, 1993; Schrijver, 1986; Korte and Vygen, 2000; Quere, 2000; Quere and De Waal, 2000) to eliminate that variable from the set of edits. Some care has to be taken in order to ensure that the IF-conditions of the resulting edits are correctly defined.

In particular, if we want to eliminate a numerical variable x_r from the current set of edits, we start by copying all edits not involving this numerical variable from the current set of edits to the new set of edits. Next, we examine all edits of the type we consider in this book, i.e. of type

IF $v_i \in F_i^j$ for i=1,…,m

THEN $(x_1,...,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + ... + a_{nj}x_n + b_j \geq 0\}$, (8.1a)

or

IF $v_i \in F_i^j$ for i=1,…,m

THEN $(x_1,...,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + ... + a_{nj}x_n + b_j = 0\}$, (8.1b)

involving x_r pair-wise. Suppose we consider the pair consisting of edit s and edit t.

We start by checking whether the intersection of the IF-conditions is non-empty, i.e. whether the intersections $F_i^s \cap F_i^t$ are non-empty for all i=1,…,m. If any of these intersections is empty, we do not have to consider this pair of edits anymore. So, suppose that all intersections are non-empty. We now construct an implicit edit. If the THEN-condition of edit s is an equality, we use the equality

$$x_r = -\frac{1}{a_{rs}}\left(b_s + \sum_{i \neq r} a_{is}x_i\right) \tag{8.2}$$

to eliminate x_r from the THEN-condition of edit t. Similarly, if the THEN-condition of edit s is an inequality and the THEN-condition of edit t is an equality, the equality in the THEN-condition of edit t is used to eliminate x_r.

If the THEN-conditions of both edit s and edit t are inequalities, we check whether the coefficients of x_r in those inequalities have opposite signs. That is, we check whether $a_{rs} \times a_{rt} < 0$. If the coefficients of x_r in the two inequalities do not have opposite signs, we do not consider this pair of edits anymore.

If the coefficients of x_r in the two inequalities do have opposite signs, one of the inequalities can be written as a lower bound on x_r and the other as an upper bound on x_r. Combining these two bounds leads to an inequality not involving x_r. We generate the THEN-condition:

$$(x_1,...,x_n) \in \{\mathbf{x} \mid \widetilde{a}_1x_1 + ... + \widetilde{a}_nx_n + \widetilde{b} \geq 0\}, \tag{8.3}$$

where

$$\widetilde{a}_i = |a_{rs}| \times a_{it} + |a_{rt}| \times a_{is} \quad \text{for all } i=1,\ldots,n \tag{8.4}$$

and

$$\tilde{b} = |a_{rs}| \times b_t + |a_{rt}| \times b_s . \qquad (8.5)$$

Note that x_r indeed does not enter the resulting THEN-condition.

The above THEN-condition forms the THEN-condition of a new implied edit. The IF-condition of this implicit edit is given by the intersections $F_i^s \cap F_i^t$ for all i=1,…,m. That the IF-condition of the new implicit edit is given by the *intersections* $F_i^s \cap F_i^t$ (i=1,…,m) is intuitively clear: two numerical THEN-conditions can only be combined into the (numerical) THEN-condition of an implicit edit for the overlapping parts of their corresponding categorical IF-conditions. Taking the intersections of the categorical IF-conditions provides a natural and convenient way of keeping track of the combinations of categorical values for which the numerical THEN-condition of an implicit edit is defined. Note that if we eliminate a numerical variable in any of the ways described above, the resulting set of edits is, given that we allow the eliminated variable to attain any possible value, a set of implicit edits for the remaining variables in the tree. That is, this resulting set of edits has to be satisfied by the remaining variables in the tree, given that the eliminated variable may in principle take any real value.

Repeatedly applying the above elimination process until all numerical variables have been eliminated results in a THEN-condition not involving any unknowns that is either true, for example "$1 \geq 0$", or a THEN-condition that is false, for example "$0 \geq 1$". The edits for which the THEN-condition is true are discarded.

Categorical variables are only treated, i.e. fixed or eliminated, once all numerical variables have been treated. So, once the categorical variables may be selected the edits in the current set of edits all have the following form:

$$\begin{array}{ll} \text{IF} & v_i \in F_i^j \qquad \text{for } i=1,\ldots,m \\ \text{THEN} & (x_1,\ldots,x_n) \in \varnothing . \end{array} \qquad (8.6)$$

To eliminate categorical variable v_r from the set of edits given by (8.6), we start by copying all edits not involving this variable to the set of implicit edits.

Next, we basically apply the method of Fellegi and Holt to the IF-conditions to generate the IF-conditions of the implicit edits (see also Daalmans, 2000). In the terminology of Fellegi and Holt, field v_r is selected as the generated field. We start by determining all index sets S such that

$$\bigcup_{j \in S} F_r^j = D_r \qquad (8.7)$$

and

$$\bigcap_{j \in S} F_i^j \neq \varnothing \qquad \text{for all } i=1,\ldots,r\text{-}1,r\text{+}1,\ldots,m. \qquad (8.8)$$

From these index sets we select the *minimal* ones, i.e. the index sets S that obey (8.7) and (8.8), but none of their subsets obey (8.7).

Given such a minimal index set S we construct the implied edit given by

$$\begin{aligned} &\text{IF} \quad && v_r \in D_r,\ v_i \in \bigcap_{j \in S} F_i^j \ \text{ for } i=1,\ldots,r\text{-}1,r\text{+}1,\ldots,m \\ &\text{THEN} \quad && (x_1,\ldots,x_n) \in \varnothing \,. \end{aligned} \tag{8.9}$$

Note that if we eliminate a categorical variable in the way described above, the resulting set of edits is, given that we allow the eliminated variable to attain any possible value in its domain, a set of implicit edits for the remaining variables in the tree. That is, this resulting set of edits has to be satisfied by the remaining variables in the tree, given that the eliminated variable may in principle take any value in its domain.

If values are missing in the original record, the corresponding variables only have to be eliminated (and not fixed) from the set of edits, because these variables always have to be imputed. At precisely what moment the variables with missing variables are eliminated is not important for obtaining all optimal solutions to the error localisation problem as long as all numerical variables are treated before any categorical variable is (see, however, also Section 8.5). However, a natural choice is to treat the variables in the following order:

- eliminate all numerical variables with missing values;
- fix or eliminate the remaining numerical variables;
- eliminate all categorical variables with missing values;
- fix or eliminate the remaining categorical variables.

We have now explained how the current set of edits changes if we fix or eliminate a variable. After all categorical variables have been treated we are left with a set of relations without any unknowns. This set of relations may be the empty set. These relations may either be contradictions or not. A contradicting relation is given by

$$\begin{aligned} &\text{IF} \quad && v_i \in D_i \ \text{ for } i=1,\ldots,m \\ &\text{THEN} \quad && (x_1,\ldots,x_n) \in \varnothing \,. \end{aligned} \tag{8.10}$$

If the set of relations is empty, it does not contain any contradictions. The relations contain no contradictions if and only if the variables that have been eliminated in order to reach the corresponding terminal node of the tree can be imputed consistently, i.e. such that all original edits can be satisfied. This statement is proved in Section 8.4.

In the algorithm we check for each terminal node of the tree whether the variables that have been eliminated in order to reach this node can be imputed consistently. Of all sets of variables that can be imputed consistently we select the ones with the lowest sums of reliability weights. In this way we find all optimal solutions to the error localisation problem (see also Section 8.4).

Equalities in THEN-conditions can be handled more efficiently than we described so far. For instance, if the numerical variable to be eliminated is involved in an equality that has to hold irrespective of the values of the categorical variables, i.e. is involved in an edit of the following type

$$\begin{aligned} &\text{IF} && v_i \in D_i \quad \text{for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}, \end{aligned} \tag{8.11}$$

then we do not have to consider all edits pair-wise in order to eliminate this variable. Instead, we only have to combine (8.11) with all other current edits. So, if there are J current edits, we do not have to consider $J(J-1)$ pairs, but only $J-1$ pairs. Besides, the number of resulting implied edits is generally less than when all pairs of current edits are considered. We will refer to this rule as the *equality-elimination rule*. In principle, similar rules can be developed for other types of edits involving an equality as THEN-condition. We will not pursue this path in this book, however.

The algorithm described in this chapter is a so-called branch-and-bound algorithm. In a branch-and-bound algorithm a tree is constructed and bounds on the objective function are used to cut off branches from the tree. In Section 8.6 we briefly explain how our bound is calculated and branches can be cut off from our tree.

8.3. Example

In this section we illustrate the idea of the algorithm presented in the previous section by means of an example. This example is similar to an example given in Quere and De Waal (2000). . In this example we will not use the equality-elimination rule described at the end of the previous section. We will not build the entire tree, because this would take too much space and would hardly teach us anything. Instead we will only generate one branch of the tree.

Suppose we have to edit a data set containing four categorical variables v_i (i=1,…,4) and three numerical variables x_i (i=1,…,3). The domains of the first two categorical variables are {1,2}, and of the last two categorical variables {1,2,3}. The set of explicit edits is given below.

$$\text{IF } (v_1 = 1 \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing \tag{8.12}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 1) \text{ THEN } \varnothing \tag{8.13}$$

$$\text{IF } (v_1 = 2 \text{ AND } v_3 \in \{1,3\} \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing \tag{8.14}$$

$$x_1 - 12 \geq 0 \tag{8.15}$$

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } x_2 = 0 \tag{8.16}$$

$$\text{IF } (v_3 = 2) \text{ THEN } x_2 - 1250 \geq 0 \tag{8.17}$$

$$\text{IF } (v_3 = 2) \text{ THEN } -875x_1 + 12x_2 \geq 0 \tag{8.18}$$

$$\text{IF } (v_3 = 2) \text{ THEN } 1250x_1 - 8x_2 \geq 0 \tag{8.19}$$

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } 1250x_1 - x_3 = 0 \tag{8.20}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } 1250x_1 + 12x_2 - x_3 + 1250 = 0 \tag{8.21}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } 1250x_1 + 12x_2 - x_3 = 0 \tag{8.22}$$

Here, if a categorical variable is not mentioned in an IF-condition, this variable may take any value. For instance, edit (8.12) actually means

$$\text{IF } (v_1 = 1 \text{ AND } v_2 \in D_2 \text{ AND } v_3 \in D_3 \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing, \tag{8.23}$$

where D_i is the domain of categorical variable *i*.

Now, suppose that a record with values $v_1 = 1$, $v_2 = 2$, $v_3 = 2$, $v_4 = 1$, $x_1 = 25$, $x_2 = 3{,}050$ and $x_3 = 90{,}000$ is to be edited. Edits (8.12) and (8.21) are failed, so this record is inconsistent. We apply the algorithm described in the previous section and start by selecting a numerical variable, say x_1. In the algorithm two branches are generated: one branch where x_1 is fixed to its original value 25, and one branch where x_1 is eliminated from the current set of edits. Here we only consider the second branch and eliminate x_1 from the current set of edits.

For instance, if we combine (8.15) and (8.18), we first take the intersection of their IF-conditions. This intersection is given by "$v_3 = 2$". This intersection is non-empty, so we proceed. We write (8.15) as a lower bound on x_1, i.e. as $x_1 \geq 12$, and the THEN-condition of (8.18) as an upper bound on x_1, i.e. as $875x_1 \leq 12x_2$. The THEN-condition of the resulting implied edit is then given by $12x_2 \geq 875 \times 12$, or equivalently by $x_2 \geq 875$. The resulting implied edit is hence given by

$$\text{IF } (v_3 = 2) \text{ THEN } x_2 - 875 \geq 0 \tag{8.24}$$

The complete set of resulting (implicit) edits is given by (8.24) and:

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } x_3 \geq 15000 \tag{8.25}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 12x_2 - 16250 \geq 0 \tag{8.26}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 12x_2 - 15000 \geq 0 \tag{8.27}$$

$$\text{IF } (v_3 = 2) \text{ THEN } x_2 \geq 0 \tag{8.28}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } 20.4x_2 - 0.7x_3 + 875 \geq 0 \tag{8.29}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } 20.4x_2 - 0.7x_3 \geq 0 \tag{8.30}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } -20x_2 + x_3 - 1250 \geq 0 \tag{8.31}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } -20x_2 + x_3 \geq 0 \tag{8.32}$$

and (8.12), (8.13), (8.14), (8.16) and (8.17).

Note that some of the generated edits may be completely useless. For instance, implicit edit (8.28) is less strong than, is *dominated by*, edit (8.24). If edit (8.24) is satisfied, then automatically edit (8.28) is satisfied. Such dominated edits may be deleted.

We select another numerical variable, say x_2, and again construct two branches: one branch where x_2 is fixed to its original value 3,050, and one branch where x_2 is eliminated from the current set of edits. Here we only consider the first branch and fix x_2 to its original value. As a result, some of the current edits may become satisfied. Those edits can be discarded. In this case, for example, edit (8.24) becomes satisfied and is discarded in the current branch of the tree. Some other edits may become violated. In such a case the current branch of the tree cannot lead to a solution to the error localisation problem. In our example none of the edits becomes violated.

The resulting set of implicit edits obtained by fixing x_2 to its original value is given by:

$$\text{IF } (v_3 \in \{1,3\}) \text{ THEN } \varnothing \tag{8.33}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 52850 \geq 0 \tag{8.34}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 51600 \geq 0 \tag{8.35}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } -0.7x_3 + 63095 \geq 0 \tag{8.36}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } -0.7x_3 + 62220 \geq 0 \tag{8.37}$$

$$\text{IF } (v_2 = 2 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 61000 \geq 0 \tag{8.38}$$

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } x_3 - 62250 \geq 0 \tag{8.39}$$

and (8.12), (8.13), (8.14) and (8.25). Edit (8.33) arises from edit (8.16) by substituting 3,050 for x_2. The resulting numerical THEN-condition is failed.

We select the final numerical variable, x_3, and split the tree into two branches: one where x_3 is fixed to its original value and one where it is eliminated. Here we only consider the branch where x_3 is fixed to its original value, 90,000. Again some of the edits become satisfied and are discarded. None of the edits become violated in our case. The resulting set of implicit edits is given by:

$$\text{IF } (v_2 = 1 \text{ AND } v_3 = 2) \text{ THEN } \varnothing, \tag{8.40}$$

and (8.12), (8.13), (8.14) and (8.33). Edit (8.40) arises from edit (8.37) by substituting 90,000 for x_3. The resulting numerical THEN-condition is failed.

All numerical variables have now been treated, either by fixing or by eliminating. We see that the current set of edits is given by the purely categorical explicit edits supplemented by categorical edits that have been generated when the numerical variables were treated. We now treat the categorical variables. We select a categorical variable, say v_1, and again split the tree into two branches: a branch where v_1 is fixed to its original value and a branch where it is eliminated. We only consider the branch where v_1 is eliminated. The resulting set of implicit edits is given by:

$$\text{IF } (v_3 \in \{1,3\} \text{ AND } v_4 \in \{1,3\}) \text{ THEN } \varnothing, \tag{8.41}$$

and (8.13), (8.33) and (8.40).

We select a categorical variable, say v_2. Fixing and eliminating this variable again results in two branches. We only consider the branch where v_2 is fixed to its original value, 2. We obtain only two implicit edits, namely (8.33) and (8.41).

Again, we select a categorical variable, say v_3. Fixing and eliminating this variable again results in two branches. We only consider the branch where v_3 is fixed to its original value, 2. The resulting set of implicit edits is empty.

This implies that the set of original, explicit edits can be satisfied by changing the values of x_1 and v_1, and fixing the other variables to their original values. In other words, a solution to the error localisation problem for this record is given by: change the values of x_1 and v_1. Possible values, the only ones in this case, are $v_1 = 2$ and $x_1 = 41.72$. It is easy to check that the resulting record indeed satisfies all explicit edits.

The other branches of the tree, which we have skipped, also need to be examined, because it is possible that they contain a better solution to the error localisation problem. By examining all branches of the tree one can obtain all optimal solutions to the error localisation problem for the record under consideration.

8.4. An optimality proof

In this section we prove that the algorithm described in Section 8.2 indeed finds all optimal solutions to the error localisation problem. We do this in three steps. We start by showing that if the variables that have been eliminated in order to reach a certain node can be imputed in such a way that the set of edits corresponding to this node become satisfied, the variables that have been eliminated in order to reach the parent node can also be imputed in such a way that the set of edits corresponding to that parent node become satisfied. This is the lifting principle that was introduced in Chapter 4. Using this result we show that if and only if the set of relations without any unknowns in a terminal node do not contradict each other, we can consistently impute the variables that have been eliminated in order to reach this terminal node, i.e. such that the original edits become satisfied. The final step consists of noticing that the terminal nodes correspond to all potential solutions of the error localisation problem, and hence that the algorithm indeed determines all optimal solutions to the error localisation problem. Steps 2 and 3 are trivial once the first step has been proved.

The proof of the first step, the lifting principle, is similar to the proof of Theorem 1 in Fellegi and Holt (1976) (see also Chapter 4 of the present book). The main differences are that our edits are more general than the edits considered by Fellegi and Holt, and that Fellegi and Holt assume that the so-called complete set of (explicit and implicit) edits has been generated.

Theorem 8.1. Suppose the set of variables in a certain node is given by T_0, and the current set of edits corresponding to that node by Ω_0. Suppose, furthermore, that a certain variable r is either fixed or eliminated. Denote the set of resulting variables by T_1, $T_1 = T_0 - \{r\}$, and the set of edits corresponding to the next node by Ω_1. Then there exist values u_i^0 for the variables in T_1 that satisfy the edits in Ω_1 if and only if there exists a

value u_r^0 for variable *r* such that the values u_i^0 for the variables in T_0 satisfy the edits in Ω_0.

Proof. It is easy to verify that if there exist values u_i^0 for the variables in T_0 that satisfy the edits in Ω_0 then the same values (except the value of the variable that is fixed or eliminated) automatically satisfy the edits Ω_1 of the next node.

It is a bit more work to prove the other part of the proof. We have to distinguish between several cases. First, let us suppose that the selected variable is fixed. This is a trivial case. It is clear that if there exist values u_i^0 for the variables in T_1 that satisfy the edits in Ω_1, there exist values u_i^0 for the variables in T_0 that satisfy the edits in Ω_0. Namely, for the fixed variable *r* we set the value u_r^0 equal to the original value of *r*.

Let us now suppose that a categorical variable *r* has been eliminated. Suppose that there exist values u_i^0 for the variables in T_1 that satisfy the edits in Ω_1, but there does not exist a value u_r^0 for the selected variable *r* such that the variables in T_0 satisfy the edits in Ω_0. Identify a failed edit in Ω_0 for each possible value v_r^k of variable *r*. The index set of these failed edits need not be a minimal one. We therefore remove some of these failed edits such that the corresponding index set *S* becomes minimal. We then construct the implicit edit given by (8.9).

Edit (8.9) is an element of Ω_1. Moreover, the values u_i^0 for the variables in T_1 do not satisfy this edit. This contradicts our assumption that these values satisfy all edits in Ω_1. So, we can conclude that a value u_r^0 for the selected variable *r* exists such that the values u_i^0 for variables in T_0 satisfy the edits in Ω_0.

Finally, let us suppose that a numerical variable *r* has been eliminated. Suppose that there exist values u_i^0 for the variables in T_1 that satisfy the edits in Ω_1. Each edit in Ω_1 is either obtained from copying the edits in Ω_0 not involving variable *r*, or from two edits in Ω_0 involving variable *r* that have been combined.

It is clear that if the edits in Ω_1 that have been obtained from copying the edits in Ω_0 not involving variable *r* are satisfied by values u_i^0 for the variables in T_1, these edits in Ω_0 are also satisfied by the same values for the variables in T_0.

It remains to prove that if the edits in Ω_1 that have been obtained by combining two edits in Ω_0 are satisfied by the variables in T_1, there exists a value for variable *r* such that all edits in Ω_0 involving variable *r* can be satisfied. To show this we fill in the values u_i^0 for the variables in T_1 in the edits in Ω_0. As a result, we obtain a number of constraints of the following types for the value of the selected variable *r*:

$$x_r = M_k^E \tag{8.42}$$

$$x_r \geq M_k^L, \tag{8.43}$$

and

$$x_r \leq M_k^U. \tag{8.44}$$

Here M_k^E, M_k^L, and M_k^U are certain constants for k=1,…,K_r, where K_r is the number of edits in Ω_0 involving variable r that are triggered by the values u_i^0 of the categorical variables in T_0.

Constraint (8.42) has been obtained from an edit k in Ω_0 of which the THEN-condition can be written in the following form

$$x_r = \sum_{i \neq r} a'_{ik} x_i + b'_k \tag{8.45}$$

by filling in the values u_i^0 for the variables in T_1. Similarly, constraints (8.43) and (8.44) have been obtained from edits in Ω_0 of which the THEN-conditions can be written in the following forms

$$x_r \geq \sum_{i \neq r} a'_{ik} x_i + b'_k \tag{8.46}$$

and

$$x_r \leq \sum_{i \neq r} a'_{ik} x_i + b'_k, \tag{8.47}$$

respectively, by filling in the values u_i^0 for the variables in T_1.

If the constraints given by (8.42) to (8.44) do not contradict each other, we can find a value for variable r such that this value plus the values u_i^0 for the variables in T_1 satisfy the edits in Ω_0.

So, suppose the constraints given by (8.42) to (8.44) do contradict each other. These constraints can only contradict each other if there are constraints s and t given by

1. $x_r = M_s^E$ and $x_r = M_t^E$ with $M_s^E \neq M_t^E$, (8.48)

2. $x_r = M_s^E$ and $x_r \geq M_t^L$ with $M_s^E < M_t^L$, (8.49)

3. $x_r \leq M_s^U$ and $x_r = M_t^E$ with $M_s^U < M_t^E$, (8.50)

or

4. $x_r \leq M_s^U$ and $x_r \geq M_t^L$ with $M_s^U < M_t^L$. (8.51)

In case 1 constraints *s* and *t* have been derived from edits in Ω_0 of which the THEN-conditions are equalities. The IF-conditions of these edits have a non-empty intersection, because both edits are triggered when we fill in the values u_i^0 for the categorical variables in T_1. So, these edits generate an implicit edit in Ω_1 if we eliminate variable *r*. The THEN-condition of this implicit edit can be written as

$$\sum_{i \neq r} a'_{is} x_i + b'_s = \sum_{i \neq r} a'_{it} x_i + b'_t , \tag{8.52}$$

where we have used (8.45).

Filling in the values u_i^0 for the variables in T_1 in this implicit edit, we find that M_s^E should be equal to M_t^E. In other words, we have constructed an edit in Ω_1 that would be failed if we filled in the values u_i^0 for the variables in T_1. This contradicts the assumption that these values satisfy all edits in Ω_1, and we conclude that the constraints given by (8.42) cannot contradict each other.

For cases 2, 3 and 4 we can show in a similar manner that we would be able to construct a failed implicit edit in Ω_1. This contradicts the assumption that the values u_i^0 for the variables in T_1 satisfy all edits in Ω_1, and we conclude that the constraints given by (8.42) to (8.44) cannot contradict each other.

In turn this allows us to conclude that a value for variable *r* exists such that this value plus the values u_i^0 for the variables in T_1 satisfy the edits in Ω_0.

Finally, note that when the equality-elimination rule (see the end of Section 8.2) has been applied to eliminate a continuous variable *r* by means of the edit

$$\begin{aligned} &\text{IF} && v_i \in D_i \quad \text{for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1s}x_1 + \ldots + a_{ns}x_n + b_s = 0\}, \end{aligned} \tag{8.53}$$

the value given by

$$x_r = -\frac{1}{a_{rs}} \left(b_s + \sum_{i \neq r} a_{is} u_i^0 \right) \tag{8.54}$$

plus the values u_i^0 for the variables in T_1 satisfy the edits in Ω_0.

This concludes the proof of Theorem 8.1. ■

Theorem 8.2. The set of edits corresponding to a terminal node, i.e. a set of relations without any unknowns, is consistent, i.e. contains no contradictions, if and only if the variables that have been eliminated in order to reach this terminal node can be imputed in such a way that the original set of edits becomes satisfied.

Proof. This follows directly from a repeated application of Theorem 8.1. ■

Theorem 8.3. The algorithm of Section 8.2 determines all optimal solutions to the error localisation problem.

Proof. The terminal nodes of the tree correspond to all possible combinations of fixing and eliminating variables. So, according to Theorem 8.2 above, the algorithm checks which of all possible sets of variables can be imputed consistently. The algorithm simply selects all optimal sets of variables that can be imputed consistently from all possible sets. So, we can conclude that the algorithm finds all optimal solutions to the error localisation problem. ■

8.5. The order of treating numerical and categorical variables

In this section we briefly consider what happens if categorical variables are treated before all numerical ones have been treated. We will see that treating the categorical variables before all numerical variables have been treated leads to a more complicated algorithm than the one described in Section 8.2, because the problem has to be split into several subproblems.

Even if we allow categorical variables to be treated before all numerical variables have been treated, fixing a variable or eliminating a numerical variable x_r from the current set of edits is done in the same way as in Section 8.2. Only eliminating categorical variables has to be done a bit differently as in Section 8.2.

To eliminate categorical variable v_r from a set of edits given by (8.1), we start by copying all edits not involving this variable to the new set of edits. Next, we determine all minimal index sets S_k such that (8.7) and (8.8) are satisfied.

Given such a minimal index set S_k we construct the implied edit given by

$$\begin{aligned} &\text{IF} \quad v_r \in D_r,\ v_i \in \bigcap_{j \in S_k} F_i^j \ \text{ for } i=1,\ldots,r\text{-}1,r\text{+}1,\ldots,m \\ &\text{THEN} \quad (x_1,\ldots,x_n) \in \bigcup_{j \in S_k} R_j, \end{aligned} \tag{8.55}$$

where either $R_j = \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}$ or $R_j = \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}$, depending on whether the THEN-condition of edit j is an inequality or an equality. That is, the THEN-condition of this implied edit consists of $|S_k|$ elementary numerical conditions

given by $\{\mathbf{x} \mid a_{1j}x_1 + ... + a_{nj}x_n + b_j \geq 0\}$ or $\{\mathbf{x} \mid a_{1j}x_1 + ... + a_{nj}x_n + b_j = 0\}$ for $j \in S_k$. An edit of format (8.55) is satisfied if the IF-condition is not satisfied, or if the IF-condition is satisfied and at least one of the elementary numerical conditions is satisfied. Note that edit (8.55) is indeed an implied edit. It has to be satisfied by the variables that have not yet been treated.

Edits of format (8.55) can be handled by splitting the error localisation problem into several subproblems. In each subproblem only one elementary numerical condition of each edit is involved. That is, in each subproblem the edits are of format (8.1). In total $\prod_k |S_k|$ subproblems have to be solved. In a latter stage of the algorithm these subproblems may themselves be split into new subproblems. The best solutions to the final subproblems are the optimal solutions to the overall problem.

If one wants to treat categorical variables before all numerical ones have been treated, one may consider allowing the broader class of explicit edits given by (8.55) instead the class given by (8.1). Edits of format (8.55) arise any way if categorical variables are treated before all numerical variables have been treated, even if the edits specified by the subject-matter specialists are of format (8.1).

It is easy to see that Theorem 8.1 is still valid if categorical variables are allowed to be treated before all numerical variables have been treated. As a consequence, Theorems 8.2 and 8.3 are also valid for this case, and we obtain the following corollary to the theorems.

Corollary to Theorems 8.1, 8.2 and 8.3. The algorithm presented in this section, where categorical variables are allowed to be treated before all numerical variables have been treated, determines all optimal solutions to the error localisation problem.

8.6. Computational aspects of the algorithm

We have demonstrated in Section 8.4 that the developed algorithm, described in Section 8.2, determines all optimal solutions to the error localisation problem for mixed data. At first sight, however, the developed algorithm may seem rather slow because an extremely large binary tree has to be generated to find all optimal solutions, even for moderately sized problems. Fortunately, the situation is not nearly as bad as it may seem.

8.6.1. Reducing the size of the tree

First of all, if the minimum number of fields that have to be changed in order to make a record pass all edits is (too) large, we feel that the record should not be edited automatically. In our opinion, the quality of such a record is simply too low to allow for automatic correction. We suggest that such a record should either be edited manually, or be discarded completely. By specifying an upper bound for the number of fields that may be changed, the size of the tree can drastically be reduced.

To illustrate the effect of the total number of variables $m + n$ and the upper bound $N_{\max}$ for the number of fields that may be changed on the maximum size of the generated tree,

we calculate the total number of nodes, both internal ones and terminal ones, in such a binary tree. Denote the number of nodes in a tree involving t variables where at most s variables may be eliminated by $F(s,t)$. This function satisfies the following recurrence relation:

$$F(s,t) = 1 + F(s-1,t-1) + F(s,t-1) \qquad \text{for } s,t \geq 1, \tag{8.56}$$

with boundary conditions

$$F(a,0) = 1 \tag{8.57}$$

and

$$F(0,b) = b+1. \tag{8.58}$$

After some puzzling we find that the solution to this recurrence relation and the boundary conditions is

$$F(s,t) = \left(2^{t+1} - 1\right) - \sum_{i=1}^{t-s} \binom{t-i}{s} \left(2^i - 1\right). \tag{8.59}$$

In our case we are interested in the value of $F(N_{\max}, m+n)$. Some numerical results to illustrate the behaviour of this function are given in Table 8.1 below.

Table 8.1. Total number of nodes in binary tree

	$N_{\max} = 1$	$N_{\max} = 2$	$N_{\max} = 5$	$N_{\max} = 10$	$N_{\max} = m + n$
$m + n = 10$	66	231	1,485	2,047	2,047
$m + n = 20$	231	1,561	82,159	1.40×10^{6}	2.10×10^{6}
$m + n = 50$	1,326	22,151	2.06×10^{6}	6.42×10^{10}	2.25×10^{15}
$m + n = 100$	5,151	171,801	1.35×10^{9}	1.80×10^{14}	2.54×10^{30}

Note that for a large tree a very substantial part of the tree may be cut off by specifying an upper bound for the number of fields that may be changed.

The size of the tree can also be reduced during the execution of the algorithm, because it may already become clear in an intermediate node of the tree that the terminal nodes in the corresponding branch cannot generate an optimal solution to the problem. For instance, by fixing the wrong variables we may make the set of edits infeasible. This may be noticed in an intermediate node.

8.6.2. Using the value of the objective function as an incumbent

The value of the objective function can also be used as an incumbent in order to reduce the size of the tree. This value cannot decrease while going down the tree. So, if the value of the objective function exceeds the value of an already found (possibly suboptimal) solution, we can again conclude that the terminal nodes in the corresponding branch cannot generate an optimal solution to the problem. In other words, the value of the best already found solution is used as the bound in our branch-and-bound scheme. During the execution of the algorithm the bound is updated.

In the present version of Leo, a prototype program implementing the algorithm of Section 8.2 that has originally been written Quere and later been adapted by Van den Broeke and De Waal, we simply compare the current value of the objective function in a node with the current bound. However, in a future version of Leo we plan to compute in each node a lower bound on the final value of the objective function, and compare that lower bound to the current bound. For each node such a lower bound on the final value of the objective function can be computed by adding the current value of the objective function to a lower bound on the sum of reliability weights of the other variables that need to be changed. A simple algorithm for determining a lower bound on the sum of reliability weights of the remaining variables that need to be changed in a certain node can be based on splitting the set of failed edits in this node into a maximum number of mutually disjoint subsets. Here, two sets of edits are said to be disjoint if and only if the set of variables involved in these edits are disjoint. In each subset of failed edits at least one variable needs to be changed in order to make these edits satisfied. A lower bound on the sum of reliability weights of the remaining variables that need to be changed is hence given by the sum of the minima of the reliability weights per subset. Splitting a set of failed edits into a maximum number of mutually disjoint edits is similar to determining the strongly connected subgraphs of a given graph (see Korte and Vygen, 2000, for an algorithm to determine strongly connected subgraphs). Such an algorithm can also be used to check before application of the branch-and-bound algorithm whether a record will require (too) many changes.

8.6.3. The number of edits due to elimination of a continuous variable

One might suspect that the number of implied edits grows rapidly. Fortunately, this is not the case in most practical situations. In fact, it is quite simple to calculate an upper bound on the number of edits after elimination of a continuous variable. Suppose our data set contains only continuous data. Let the total number of current edits be given by t. Suppose the variable to be eliminated occurs in r current edits. Suppose also that s of those r edits are inequalities, and that u of those r edits are equalities. Obviously, $s + u = r$.

We start by copying all t-r current edits not involving the variable under consideration to the new set of edits. If the equality-elimination rule is not used, each equality can be used in combination with any of the other r-1 edits involving the variable under consideration to eliminate this variable. This yields

$$\binom{u}{2} + us \tag{8.60}$$

new edits.

If the equality-elimination rule is used, one equality involving the variable under consideration is combined with any of the other r-1 edits involving this variable in order to eliminate this variable. This yields r-1 new edits.

The number of new edits due to combining inequalities is given by pq, where p is the number of inequalities that can be written as an upper bound on the value of the variable under consideration and q the number of inequalities that can be written as a lower bound. If s is even, the worst case is given by $p=q=s/2$. If s is odd, the worst case is given by $p=(s+1)/2$ and $q=(s-1)/2$, or vice versa. The maximum number of new edits due to elimination of the variable under consideration by combining inequalities only is hence given by:

- $s^2/4$ if s is even; (8.61)
- $(s+1)(s-1)/4$ if s is odd. (8.62)

If the equality-elimination rule is not used, an upper bound on the number of edits after elimination of a continuous variable is hence given by

- $t-(u+s)+u(u-1)/2+us+s^2/4$ if s is even; (8.63)
- $t-(u+s)+u(u-1)/2+us+(s+1)(s-1)/4$ if s is odd. (8.64)

In other words, if the equality-elimination rule is not used the expected increase in the number of edits due to elimination is given by

- $u(u-1)/2+us+s^2/4-u-s$ if s is even; (8.65)
- $u(u-1)/2+us+(s+1)(s-1)/4-u-s$ if s is odd. (8.66)

Table 8.2 below shows the maximum increase in the number of edits due to elimination of a continuous variable for several values of s and u if the equality-elimination rule is not used.

Table 8.2. The maximum increase in number of edits due to elimination of a continuous variable.

	u=0	u=1	u=2	u=3	u=4	u=5	u=10
s=0	0	-1	-1	0	2	5	35
s=1	-1	-1	0	2	5	9	44
s=2	-1	0	2	5	9	14	54
s=3	-1	1	4	8	13	19	64
s=4	0	3	7	12	18	25	75
s=5	1	5	10	16	23	31	86
s=6	3	8	14	21	29	38	98
s=7	5	11	18	26	35	45	110
s=8	8	15	23	32	42	53	123
s=9	11	19	28	38	49	61	136
s=10	15	24	34	45	57	70	150
s=20	80	99	119	140	162	185	315
s=50	575	624	674	725	777	830	1,110
s=100	2,400	2,499	2,599	2,700	2,802	2,905	3,435

If the equality-elimination rule is used, the increase in the number of edits due to elimination of a continuous variable is given by -1 if $u > 0$. If this rule is applied and $u = 0$, the maximum increase for various values of s is given by the column $u = 0$.

For high values of s and u, the maximum number of edits due to elimination of a continuous variable grows quickly. In most practical applications, however, the number of times that a variable is involved in the edits is rather low on the average. It is quite exceptional for a variable to be involved in six or more edits. This means that the number of edits hardly grows for most practical applications, and often does not grow at all. This is especially true if the equality-elimination rule is applied.

Assuming the natural probability model that, given that the variable to be eliminated is involved in a certain inequality, it is equally likely that this inequality provides an upper bound on the variable's value as it is likely that the inequality provides a lower bound, we can also calculate the expected number of new edits due to elimination of a continuous variable. Using the same notation as above, this expected number of new edits by combining inequalities only is given by

$$\mathrm{E}(\text{new edits}) = (1/2)^s \sum_{i=0}^{s} \binom{s}{i} i(s-i) \,. \tag{8.67}$$

Table 8.3 below shows the expected increase in the number of edits due to elimination of a continuous variable if the equality-elimination rule is not used for the same values of s and u as in Table 8.2.

Table 8.3. The expected increase in number of edits due to elimination of a continuous variable.

	u=0	u=1	u=2	u=3	u=4	u=5	u=10
s=0	0	-1	-1	0	2	5	35
s=1	-1	-1	0	2	5	9	44
s=2	-1.5	-0.5	1.5	4.5	8.5	13.5	53.5
s=3	-1.5	0.5	3.5	7.5	12.5	18.5	63.5
s=4	-1	2	6	11	17	24	74
s=5	0	4	9	15	22	30	85
s=6	1.5	6.5	12.5	19.5	27.5	36.5	96.5
s=7	3.5	9.5	16.5	24.5	33.5	43.5	108.5
s=8	6	13	21	30	40	51	121
s=9	9	17	26	36	47	59	134
s=10	12.5	21.5	31.5	42.5	54.5	67.5	147.5
s=20	75	94	114	135	157	180	310
s=50	562.5	611.5	661.5	712.5	764.5	817.5	1,097.5
s=100	2,375	2,474	2,574	2,675	2,777	2,880	3,410

Again, if the equality-elimination rule is used, the increase in the number of edits due to elimination of a continuous variable is given by –1 if $u > 0$. If this rule is applied and $u = 0$, the maximum increase for various values of s is given by the column $u = 0$.

The values in Table 8.3 are quite close to the values in Table 8.2, demonstrating that the "average", i.e. expected, case is quite close to the worst case. However, as we already remarked when discussing Table 8.2, in practice variables are unlikely to occur in many edits. Table 8.3 shows that, if the equality-elimination rule is applied, as long as variables on the average occur in at most 5 inequalities, the number of edits can on the average be expected to decrease while eliminating continuous variables.

8.6.4. Using good branching rules

Because the size of the tree, and hence the computing time of the algorithm, can be influenced by the order in which the variables are treated, this order is very important in practice. For purely categorical data Daalmans (2000) has tested several orders in which to treat the variables, namely:

a) select the variable that has not yet been eliminated and that appears first;

b) select a variable that has not yet been eliminated and that is involved in the largest number of failed edits;

c) select a variable that has not yet been eliminated and that is involved in the smallest number of failed edits;

d) select a variable that has not yet been eliminated, that is involved in the largest number of satisfied edits and is involved in at least one failed edit;

e) select a variable that has not yet been eliminated, that is involved in the smallest number of satisfied edits and is involved in at least one failed edit;

For a slightly different version of the branch-and-bound algorithm Daalmans (2000) has also tested the following orders:

f) select the failed edit that appears first in the set of edits, and select the variable that has not yet been eliminated and appears first in the selected failed edit;

g) select a failed edit that involves the largest number of variables, and select the variable that has not yet been eliminated and appears first in the selected failed edit;

h) select a failed edit that involves the smallest number of variables, and select the variable that has not yet been eliminated and appears first in the selected failed edit;

i) select a failed edit that involves the smallest number of variables, and select a variable that has not yet been eliminated and is involved in the largest number of failed edits;

j) select a failed edit that involves the smallest number of variables, and select a variable that has not yet been eliminated and is involved in the smallest number of failed edits;

k) select a failed edit that involves the smallest number of variables, and select a variable that has not yet been eliminated and is involved in the largest number of satisfied edits;

l) select a failed edit that involves the smallest number of variables, and select a variable that has not yet been eliminated and is involved in the smallest number of satisfied edits;

m) select a failed edit that involves the smallest number of variables, and select a variable that has not yet been eliminated and is involved in the largest number of edits (either satisfied or failed).

In Daalmans' implementation of the algorithm variables are first eliminated and then fixed to their original values. In Leo the opposite order has been implemented, i.e. variables are first fixed to their original values and then eliminated.

Daalmans has used six data sets to test his algorithm and related computer program for categorical data. The data sets were generated synthetically. They differ from each other with respect to the number of records, the number of variables, the reliability weights of the variables and the percentage of fields involved in the individual edits. Below we have listed some characteristics of the data sets.

- Data set A contains 96 fields involved in 11 edits. The individual edits involve 25% to 75% of the fields. Furthermore, data set A contains 10 records of which 1 is consistent. All reliability weights are equal to 1.
- Data set B differs from data set A with respect to the edits only. The edit set of data set B contains 6 edits. The individual edits involve 12.5% to 37.5% of the fields.
- Data set C contains 48 fields, all with a reliability weight of 1. These fields are involved in 11 edits. The individual edits involve 25% to 75% of the fields. There are 10 erroneous records. Each record has been replicated 10 times. So, in total the data set contains 100 records.
- Data set D differs from data set C with respect to the edits only. The edit set of data set D also contains 11 edits, but the individual edits involve only 12.5% to 37.5% of the fields.
- Data set E contains 24 fields, 11 edits and 12 records of which 1 is consistent. The individual edits involve 25% to 75% of the fields. The reliability weights of all variables are equal to 1.
- Data set F has been constructed by the Bureau of the Census. It contains 10 fields, involved in 27 edits. Each edit only involves two fields. Furthermore, the data contains 10 records of which 4 are consistent. Contrary to the first five data sets not all reliability weights are equal.

The results of Daalmans are listed in the tables below. In those tables run time is measured in seconds. Those results were obtained from simulations on a Pentium running at 500 MHz and with 127.0 MB of RAM. In the tables the column ‘Nodes’ refers to the number of nodes in the tree, so it measures the size of the tree. The third column ‘Implied’ refers to the number of implied edits that have been generated during execution of the program.

Table 8.4. Results for data set A

Version	Run time	Nodes	Implied
a	11.54	407770	2968
b	7.91	66912	2358
c	24.33	444332	7974
d	7.74	75225	1450
e	29.88	474629	7561
f	7.91	56368	2238
g	8.57	148977	1898
h	7.58	55240	2569
i	7.80	47035	3134
j	12.73	69173	6504
k	9.06	57013	4513
l	13.08	63077	4146
m	8.95	54316	4553

Table 8.5. Results for data set B

Version	Run time	Nodes	Implied
a	9.39	715818	580
b	3.68	98899	701
c	5.93	218383	1837
d	4.29	122202	229
e	6.34	263982	1138
f	3.68	98717	1457
g	3.52	102031	2237
h	3.68	91424	1346
i	3.46	74126	1185
j	4.89	126500	1578
k	3.46	85533	992
l	7.53	119641	2257
m	3.47	79215	1054

Table 8.6. Results for data set C

Version	Run time	Nodes	Implied
a	27.19	1328640	18160
b	34.44	518770	35110
c	67.78	1512440	67710
d	26.86	366030	25070
e	77.89	1348160	83320
f	27.46	317210	37470
g	26.86	510190	36630
h	28.16	306010	35410
i	34.98	450220	38110
j	48.23	345160	74530
k	31.08	308710	42580
l	70.03	546500	82830
m	37.13	457990	40010

Table 8.7. Results for data set D

Version	Run time	Nodes	Implied
a	58.05	3262140	25200
b	27.62	507310	30370
c	57.94	1107170	62070
d	22.03	501610	15290
e	52.82	1092450	41220
f	17.91	273230	22270
g	27.84	500150	35810
h	18.13	241410	23100
i	21.64	271640	29950
j	20.05	294320	21670
k	13.73	212320	21660
l	26.68	322040	30850
m	19.43	266340	24550

Table 8.8. Results for data set E

Version	Run time	Nodes	Implied
a	0.49	19568	966
b	0.55	7838	1539
c	0.88	18877	2167
d	0.44	7181	1149
e	1.04	8146	2786
f	0.49	5693	1441
g	0.44	8765	1481
h	0.44	5487	1400
i	0.55	6188	1474
j	0.60	6008	2068
k	0.49	5523	1604
l	0.82	7376	2274
m	0.55	6333	1571

Table 8.9. Results for data set F

Version	Run time	Nodes	Implied
a	0.05	317	48
b	0.00	141	45
c	0.05	208	69
d	0.05	152	50
e	0.06	64	79
f	0.00	48	43
g	0.00	48	43
h	0.00	48	43
i	0.00	45	42
j	0.00	51	47
k	0.00	47	43
l	0.06	47	43
m	0.00	45	42

The number of solutions to the error localisation problem does not depend on the version of the program used. The number of optimal solutions found by the program can be huge for some records. For example, in data set A we have one record with 15,200 optimal solutions.

When we consider the columns ‘Run time’ we observe that most versions require less than 1 second to process one record on the average for each data set. This confirms that a branch and bound method seems to be suitable for practical usage

From the results of the performed tests we cannot conclude that the run time is mainly affected by either the size of the tree or the number of implied edits alone. It seems to be the combination of the size of the tree and the number of implied edits that determines the run time. Besides, there are more factors than just the size of the tree and the number of implied edits that influence the run time. Consider, for example, data set C and compare version g with version h. Although the number of implied edits is lower and the size of the tree is smaller when version h is implemented, version g is faster.

None of the implemented versions has the best performance, i.e. smallest run time, for each of the tested data sets. However, some of the versions are faster than other versions for all tested data sets. For example, versions b, d and k are respectively faster than versions c, e and l. Although the “best” overall order could not be identified from Daalmans’ work, the results suggest that order d, or order k for the slightly modified version of the branch-and-bound algorithm, is a good order in practice.

If we are dealing with a mix of categorical and continuous data, the order in which the variables are treated not only influences the computation time of the algorithm but also its complexity. If we allow categorical variables to be treated before all numerical variables have been treated, the complexity of the algorithm increases, as is discussed in Section 8.5, because the problem has to be split into several subproblems.

8.7. Discussion

In this chapter we have presented an algorithm for solving the error localisation problem in categorical and continuous data that appears to be very promising. There are several reasons why we consider the algorithm so promising.

First, the algorithm proposed in this chapter is quite simple. It is (much) simpler to understand and implement than the algorithms presented in previous chapters. One of the reasons for the simplicity of the algorithm is that it is a very "natural" one. For instance, in the algorithm categorical and continuous variables are treated in almost the same manner, only the underlying method to generate implicit edits differs. Moreover, searching for optimal solutions to the error localisation problem is also a natural process. All possible solutions are simply checked, and the best solutions found are the optimal ones. Because of the simplicity of the branch-and-bound algorithm discussed in this chapter, maintaining software based on this algorithm is (much) simpler than maintaining software based on the algorithms discussed in previous chapters. Not only Operations Research specialists can understand the algorithm in detail, but also the IT-specialists who develop and maintain the final computer program based on the mathematical algorithm.

Second, the algorithm can be extended to solve the error localisation problem in categorical, continuous and *integer* data. This is discussed in detail in the next chapter. Handling integer data properly with any of the algorithms presented in previous chapters would either be impossible or would lead to extremely complicated algorithms.

Third, the algorithm can, without too much trouble, also be used for other purposes. For instance, the algorithm can be used to test whether a set of edits is consistent or not, i.e. whether these edits contradict each other or not. The algorithm can also be used to detect redundant edits in a given set of edits. These two topics are discussed in Chapter 13. Moreover, a slightly modified version of the algorithm can be applied to ensure that imputed data satisfy all edits. This problem of ensuring that imputed data satisfy all edits, the so-called consistent imputation problem, and a heuristic for solving it are discussed in Chapter 12.

9. The Error Localisation Problem for Integer Data

9.1. Introduction

In this chapter we consider the error localisation problem for a mix of categorical, continuous and integer data. In Section 9.2 we sketch the problem by means of a simple example. Section 9.3 provides a mathematical formulation for the error localisation problem for a mix of categorical, continuous and integer data. In Section 9.5 we describe an extension of the branch-and-bound algorithm presented in Chapter 8 that solves the error localisation problem for a mix of categorical, continuous and integer data to optimality. Essential in this extended algorithm is Fourier-Motzkin elimination for integer data, which we describe in Section 9.4. We conclude the chapter with a brief discussion in Section 9.6.

The method to apply Fourier-Motzkin elimination on integer data discussed in Section 9.4 is due to Pugh (see Pugh, 1992; Pugh and Wonnacott, 1994). Pugh applied this technique to develop so-called array data dependence testing algorithms. Such algorithms are used to detect ordering constraints among references to an array. These ordering constraints are used in parallelising and vectorising computer compilers. In other words, array data dependence testing algorithms are used to optimise computer compilers. Our algorithm for solving the error localisation problem for a mix of categorical, continuous and integer data efficiently combines Fourier-Motzkin elimination in integer data with the algorithm of Chapter 8. The application of Pugh's method for Fourier-Motzkin elimination in integer data to the error localisation problem, see Section 9.5, was first described in De Waal (2001a).

9.2. The problem for integer data

In this section we sketch the difference between the error localisation problem for a mix of categorical and continuous data and the error localisation problem for a mix of categorical, continuous and integer data. We also sketch the idea of the solution method, which basically consists of checking whether all integer-valued variables involved in a solution to the continuous error localisation problem, i.e. the error localisation problem where we temporarily assume all numerical variables to be continuous, can indeed attain integer values.

Suppose a set of explicit edits is given by

$$T = P + C, \tag{9.1}$$

$$0.5 \leq \frac{C}{T} \leq 1.1, \tag{9.2}$$

$$0 \leq \frac{T}{N} \leq 550, \tag{9.3}$$

$$T \geq 0\,, \tag{9.4}$$

$$C \geq 0\,, \tag{9.5}$$

$$N \geq 0\,, \tag{9.6}$$

and

$$N \leq \frac{C}{320}\,. \tag{9.7}$$

where T denotes the turnover of an enterprise, P its profit, C its costs, and N the number of employees. The turnover, profit and costs are continuous variables, the number of employees an integer one. The turnover, profit and costs are given in thousands of Euros. By multiplying (9.2) by T and (9.3) by N both edits can be written as two linear inequality edits each.

Let us consider a specific record with values T = 5,060, P = 2,020, C = 3,040 and N = 5. This record fails both (9.3) and (9.7). We assume that the reliability weights for T, P and C are equal to 1, and the reliability weight of N to 2.

If N were continuous, the only optimal solution to the above error localisation problem would be: change the value of N. However, N is an integer-valued variable. So, we need to check whether a feasible *integer* value for N exists. By filling in the values for T, P, and C in (9.3) and (9.7) we find $9.2 \leq N \leq 9.5$. In other words, a feasible integer value for N does not exist. Changing the value of N is hence not a solution to this error localisation problem.

There is one next best solution to the continuous error localisation problem. This next best solution is given by: change the values of T, P and C. This is obviously also a feasible solution to the problem at hand, as variable N retains its original value (5), which is integer. It is the optimal solution to our problem as this is the best solution to the continuous error localisation problem where all integer-valued variables can indeed attain integer values.

9.3. The error localisation problem for categorical, continuous and integer data

The error localisation problem for a mix of categorical, continuous and integer data can be formulated as follows:

Minimise

$$\sum_{i=1}^{m} w_i \delta(v_i^0, v_i) + \sum_{i=1}^{n} w_{m+i} \delta(x_i^0, x_i) \tag{9.8}$$

so that all edits j=1,…,J given by either

$$\begin{aligned} &\text{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}\,, \end{aligned} \tag{9.9a}$$

or

$$\begin{aligned} &\text{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}\,, \end{aligned} \qquad (9.9b)$$

are satisfied, where x_i is integer for $i \in I$, and the remaining x_i are continuous.

Here I denotes the index set of the integer variables. The a_{ij} in (9.9) are assumed to be rational numbers. By multiplying the coefficients a_{ij} involved in the statement after the THEN-condition of (9.9) by an appropriately chosen integer we can ensure that in each THEN-condition these coefficients become integral and that their greatest common divisor equals 1.

All edits given by (9.9) and all integrality constraints have to be satisfied simultaneously. We assume that the edits and integrality constraints can indeed be satisfied simultaneously. We also assume that none of the values of the variables entering the edits may be missing and that any non-integral value for an integer-valued variable is considered erroneous.

Our aim is to find all optimal solutions to this error localisation problem. Later, one of these optimal solutions is selected, using a secondary criterion. The variables involved in the selected solution are set to missing.

9.4. Fourier-Motzkin elimination in integer data

An important technique used in the algorithm described in Chapter 8 is Fourier-Motzkin elimination for eliminating a continuous variable from a set of linear equalities and inequalities. Fourier-Motzkin elimination can be extended to integer data in several ways. For example, Dantzig and Eaves (1973) and Williams (1976 and 1983) describe extensions of Fourier-Motzkin elimination to integer programming problems. Unfortunately, these methods appear to be rather inefficient, i.e. they are too time-consuming to be applied in many practical cases.

Pugh (1992) proposes an alternative extension of Fourier-Motzkin elimination to integer-valued data. He refers to this extension as the Omega test, and we will follow his terminology. Below we briefly explain the Omega test. For more details on Fourier-Motzkin elimination in integer data and the Omega test we refer to Pugh (1992), Pugh and Wonnacott (1994), and Van den Broeke (2000). This section is for a substantial part based on those papers, especially the last one.

The Omega test has been designed to determine whether an integer-valued solution to a given set of linear equalities and inequalities exists. Suppose the linear inequalities and equalities are given by

$$a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0 \tag{9.10a}$$

and

$$a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\,. \tag{9.10b}$$

To simplify our notation we define $x_0 = 1$ and $a_{0j} = b_j$, and re-write the linear inequalities and equalities as

$$a_{0j}x_0 + a_{1j}x_1 + \ldots + a_{nj}x_n \geq 0\,, \tag{9.11a}$$

respectively as

$$a_{0j}x_0 + a_{1j}x_1 + \ldots + a_{nj}x_n = 0\,. \tag{9.11b}$$

Like we mentioned in Section 9.3 we can assume – without loss of generality – that all equalities and inequalities are normalised, i.e. that all a_{ij} are integer and the greatest common divisor of the a_{ij} in each constraint *j* equals 1. All variables x_i are integer-valued in this section.

We start by "eliminating" all equality constraints until we arrive at a new problem involving only inequality constraints. In this context, we say that all equalities have been eliminated once we have transformed the original system of (in)equalities (9.11) into an equivalent system of (in)equalities of the following type:

$$x'_k = \sum_{i>k} a'_{ij_k} x'_i \qquad \text{for } k=0,\ldots,s\text{-}1, \tag{9.12a}$$

$$x'_k \geq \sum_{i\geq s, i\neq k} a'_{ij_k} x'_i \qquad \text{for } k=s,\ldots,J'\,, \tag{9.12b}$$

where the a'_{ij} are integer. The x'_i are a permutation of the x_i, possibly supplemented by some additional variables (see Subsection 9.4.1). The first s x'_i, which are only involved in equalities, are expressed in terms of the remaining variables, which may also be involved in inequalities. Due to the possible introduction of additional variables, the system (9.12) may have more equalities than the original system (9.11). The original system (9.11) has an integer-valued solution if and only if the system (9.12b) has an integer-valued solution. Namely, an integer solution for the x'_i ($k \geq s$) yields an integer solution for the x'_i ($k < s$) by back-substitution. In other words, to check whether a system (9.11) has an integer-valued solution, we only need to check whether the inequalities (9.12b) of the equivalent system (9.12) have an integer-valued solution. In this sense, the equalities of (9.11) have been eliminated once we have transformed a system given by (9.11) into an equivalent system given by (9.12). How equalities involving only integer variables can be eliminated is explained in the next subsection.

9.4.1. Eliminating equalities

We define the following operation involving two integers *a* and *b*:

$$a \overline{\bmod}\ b = a - b\lfloor a/b + 1/2 \rfloor, \tag{9.13}$$

where $\lfloor u \rfloor$ denotes the largest integer less than or equal to u. If b is odd, the value of $a \overline{\bmod} b$ lies in $[-(b-1)/2, (b-1)/2]$. If b is even, the value of $a \overline{\bmod} b$ lies in $[-b/2, b/2-1]$. If $\lfloor a/b \rfloor < 1/2$, then $a \overline{\bmod} b = a \bmod b$. If $\lfloor a/b \rfloor \geq 1/2$, $a \overline{\bmod} b = -a \bmod b$. Here, the mod b operator assumes values in [0, b-1].

To eliminate an equality given by

$$\sum_{i=0}^{n} a_{is} x_i = 0, \tag{9.14}$$

we do the following.

We first check whether there is a $j \neq 0$ such that $|a_{js}| = 1$. If this is the case, we eliminate the constraint by using this constraint to express x_j in the other variables. This expression for x_j is then substituted into the other constraints.

If such a j does not exist, we select a variable k for which the coefficient a_{ks} has the lowest absolute value in this equality. We define $m = |a_{ks}| + 1$.

Now we introduce a new variable σ defined by the following relation:

$$m\sigma = \sum_{i=0}^{n} (a_{is} \overline{\bmod} m)\, x_i \tag{9.15}$$

This variable σ is integer-valued. This can be shown as follows.

$$\sum_{i=0}^{n} (a_{is} \overline{\bmod} m)\, x_i = \sum_{i=0}^{n} \left(a_{is} - m\lfloor a_{is}/m + 1/2 \rfloor\right) x_i =$$

$$\sum_{i=0}^{n} a_{is} x_i - \sum_{i=0}^{n} m\lfloor a_{is}/m + 1/2 \rfloor x_i = -\sum_{i=0}^{n} m\lfloor a_{is}/m + 1/2 \rfloor x_i. \tag{9.16}$$

So, σ equals $-\sum_{i=1}^{n} \lfloor a_{is}/m + 1/2 \rfloor x_i$, which is integer because the x_i and their coefficients in (9.16) are integer.

It is easy to see that $a_{ks} \overline{\bmod} m = -\text{sign}(a_{ks})$. Now we use constraint (9.15) to express x_k in terms of the other variables.

$$x_k = -\text{sign}(a_{ks}) m\sigma + \sum_{i=0, i\neq k}^{n} \text{sign}(a_{ks})(a_{is} \overline{\bmod} m)\, x_i \tag{9.17}$$

Substituting (9.17) into the original equality (9.14) gives

$$-|a_k|m\sigma + \sum_{i=0,i\neq k}^{n}(a_{is} + |a_{ks}|(a_{is} \overline{\bmod} m))\, x_i = 0\,. \tag{9.18}$$

Because $|a_{ks}| = m-1$, (9.18) can be written as

$$-|a_{ks}|m\sigma + \sum_{i=0,i\neq k}^{n}(a_{is} - (a_{is} \overline{\bmod} m) + m(a_{is} \overline{\bmod} m))\, x_i = 0 \tag{9.19}$$

Dividing (9.19) by *m* and using (9.13) gives

$$-|a_{ks}|\sigma + \sum_{i=0,i\neq k}^{n}\left(\lfloor a_{is}/m + 1/2\rfloor + (a_{is} \overline{\bmod} m)\right)x_i = 0\,. \tag{9.20}$$

In (9.20) all coefficients are integer-valued.

It is clear that if the coefficient of variable *i* equals zero in (9.14) the corresponding coefficient in (9.20) also equals zero. It is also clear that the absolute value of the coefficient of σ in (9.20) is equal to the coefficient of x_k in (9.14). However, for all other variables with a non-zero coefficient in (9.14) the absolute value of the coefficient in (9.20) is smaller than the absolute value of the corresponding coefficient in (9.14). To prove this statement we first re-write the value of the coefficient of x_i ($i \neq k$) in (9.20) in the following way:

$$\lfloor a_{is}/m + 1/2\rfloor + (a_{is} \overline{\bmod} m) = \lfloor a_{is}/m + 1/2\rfloor + a_{is} - m\lfloor a_{is}/m + 1/2\rfloor =$$

$$-|a_{ks}|\left\lfloor \frac{a_{is}}{|a_{ks}|+1} + \frac{1}{2}\right\rfloor + a_{is} \equiv \hat{a}_{is}\,. \tag{9.21}$$

We now consider the cases that a_{is} is positive and negative separately. If $a_{is} > 0$, then $a_{is} \geq |a_{ks}|$. Suppose $a_{is} = \lambda|a_{ks}|$, where $\lambda \geq 1$. We then have

$$\hat{a}_{is} = a_{is}\left(-\frac{1}{\lambda}\left\lfloor \frac{\lambda|a_{ks}|}{|a_{ks}|+1} + \frac{1}{2}\right\rfloor + 1\right). \tag{9.22}$$

Using

$$1 \leq \left\lfloor \frac{\lambda|a_{ks}|}{|a_{ks}|+1} + \frac{1}{2}\right\rfloor \leq \lambda\,, \tag{9.23}$$

we obtain $0 \leq \hat{a}_{is} \leq (1-\frac{1}{\lambda})a_{is}$. Hence, we can conclude that $|\hat{a}_{is}| < |a_{is}|$.

Similarly, we can show that if $a_{is} < 0$, then too $|\hat{a}_{is}| < |a_{is}|$. This is left for the reader to verify (alternatively, the reader is referred to Van den Broeke, 2000).

After repeated application of the above substitution rule, where each time a new variable is introduced and an old variable is eliminated, to the original equality (9.14) and its derived form (9.20) the equality is transformed into an equality in which (at least) one of the coefficients has absolute value 1. The corresponding variable can then be used to eliminate the equality.

This process continues until we have eliminated all equalities and we are left only with inequalities. In the next subsection we explain how integer variables can be eliminated from a set of linear inequalities, but first we give an example of how equalities are eliminated.

Example 9.1:

Pugh (1992) illustrates his method to eliminate equalities by means of an example. In this example, six constraints have been specified:

$$7x + 12y + 31z = 17, \tag{9.24}$$

$$3x + 5y + 14z = 7, \tag{9.25}$$

$$1 \le x \le 40, \tag{9.26}$$

$$-50 \le y \le 50. \tag{9.27}$$

Note that (9.26) and (9.27) each stand for two constraints. We begin by eliminating equality (9.24). Note that m = 8, and using (9.15) we start by introducing a variable σ defined by

$$8\sigma = -x + 4y + z + 1. \tag{9.28}$$

We eliminate x from (9.24) to (9.26) by substituting this expression into these constraints. This yields

$$-7\sigma - 2y + 3z = 3, \tag{9.29}$$

$$-24\sigma - 7y + 11z = 10, \tag{9.30}$$

$$1 \le -8\sigma - 4y - z - 1 \le 40. \tag{9.31}$$

The set of constraints (9.24) to (9.27) is hence equivalent to (9.27) to (9.31).

Equation (9.29) has been obtained from equation (9.24). We continue with the elimination of this equation. We now introduce a variable τ, defined by

$$3\tau = y - \sigma. \tag{9.32}$$

We eliminate y from (9.27), and (9.29) to (9.31) by substituting (9.32) into these constraints. This yields

$$-3\sigma - 2\tau + z = 1, \tag{9.33}$$

$$-31\sigma - 21\tau + 11z = 10, \tag{9.34}$$

$$1 \le -12\sigma - 12\tau - z - 1 \le 40\,, \tag{9.35}$$

$$-50 \le \sigma + 3\tau \le 50\,. \tag{9.36}$$

The original set of constraints is equivalent to (9.28), (9.32) and (9.33) to (9.36). Equation (9.33) has been obtained from (9.29), and hence indirectly from (9.24). We continue with the elimination of this equation. Variable z has coefficient 1 in this equation, so we eliminate z from (9.34) to (9.36). We obtain

$$2\sigma + \tau = -1\,, \tag{9.37}$$

$$1 \le -15\sigma - 14\tau - 2 \le 40\,, \tag{9.38}$$

and (9.36). The original set of constraints is equivalent to (9.28), (9.32), (9.33), (9.36), (9.37) and (9.38). Variable τ has coefficient 1 in (9.37). We use this equation to eliminate τ from (9.36) and (9.38). We obtain

$$1 \le 13\sigma + 12 \le 40\,, \tag{9.39}$$

$$-50 \le -5\sigma - 3 \le 50\,. \tag{9.40}$$

We can re-write (9.39) and (9.40) as

$$0 \le \sigma \le 2\,. \tag{9.41}$$

We are now done. The original set of constraints is equivalent to (9.28), (9.32), (9.33), (9.37) and (9.41). This latter set of constraints can be written in the desired format (9.12). ■

9.4.2. Eliminating an integer variable from a set of inequalities

When an integer variable is eliminated from a set of inequalities, two different sets of inequalities are determined. The first set is referred to as the *real shadow*. This is simply the set of inequalities that results if we apply the standard form of Fourier-Motzkin elimination. That is, the real shadow results if we treat the integer variable that is being eliminated as continuous.

The second set of resulting inequalities is referred to as the *dark shadow*. This dark shadow is constructed in such a way that if it contains a feasible (integer) solution, then there is an integer value for the eliminated variable such that all original inequalities become satisfied.

The construction of the dark shadow is quite simple. Suppose that two inequalities

$$ax \le \alpha \tag{9.42}$$

and

$$bx \ge \beta \tag{9.43}$$

are combined to eliminate the integer variable x. Here a and b are positive integer constants, and α and β are linear expressions that can involve all variables except x. Each variable involved in α or β is assumed to have an integer coefficient. The real shadow of (9.42) and (9.43) obviously is $a\beta \le b\alpha$.

Integer Data

Consider the case in which there is an integer value larger than or equal to $a\beta$ and smaller than or equal to $b\alpha$, but there is no integer solution for x to $a\beta \le abx \le b\alpha$. Let $i = \lfloor \beta / b \rfloor$, then

$$abi < a\beta \le b\alpha < ab(i+1) \tag{9.44}$$

We clearly have $a(i+1) - \alpha > 0$. Since the values of a, b, α and β are integer, we have $a(i+1) - \alpha \ge 1$, and hence

$$ab(i+1) - b\alpha \ge b\,. \tag{9.45}$$

Similarly, we obtain

$$a\beta - abi \ge a\,. \tag{9.46}$$

Combining (9.45), (9.46), we arrive at

$$b\alpha - a\beta \le ab - a - b\,. \tag{9.47}$$

In other words, if

$$b\alpha - a\beta \ge ab - a - b + 1 = (a-1)(b-1)\,, \tag{9.48}$$

then an integer solution for x necessarily exists.

To be able to satisfy (9.42) and (9.43) by an integer value for x it is sufficient that (9.48) holds true. We therefore define the dark shadow of (9.42) and (9.43) obtained by eliminating variable x by (9.48). Note that if (9.48) holds true, there is an integer value larger than or equal to $a\beta$ and smaller than or equal to $b\alpha$.

Note that if $a = 1$ or $b = 1$, the real shadow and the dark shadow are identical. If the real shadow and the dark shadow resulting from each combination of an upper bound and a lower bound for x are identical, we say that the elimination, or projection, is exact. If the projection is exact, an integer solution exists if and only if an integer solution to the real/dark shadow exists.

If the real shadow and the dark shadow are not identical, we have the following possibilities:

- If the dark shadow has an integer solution, the system (9.42) and (9.43) has an integer solution for x.
- If the real shadow does not contain a feasible (integer) solution, there is no integer solution for x to (9.42) and (9.43). Note that the real shadow contains a feasible integer solution if and only if it has a feasible solution, because (9.42) and (9.43) involve only integer-valued coefficients and variables.
- In all other cases, it is not yet clear whether an integer solution to (9.42) and (9.43) exists. We know that if in such a case an integer solution to (9.42) and (9.43) exists, a pair of constraints $ax \le \alpha$ and $\beta \le bx$ exists such that $ab - a - b \ge b\alpha - a\beta$ and $b\alpha \ge abx \ge a\beta$. From this we can conclude that in such a case an integer solution to (9.42) and (9.43) would satisfy $ab - a - b + a\beta \ge abx \ge a\beta$.

In the latter case we can check whether such a solution exists by basically examining all possibilities. That is, we determine the largest coefficient a of x for all upper bounds on x. For each lower bound $\beta \leq bx$ we then test whether an integer solution exists to the original constraints combined with $bx = \beta + i$ for each integer i satisfying $(ab - a - b)/a \geq i \geq 0$.

In other words, in the latter case we examine smaller subproblems of the original problem. These smaller subproblems are referred to as *splinters*. Testing all splinters for integer solutions can be quite time-consuming. Therefore, creating splinters and testing them for integer solutions should be avoided as much as possible.

If $n = 1$, i.e. if there is only variable x involved in the set of inequalities, the dark shadow and the real shadow involve only numbers and no unknowns. We then have the following possibilities:

- If the set of inequalities defining the dark shadow does not contain an inequality that is a contradiction, the system (9.42) and (9.43) has an integer solution for x.
- If the set of inequalities defining the real shadow contains a contradiction, the system (9.42) and (9.43) has no integer solution for x.
- In all other cases, we add the constraints $bx = \beta + i$ for $(ab - a - b)/a \geq i \geq 0$ (i integer) to the original set of constraints, and check whether there is an integer solution to the resulting set of constraints. Again, the result of such a check will be that either we obtain a set of inequalities not involving numbers that contains a contradiction (in which case this particular splinter does not contain an integer solution to the original set of constraints) or we obtain a set of inequalities not involving numbers that does not contain a contradiction (in which case this particular splinter does contain an integer solution to the original set of constraints). If none of the examined splinters contains an integer solution, there is no integer solution to the original set of constraints.

We illustrate the procedure by means of the simple example below.

Example 9.2:
We consider two inequalities involving two unknowns:

$$3x_1 + 4x_2 \geq 10 \tag{9.49}$$

and

$$2x_1 - 5x_2 \geq 2\,. \tag{9.50}$$

The real shadow obtained by eliminating x_2 from (9.49) and (9.50) is given by

$$23x_1 \geq 58\,. \tag{9.51}$$

The dark shadow obtained by eliminating x_2 from (9.49) and (9.50) is given by

$$23x_1 \geq 70\,. \tag{9.52}$$

There are three splinters, each defined by three constraints. Two of these constraints are the same for all three splinters. They are given by (9.49) and (9.50). The third constraint differs for each splinter. This constraint is given by

$$4x_2 = 10 - 3x_1 + i\,, \tag{9.53}$$

for $i = 0, 1, 2$. ■

It is clear that if the dark shadow or one of the splinters has an integer solution, then also the original set of equalities and inequalities has an integer solution. On the other hand, because we basically examine all possibilities, it also holds true that if the original set of equalities and inequalities has an integer solution then also the dark shadow or one of the splinters has an integer solution. These two solutions are essentially the same, except for some transformations, and the possible introduction of some auxiliary variables (if a splinter has been eliminated).

So, we have the following theorem.

Theorem 9.1. If and only if an integer solution to the dark shadow (involving only n-1 variables) or one of the splinters (involving only n-1 integer variables after the added equalities have been eliminated) exists, then an integer solution to the original set of equalities and inequalities (involving n integer variables) exists.

This is a lifting principle. A property involving only n-1 variables is lifted to the corresponding property for n variables.

We have now explained how we can check whether a feasible integer value exists for an integer variable involved in a set of linear inequalities by eliminating this integer variable. In the next subsection we examine how we can check whether an integer solution exists for several variables simultaneously by eliminating these variables.

9.4.3. Eliminating several variables from a set of inequalities

Suppose we want to solve the problem whether an integer solution exists for a set of linear constraints involving n variables. We solve this problem by eliminating these n variables. During the elimination process the original problem may split into several subproblems due to the splinters that may arise. We apply the procedure sketched below.

We treat each subproblem separately. We first eliminate all variables by standard Fourier-Motzkin elimination, i.e. we repeatedly determine the real shadow until all variables have been eliminated. If the final real shadow is inconsistent, the subproblem does not have a continuous solution, let alone an integer solution. In such a case this subproblem can be discarded.

If the final real shadow of a subproblem is consistent, we examine the subproblem again and check whether there is an integer solution to this subproblem. For this subproblem we

iteratively select a variable from the variables that have not yet been eliminated. The selected variable will be eliminated. In order to keep the number of computations limited we choose the variable so that the elimination will be exact if possible. As a secondary aim we may then also minimise the number of constraints resulting from the combination of upper and lower bounds. If an exact elimination/projection is not possible, we select a variable with coefficients as close as possible to zero. For such a variable the number of splinters will be relatively small. For the subproblem under consideration, we determine the dark shadow and the splinters by eliminating the selected variable.

We continue this process until all n variables have been eliminated. The final "subproblems", or better: final sets of relations, only involve numbers and no unknowns. We have the following theorem.

Theorem 9.2. If any of the final sets of relations does not contain a contradiction, the original set of constraints has an integer solution. If all final sets of relations contain a contradiction, the original set of constraints does not have an integer solution.

Proof. This follows directly from a repeated application of Theorem 9.1. ∎

In more formal notation, the above procedure can be sketched as follows:

0. The list **L** of (sub)problems to solve contains only the original problem.
1. Select a subproblem s from **L**. If **L** is empty or if the selected subproblem contains no variables go to Step 4, else go to Step 2.
2. Temporarily eliminate all variables from the selected subproblem s by standard Fourier-Motzkin elimination, i.e. repeatedly determine the real shadow until all variables have been eliminated. If the final set of relations involving only numbers and no unknowns is inconsistent, subproblem s is deleted from **L**. In this case we go to Step 1, else we go to Step 3.
3. For the subproblem s selected in Step 1, select a variable x_r that has not yet been eliminated. Determine the dark shadow and splinters of subproblem s obtained by eliminating variable x_r. Eliminate the equality, and a corresponding unknown, from each splinter. Add the dark shadow and the splinters to **L**, and delete subproblem s from **L**. Go to Step 1.
4. If **L** is empty, the original problem does not have an integer solution and the procedure terminates. If **L** is not empty and the selected subproblem s does not contain any variables, check whether the corresponding set of relations involving only numbers and no unknowns is consistent or not. If the set is inconsistent, delete s from **L** and go to Step 1. If the set is consistent, an integer solution to the original problem exists and the procedure terminates.

Now we have described the Omega test we are ready to move on to the error localisation problem for categorical, continuous and integer data.

9.5. Error localisation in categorical, continuous and integer data

At first sight it may not be clear that the Omega test method can be integrated efficiently with the branch-and-bound approach proposed by Quere and De Waal (see Chapter 8, and De Waal and Quere, 2003) for solving the error localisation problem for categorical and continuous data. In this section we describe a simple way to integrate both methods.

Before we describe the algorithm, we first explain in Section 9.5.1. how equalities in THEN-conditions involving integer variables can be eliminated. Next, in Section 9.5.2 we explain how integer variables can be eliminated from inequalities in THEN-conditions. Finally, in Section 9.5.3 we describe our algorithm for solving the error localisation problem in categorical, continuous and integer data.

9.5.1. Error localisation: eliminating equalities involving integer variables

In our algorithm (see Section 9.5.3) integer variables are treated after all continuous variables have been treated and before any categorical variable is treated. That is, once the integer variables are treated all edits involve only categorical and integer variables.

If integer variables are involved in equalities in THEN-conditions of edits, we first eliminate these equalities. We select an equality, and apply the technique explained in Section 9.4.1 to arrive at an equality in which the coefficient of a variable equals 1. During this process the IF-condition of the edit under consideration does not alter. That is, if the selected edit is given by

$$\begin{aligned} &\text{IF} && v_i \in F_i^s \ \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} a_{is} x_i = 0\}, \end{aligned} \tag{9.54}$$

we transform this edit into an edit given by

$$\begin{aligned} &\text{IF} && v_i \in F_i^s \ \text{ for } i=1,\ldots,m \\ &\text{THEN} && (\widetilde{x}_1,\ldots,\widetilde{x}_n) \in \{\widetilde{\mathbf{x}} \mid \sum_{i=0}^{\widetilde{n}} \widetilde{a}_{is} \widetilde{x}_i = 0\}, \end{aligned} \tag{9.55}$$

where the $\widetilde{a}_{ij}$ are integers, and the $\widetilde{x}_i$ are the transformed integer variables, possibly supplemented by some additional variables due to the elimination of the equalities. In (9.55), $\widetilde{n}$ denotes the total number of variables $\widetilde{x}_i$, i.e. the number of transformed original variables plus the number of additional variables due to the elimination of the equalities. In (9.55) at least one integer variable, say $\widetilde{x}_r$ ($r > 0$), has a coefficient with absolute value

equal to one. The other edits are also written in terms of the $\tilde{x}_i$ by applying rule (9.17) to the numerical THEN-conditions. The IF-conditions of the other edits are not altered by this transformation process. Because auxiliary variables may need to be introduced during this transformation process, we may in fact need to introduce some auxiliary equations of type (9.55). In each of these auxiliary equations, the new auxiliary variable is expressed in terms of the other variables, i.e. the original variables and the already generated auxiliary variables (see (9.15) and (9.16)).

For notational convenience, we write the transformed coefficients $\tilde{a}_{ij}$ and transformed variables $\tilde{x}_i$ again as a_{ij} and x_i, respectively. It is important to keep in mind, though, that these coefficients and variables may differ from the original coefficients and variables.

Once we have a (transformed) coefficient a_{rs} ($r > 0$) such that $|a_{rs}| = 1$ in our selected edit, we use the THEN-condition of this edit to express the (transformed) variable x_r in terms of the other variables. That is, we use

$$x_r = -\text{sign}(a_{rs}) \sum_{i \neq r} a_{is} x_i \ . \tag{9.56}$$

This expression for x_r is then substituted into the THEN-conditions of the other edits as far as this is possible. The IF-conditions of these other edits are changed by the substitution process. In particular, due to this substitution process an edit given by

$$\begin{aligned} &\textsf{IF} \quad && v_i \in F_i^t \ \text{ for } i=1,\ldots,m \\ &\textsf{THEN} \quad && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} a_{it} x_i \geq 0\} \ , \end{aligned} \tag{9.57}$$

involving x_r in its THEN-condition leads to

$$\begin{aligned} &\textsf{IF} \quad && v_i \in F_i^t \cap F_i^s \quad \text{for } i=1,\ldots,m \\ &\textsf{THEN} \quad && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i \neq r} (a_{it} - \text{sign}(a_{rs}) a_{rt} a_{is}) x_i \geq 0\} \ , \end{aligned} \tag{9.58}$$

and

Integer Data

$$\text{IF} \quad v_i \in F_i^t - F_i^s \quad \text{for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} a_{it} x_i \geq 0\}\,, \tag{9.59}$$

In (9.57) to (9.59) the inequality sign may be replaced by an equality sign.

Edits of type (9.58) for which $F_i^t \cap F_i^s = \varnothing$ may be discarded. Likewise, edits of type (9.59) for which $F_i^t - F_i^s = \varnothing$ may also be discarded. After all edits of types (9.58) and (9.59) have been generated, the selected edit with (9.56) as THEN-condition is deleted. Edits given by (9.57) not involving x_r in their THEN-condition are not modified, neither are they deleted.

It is obvious that the new system of edits is equivalent to the original system of edits. Namely, for the categorical values for which we can use equation (9.56) to eliminate variable x_r, we do this (see (9.58)). For the categorical values for which we cannot use equation (9.56) to eliminate this variable, we simply leave x_r untouched (see (9.59)). An implicit edit of type (9.58) where x_r has been eliminated will never be combined with an edit still involving x_r, because the IF-conditions of these edits have an empty overlap (see also Section 9.5.2).

We continue eliminating the equalities until they have all been eliminated. Note that these equalities will be eliminated after finitely many steps. Each time we eliminate an equality we may generate more equalities (see (9.58) and (9.59)), but at least one of the sets of categorical values in the IF-conditions corresponding to the new equalities is smaller than the set of categorical values in the IF-condition of the edit (9.57) under consideration. After a finite number of steps, these sets of categorical values inevitably become empty. After all equalities have been eliminated, we are left with only linear inequalities involving only integer variables as THEN-conditions. Because additional, auxiliary variables may have been introduced to eliminate variables from equalities while the original variables may still occur in other edits, the total number of integer-valued variables in this system of inequalities may be larger than the original number of integer-valued variables. How we deal with a set of THEN-conditions consisting of inequalities involving only integer-valued variables is explained in the next section.

9.5.2. Error localisation: eliminating integer variables from inequalities

We assume that all THEN-conditions are linear inequalities involving only integer variables. When an integer variable is eliminated from a set of linear inequalities, a dark shadow and possibly several splinters are generated. Below we describe how this dark shadow and these splinters are defined.

We start by selecting an integer variable that we want to eliminate, say variable x_r. All current edits not involving x_r are retained. The current edits involving x_r are combined into implicit edits not involving x_r.

We consider all edits involving x_r pair-wise. Such a pair of edits is given by

$$\begin{aligned} &\text{IF} && v_i \in F_i^s \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} a_{is} x_i \geq 0\} \end{aligned} \qquad (9.60)$$

and

$$\begin{aligned} &\text{IF} && v_i \in F_i^t \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} a_{it} x_i \geq 0\} \end{aligned} \qquad (9.61)$$

where all involved numerical variables are integer-valued. We assume that the a_{is} (i=0,…,n) are normalised, i.e. they are integer and their greatest common divisor equals 1. Similarly, the a_{it} (i=0,…,n) are assumed to be normalised.

Like the real shadow, the dark shadow is only defined if $a_{rs} \times a_{rt} < 0$. In that case one coefficient is larger than zero, say $a_{rs} > 0$, and the other coefficient is less than zero, $a_{rt} < 0$. The dark shadow of the above pair of edits (9.60) and (9.61) obtained by eliminating x_r is then defined as

$$\begin{aligned} &\text{IF} && v_i \in F_i^s \cap F_i^t \quad \text{for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid \sum_{i=0}^{n} (a_{rs}a_{it} - a_{rt}a_{is}) x_i \geq (a_{rs}-1)(-a_{rt}-1)\}. \end{aligned} \qquad (9.62)$$

If $F_i^t \cap F_i^s$ is empty for some categorical variable i (i=1,…,m), edit (9.62) is deleted. Once again, the IF-condition of an implicit edit (9.62) is given by the intersections $F_i^s \cap F_i^t$

(i=1,…,m), because two numerical THEN-conditions can only be combined into an implicit edit for the overlapping parts of their corresponding categorical IF-conditions.

Defining the splinters obtained by eliminating x_r from a set of edits with only linear inequalities as THEN-conditions is more tricky than in Section 9.4.2. The reason is that here we want to define splinters for various combinations of categorical values, whereas in Section 9.4.2 we only had to define the splinters once. There are two possibilities to define splinters.

According to the first possibility we determine the intersections $F_i^s \cap F_i^t \equiv M_i(s,t)$ (for i=1,…,m) for each pair of edits s and t with $a_{rs} > 0$ and $a_{rt} < 0$. We only consider those pairs s and t for which $M_i(s,t)$ is non-empty for all i. For each unique combination of the sets $M_i(s,t)$ (i=1,…,m) occurring in the current set of edits we determine the largest coefficient a_{ri} of x_r for all upper bounds on x_r in THEN-conditions with corresponding IF-conditions that are triggered by this combination of the $M_i(s,t)$ (i=1,…,m). Suppose that this largest coefficient occurs in the u-th edit. For each lower bound

$$\begin{aligned} &\text{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\text{THEN} && \sum_{i \neq r} a_{ij} x_i \leq a_{rj} x_r \end{aligned} \tag{9.63}$$

we then test whether an integer solution exists to the original edits combined with

$$\begin{aligned} &\text{IF} && v_i \in F_i^u \cap F_i^j \text{ for } i=1,\ldots,m \\ &\text{THEN} && a_{rj} x_r = \sum_{i \neq r} a_{ij} x_i + k \end{aligned} \tag{9.64}$$

for each k satisfying $(a_{ru}a_{rj} - a_{ru} - a_{rj})/a_{ru} \geq k \geq 0$ (k integer). That is, $\lfloor (a_{ru}a_{rj} - a_{ru} - a_{rj})/a_{ru} \rfloor + 1$ splinters are considered for each unique combination of the sets $M_i(s,t)$ (i=1,…,m) occurring in the current set of edits.

If $F_i^u \cap F_i^j$ is empty for some categorical variable i, edit (9.64) is not used. The intersections $F_i^u \cap F_i^j$ ensure that the equalities (9.64) are only defined for those combinations of categorical values for which they should be defined and not for others.

This way to define splinters has the advantage that for each combination of the sets $M_i(s,t)$ (i=1,…,m) occurring in the data we will not consider too many subproblems. It

has the disadvantage, though, that the largest coefficient a_{ru} has to be determined for each unique combination of the sets $M_i(s,t)$ (i=1,…,m) separately, which is possibly a lot of work.

The alternative way to define splinters is to determine the largest coefficient a_{ri} of x_r for all upper bounds on x_r in THEN-conditions. Say this largest coefficient is given by a_{rq}. For each lower bound (9.63) we then test whether an integer solution exists to the original edits combined with

$$\begin{aligned} &\text{IF} \quad && v_i \in F_i^j \ \text{ for } i=1,\ldots,m \\ &\text{THEN} \quad && a_{rj}x_r = \sum_{i \neq r} a_{ij}x_i + k \end{aligned} \tag{9.65}$$

for each k satisfying $(a_{rq}a_{rj} - a_{rq} - a_{rj})/a_{rq} \geq k \geq 0$ (k integer).

This way of defining the splinters has the advantage that only one upper bound has to be determined. It has the disadvantage that for certain combinations of categorical values possibly too many subproblems have to be considered. Defining splinters in the second way has the effect that more subproblems have to be considered than defining splinters in the first way. This becomes clear as soon as we combine the relations given by (9.65) with the edits: we get the relations given by (9.64) plus some additional ones that are redundant. The redundant relations correspond to redundant subproblems. Examining a redundant problem only costs more computing time, but does not do any harm otherwise.

Independent on which of the two ways of defining splinters is used, we have the following important theorem.

Theorem 9.3. The original set of edits with linear inequalities involving only integer variables as THEN-conditions has a solution if and only if the dark shadow or a splinter resulting from the elimination of variable x_r has a solution.

Proof. It is clear that if the original set of edits has a solution, then the same solution is also a solution to either the dark shadow or a splinter.

Suppose on the other hand that the dark shadow or a splinter has a solution. Fill the values of this solution in into all original edits. This results in a set of edits involving only the eliminated integer variable x_r. Now, an integer value exists for variable x_r such that all numerical THEN-conditions of the original edits that are triggered by the categorical values involved in the solution to the dark shadow or a splinter become satisfied. Namely, suppose such an integer value would not exist. In that case there is a set of edits with contradicting THEN-conditions. These edits have a non-empty categorical intersection, because they are all triggered by the categorical values of the solution to the dark shadow or a splinter. Hence, these edits would have generated a dark shadow (defined by (9.62))

and splinters (defined by (9.64) or (9.65)) without any solutions. This contradicts our assumption that there is a solution to the dark shadow or a splinter. ■

Theorem 9.3 is again a lifting principle. The existence of a solution for a certain set of (implicit) edits is lifted to a set of edits involving more variables.

9.5.3. Error localisation: an algorithm for categorical, continuous and integer data

We denote the error localisation problem in categorical, continuous and integer data, i.e.

> Minimise (9.8) so that (9.9) is satisfied for all edits j=1,…,J, x_i is integer for $i \in I$, and the remaining x_i are continuous,

by P_I.

To solve the error localisation problem for a mix of categorical, continuous and integer-valued data we first apply the branch-and-bound algorithm presented in Chapter 8 without taking into account that some of the variables are integer-valued, i.e. we first treat these variables as being continuous. We denote the problem where integer variables are treated as continuous ones by P_C.

Let B denotes the value of (9.8) for the currently best solution to P_I, and **S** the current set of optimal solutions to P_I. We initialise B to ∞, and **S** to $\varnothing$.

A solution to P_C not involving any integer variables is automatically also a solution to P_I. So, whenever we find a solution to P_C not involving any integer variables for which (9.8) is less than B, we update B with that value of (9.8) and set **S** equal to the current solution to P_C. Furthermore, whenever we find a solution to P_C not involving any integer variables for which (9.8) is equal to B, we add the current solution to P_C to **S**.

Whenever we find a solution to problem P_C involving integer variables for which (9.8) is at most equal to B, we consider problem P_I. We know, of course, which variables have been eliminated to arrive at the current optimal solution to problem P_C. We now check whether these variables also constitute a solution to P_I. To do this we first eliminate the continuous variables. This yields a system in which only integer-valued and categorical variables occur. The current edits are edits of type (9.9) where all involved numerical variables x_i are integer-valued.

Next, we eliminate all integer-valued variables involved in the found current optimal solution to P_C in the manner described in Sections 9.5.1 and 9.5.2. During this process the original problem may be split into several subproblems due to the splinters that may arise. Subsequently, we eliminate all categorical variables from these subproblems. For each subproblem we end up with a set of relations not involving any unknowns. This set of relations may be empty. If a set of relations we obtain in this way does not contain a

contradiction, which is for instance (by definition) the case if the set of relations is empty, we have found a current optimal solution to P_I. In that case, if the value of (9.8) for the current solution to P_I is less than B we update B accordingly and set **S** equal to the current solution of P_I, else we add the current solution to P_I to **S**. If none of the subproblems leads to a solution to P_I, the solution to P_C under consideration is not a solution to P_I. In that case B is not updated, and we continue with finding solutions to P_C.

In the above approach, the relatively time-consuming algorithm to check for solutions of P_I is only invoked once a solution to P_C with an objective value of B or less involving integer-valued variables has been found, so only rather infrequently.

If we only want to find all optimal solutions to P_I in the case that the optimal value of (9.8) is less than a certain maximum B_0, we simply initialise B to B_0. All solutions to P_I for which the value of (9.8) is larger than B_0 will then automatically be discarded.

We have the following theorem.

Theorem 9.4. The above procedure finds all optimal solutions to problem P_I.

Proof. The proof that our procedure finds all optimal solutions to problem P_I is similar to the proofs of Theorems 8.2 and 8.3 that the branch-and-bound algorithm for categorical and continuous data finds all optimal solutions to the corresponding problem.

We start by noting that a set of variables is a solution to P_I if and only if eliminating these variables does not lead to inconsistencies in all subproblems. For all three kinds of variables a lifting principle holds true: if we eliminate a variable, the current set of edits can be satisfied if and only if at least one of the resulting sets of implicit edits can be satisfied. Repeated application of the lifting principle shows that if and only if any of the final sets of relations does not contain a contradiction, the original set of edits can be satisfied.

Now, the branch-and-bound algorithm for categorical and continuous data can be used to find all solutions to the corresponding problem P_C with an objective value (9.8) of B or less. For each solution to P_C with a value for (9.8) equal to or less than B, we check whether it is also a solution to P_I. The result of this test is conclusive. We update B whenever we have found a better solution to P_I than the best one found so far. In other words, all possible optimal solutions to P_I are considered by the procedure, and all optimal solutions to P_I are indeed identified as such. ∎

We illustrate the procedure by means of the simple example involving only integer-valued variables below.

Example 9.3:
We consider the case where we have only three variables x_1, x_2 and x_3, and only four edits that are given below.

$$x_1 = 0\,, \tag{9.66}$$

$$-2x_3 + 5 \geq 0\,, \tag{9.67}$$

$$5x_2 - x_3 \geq 0 \tag{9.68}$$

and

$$-3x_2 + 2x_3 \geq 0\,. \tag{9.69}$$

All three variables are integer-valued, and all reliability weights equal one. The original, incorrect, record is given by $(x_1, x_2, x_3) = (0, 1, 1)$. We only want to find all optimal solutions where the value of at most one variable has to be changed. Solutions where the values of two or even all three variables have to be changed are not considered. To this end we initialise B to 1. The set of optimal solutions **S** is, of course, initially set to $\varnothing$.

We start by solving problem P_C. To this end we select a variable, say x_1, and construct two branches: in the first branch we eliminate x_1 from the current set of edits, in the second branch we fix x_1 to its original value. The first branch corresponds to deciding to change the value of x_1, the second branch corresponds to accepting the original value of x_1 as being correct.

When we eliminate x_1 from the set of edits we obtain (9.67) to (9.69) as our new current set of edits. This set of edits is not satisfied by the original values of x_2 and x_3. We need not consider this branch any further, because we are only interested in solutions where the value of at most one variable has to be changed.

We now consider the branch where x_1 is fixed to its original value 0. Our new current set of edits is again given by (9.67) to (9.69). We again select a variable, say x_2, and construct two branches: in the first branch we eliminate x_2 from the current set of edits, in the second branch we fix x_2 to its original value.

When we eliminate x_2 from the current set of edits we obtain (9.67) and $x_3 \geq 0$ as our new current set of edits. This new current set of edits is satisfied by the original value of x_3. Hence, we have found a solution to P_C. The value of (9.8) for this solution to P_C is equal to 1. We therefore check whether changing the value of x_2 only is also a solution to P_I.

To check whether changing the value of only x_2 is a solution to P_I we start by filling in the original values of x_1 and x_3 into the original set of edits. We obtain the set of edits involving only x_2 given below.

$$5x_2 - 1 \geq 0 \tag{9.70}$$

and

$$-3x_2 + 2 \geq 0\,. \tag{9.71}$$

We do not need to check whether the real shadow obtained by eliminating x_2 from these two inequalities is consistent. This is necessarily the case, because changing the value of x_2 is a solution to P_C.

The dark shadow (see (9.62)), obtained by eliminating x_2 from (9.70) and (9.71), is given by

$$7 \geq 8, \tag{9.72}$$

which is clearly a contradiction.

We therefore have to consider the splinters. In this simple case there are only two splinters. For the first splinter we have to add the constraint

$$5x_2 = 1 \tag{9.73}$$

to (9.70) and (9.71), for the second one we have to add the constraint

$$5x_2 = 2 \tag{9.74}$$

to (9.70) and (9.71). It is clear that neither of the two splinters has an integer solution for x_2. This also follows if we would apply the proposed algorithm. We can conclude that although changing x_2 is a solution to P_C, it is not a solution to P_I.

After this intermezzo during which we checked whether changing the value of only x_2 is a solution to P_I we continue with finding solutions to P_C. We now consider the branch where both x_1 and x_2 are fixed to their original values. The current set of edits is given by (9.67),

$$-x_3 + 5 \geq 0 \tag{9.75}$$

and

$$2x_3 - 3 \geq 0\,. \tag{9.76}$$

By eliminating x_3 we see that changing the value of only x_3 is a solution to P_C. The value of (9.8) for this solution to P_C is equal to 1. We therefore check whether changing the value of x_3 only is also a solution to P_I.

To check whether changing the value of only x_3 is a solution to P_I we start by filling in the original values of x_1 and x_2 into the original set of edits. We obtain the system (9.67), (9.75) and (9.76). Edit (9.75) is clearly redundant and can be discarded. The real shadow

obviously is consistent. The dark shadow of (9.67) and (9.76) (see (9.62)) obtained by eliminating x_3 is given by

$$4 \geq 1. \tag{9.77}$$

Relation (9.77) is consistent, so we can conclude that changing the value of x_3 is a solution to P_C. We can even conclude that this is the only optimal solution to P_C. A feasible value for x_3, the only one in this case, is 2. ■

9.6. Discussion

The method described in the previous section may appear to be very slow in many cases. Indeed, it is not difficult to design sets of edits for which the method is extremely slow. However, we argue that in practice the situation is not so bad. First, as we already mentioned the time-consuming algorithm to check for solutions of P_I is only invoked once a new solution to P_C with an objective value less than or equal to the current value of B has been found. In practice, the number of times that such a solution to P_C is found is in most cases rather limited.

Second, once we have found a new solution to P_C with a target value less than or equal to the current value of B we only have to test whether the variables occurring in this particular solution also form a solution to P_I. Moreover, often one is only interested in solutions to the error localisation problem with a few variables, say less than 10. In case a record requires more values to be changed than a pre-set limit, the record will not be edited automatically for else the statistical quality of the edited record would be too low. This means that the number of variables involved in our test will never be more than the pre-set limit. In other words, our test involves only a few variables.

Third, the algorithm to check for solutions of P_I becomes only really time-consuming when many splinters have to be considered. However, in most edits, either explicit or implicit ones, encountered in practice the coefficients of many integer variables equal –1 or +1. This is especially true for edits expressing equalities. For integer variables with coefficient –1 or +1 elimination from inequalities will be exact, i.e. the dark shadow and real shadow coincide and no splinters have to be generated. For the same reason, elimination of equalities can generally be performed very fast in practice.

Fourth, we can resort to simple heuristics to improve the performance of the algorithm. For instance, we may first try to find a solution to P_I without splitting the problem into subproblems. That is, after all continuous variables involved in the current solution to P_C have been eliminated we repeatedly use relation (9.62) to eliminate the integer-valued variables involved in this solution to P_C. Each time we thus compute a dark shadow until all integer variables involved in the current solution to P_C have been eliminated. This results in a system in which only categorical variables occur. We eliminate the categorical variables involved in the current solution to P_C to check whether a solution P_I to exists.

If eliminating these categorical variables does not lead to a contradiction, we have found a current optimal solution to P_I. Only when the sketched approach does not lead to a solution to P_I, we could try to find a solution to P_I by splitting the problem into a number of subproblems, not necessarily all subproblems. The simplest approach is, of course, not splitting the problem into subproblems at all, but assuming instead that if the above-sketched approach does not lead to a solution to P_I, P_I does not have a solution. By not checking all subproblems, we can restrict the number of subproblems that we have to consider. The price we have to pay for this is that we may sometimes conclude that an integer solution to P_I does not exist, whereas in fact it does.

We conclude that the performance of the described test for checking the integrality of a proposed solution is likely to be good in practice. Moreover, there is hardly an alternative to applying this test. The simple approach of not applying any test for the integrality of the solutions, and rounding the found continuous solutions does not always work. In practice, cases do occur where the found continuous solutions cannot be rounded to integer values such that all edits become satisfied. The only remaining alternative to the described test we see at the moment is the application of a commercial solver for mixed integer programming problems. Using such a solver is, however, expensive, and is likely to be more time consuming than our simple test.

10. Cutting Plane Algorithms

10.1. Introduction

Garfinkel, Kunnathur and Liepins (1986) present a so-called cutting plane algorithm for solving the error localisation problem for categorical data. The algorithm is based on solving the set-covering problem associated to the error localisation problem (see e.g. Section 3.3), and subsequently adding constraints, cuts, to this set-covering problem in case the set-covering solution turns out to be infeasible for the error localisation problem. This algorithm is described in Section 10.2.

In 1988 Garfinkel, Kunnathur and Liepins proposed a similar cutting plane algorithm for continuous data. This algorithm was improved upon by Ragsdale and McKeown (1996), who instead of the associated set-covering problem considered a modified set-covering problem associated to the error localisation problem. These algorithms are described in Section 10.3. Our proof that these algorithms find an optimal solution to the error localisation problem in a finite number of steps is more general than the proof given by Garfinkel, Kunnathur and Liepins (1988), and is not restricted to only continuous data.

In Section 10.4 we propose an extension of the algorithms by Garfinkel, Kunnathur and Liepins (1986 and 1988) to a mix of categorical and continuous data. Such an extension has not yet been described in literature before. Our algorithm is again a cutting plane algorithm.

In the previous two chapters we have developed theory for solving the error localisation problem. The basis of this theory is formed by the elimination methods of Sections 8.2 and 9.5 for continuous, integer-valued and categorical data. Now, we utilise the developed elimination techniques to develop alternative cutting plane algorithms.

The algorithms we propose in Sections 10.5 to 10.8 of this chapter are similar to the ones proposed by Garfinkel, Kunnathur and Liepins (1986 and 1988), and Ragsdale and McKeown (1996). The new algorithms are again cutting plane algorithms, and as in the algorithms of Sections 10.2 to 10.4 a potential solution to the error localisation problem is proposed that is subsequently checked for feasibility. Instead of executing the first phase of the simplex algorithm (if any continuous variable is involved in the potential solution) for each combination of values of the categorical variables (if any categorical variable is involved in the potential solution) involved in the potential solution as in the algorithms of Sections 10.2 to 10.4, we now eliminate all variables involved in the proposed solution. Continuous variables are eliminated by (an extension of) Fourier-Motzkin elimination, categorical variables are eliminated by an elimination method proposed by Fellegi and Holt (1976). By eliminating the variables we can check the feasibility of a proposed solution and simultaneously generate additional cuts in case the proposed solution turns out to be infeasible. In other words, the former two steps of checking the feasibility of a proposed solution and generating additional constraints are combined into a single step. Intuitively, our proposed algorithm therefore seems to be more efficient than the ones due

to Garfinkel, Kunnathur and Liepins (1986 and 1988), and Ragsdale and McKeown (1996).

In Section 10.5 we propose a cutting plane algorithm for categorical or continuous data based on elimination. Section 10.6 illustrates this algorithm by means of an example. In Section 10.7 we extend this cutting plane algorithm to a mix of categorical and continuous data, and Section 10.8 we further extend the algorithm to a mix of categorical, continuous and integer data. This latter algorithm is the culmination of our work on cutting plane algorithms so far. Section 10.9 concludes the chapter with a brief discussion. The material in Sections 10.5 and 10.9 are original work.

Sections 10.2 to 10.4 are based on De Waal (1997a), Sections 10.5 to 10.8 are based on De Waal (2002b).

10.2. The Garfinkel, Kunnathur and Liepins method for categorical data

We assume that a record $(v_1^0,...,v_m^0)$ with only categorical data has to be edited. The edits we consider are given by:

$$\text{IF } v_i \in F_i^j \text{ (for } i=1,\ldots,m) \text{ THEN } \varnothing. \tag{10.1}$$

An edit given by (10.1) is violated if $v_i \in F_i^j$ for all i=1,…,m. Otherwise, the edit is satisfied.

The cutting plane algorithm of Garfinkel, Kunnathur and Liepins for categorical data is defined in the following way:

Algorithm 10.1:

1. Determine the constraints of the set-covering problem associated with the error localisation problem (see Section 3.3).
2. Solve the set-covering problem. Its solution is denoted by $\hat{\mathbf{y}}$.
3. Let $\hat{I} = \{i \mid \hat{y}_i = 1\}$. Set $v_i = v_i^0$ for $i \notin \hat{I}$. For every $i \in \hat{I}$, let v_i assume each of the values in D_i. Test for each of the $\prod_{i \in \hat{I}} |D_i|$ records constructed in this way whether they satisfy all edits. If such a record is found, then $\hat{\mathbf{y}}$ corresponds to a solution to the error localisation problem, and we are done. Otherwise, go to Step 4.
4. Construct any prime cover, i.e. a cover such that if any element is removed from the cover the remaining set is no longer a cover, $\mathbf{u^0}$ to

$$\mathbf{uQ} \geq 1, \tag{10.2}$$

$$u_j \in \{0,1\}, \tag{10.3}$$

where the matrix $\mathbf{Q}$ is defined by

$$q_{jk} = \begin{cases} 1 & \text{if } E^j \text{ is failed by record } k \text{ of Step 3} \\ 0 & \text{otherwise.} \end{cases} \tag{10.4}$$

Let $\hat{J} = \{j \mid u_j^0 = 1\}$.

5. Now the following edit is implied by the explicit ones:

$$\hat{F}_i = \begin{cases} D_i & i \in \hat{I} \\ \bigcap_{j \in \hat{J}} F_i^j & i \notin \hat{I} \end{cases} \tag{10.5}$$

6. Add the set-covering constraint corresponding to the implied edit (10.5) to the already defined set-covering constraints, and return to Step 2.

The algorithm works because of the following reasons. Any record satisfying the explicit edits automatically satisfies the implied edit (10.5) as we demonstrate in the next paragraph. Any feasible solution to the error localisation problem is also a feasible solution to the associated set-covering problem (with constraints corresponding to the explicit edits and the already constructed implied edits). So, any optimal solution to the set-covering problem in Step 2 that can be imputed consistently is an optimal solution to the error localisation problem. Step 3 checks whether an optimal solution to the current set-covering problem can be imputed consistently. The algorithm terminates after a finite number of iterations, because due to the added constraints in Step 6 during each iteration a different optimal cover to the set-covering problem is found in Step 2. Such a cover found in Step 2 is also a cover of the violated explicit edits. There are only finitely many different covers of the violated explicit edits.

It has to be shown that the edit constructed in Step 5 is indeed an implied edit. This can be proved by making the following two observations. First, after Steps 3 and 4 it is clear that the edit E^0 defined by $\hat{F}_i = D_i$ for $i \in \hat{I}$, and $\hat{F}_i = \{v_i^0\}$ for $i \notin \hat{I}$ is implied by the edits in $\hat{J}$. Second, according to Theorem 2 of Fellegi and Holt (1976), for any implied edit E there is, in turn, an implied edit E' of the following form:

$$\widetilde{F}_i = \bigcap_{r \in S} F_i^r \quad \text{for } i=1,\ldots,m; i \neq i_0 \tag{10.6}$$

and

$$\widetilde{F}_{i_0} = \bigcup_{r \in S} F_{i_0}^r, \tag{10.7}$$

where S is a subset of the explicit edits, such that when E is violated then so is E'. In other words, E is dominated by E' in the sense that E' whenever E is.

Because edit E^0 is an implied edit, it has to be dominated by an implied edit of the above form. Therefore, we can conclude that edit (10.5) is indeed an implied edit.

Steps 4 and 5 may seem redundant, because after Step 3 we can immediately deduce that the edit given by $\hat{F}_i = D_i$ for $i \in \hat{I}$, and $\hat{F}_i = \bigcap_{j \in \Omega_V} F_i^j$ for $i \notin \hat{I}$ (where Ω_V is the set of all violated explicit edits) is violated. We can then add the corresponding constraint to the

set-covering problem, and return to Step 2. However, the set-covering constraint generated by that edit can be less strong, i.e. involves more variables, than the set-covering constraint generated by the edit determined in Steps 4 and 5. This is the case when $\bigcap_{j\in\Omega_{V'}} F_i^j \neq D_i$ and $\bigcap_{j\in\hat{J}} F_i^j = D_i$ for $i \notin \hat{I}$. In other cases the same set-covering constraint is generated by both implied edits. By using the possibly stronger constraint determined in Steps 4 and 5 we hope to find a solution to the error localisation problem in less iterations than when we use the simple edit which we can deduce immediately after Step 3.

Determining a prime cover in Step 4 is easy, and does not cost much time. Just start with the cover consisting of all violated edits, and then sequentially discard any edit not uniquely responsible for covering of at least one record considered in Step 3. So, it seems worthwhile performing Steps 4 and 5 of the algorithm, because probably more time is won because less iterations have to be performed than lost because prime covers have to be determined.

An even stronger set-covering constraint can be obtained by determining an optimal cover, i.e. a cover with a minimum of non-zero coefficients, in Step 4 instead of just any prime cover. The resulting implied edit is generally stronger than when just any prime cover is determined, but determining an optimal cover is much more difficult than determining an arbitrary prime cover. All in all, it is likely that more time is lost than gained, when we solve the set-covering problem of Step 4 to optimality.

In principle, Step 2 is the most difficult step to solve. Standard techniques for solving set-covering problems may be too slow when the problems become large. In such a case one should resort to simple heuristics (see e.g. Vasko and Wilson, 1986; Vazirani, 2001). However, as it is not likely that many values of variables are incorrect and that many explicit edits are violated the standard techniques may be fast enough in most cases. Moreover, when there are many mistakes in a record, then the information in this record is probably worthless anyway, and one may just as well discard this record entirely.

10.3. A cutting plane algorithm for error localisation in numerical data

In this section we assume that a record with only numerical data is to be edited. We therefore assume that the edits are given by

$$a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\,, \tag{10.8}$$

The edits of the form (10.8) have to hold simultaneously for a record to pass all edits. Equivalently, we can say that a record is faulty if at least one of the relations

$$a_{1j}x_1 + \ldots + a_{nj}x_n + b_j < 0 \tag{10.9}$$

holds true.

Given a record with values $(x_1^0,\ldots,x_n^0)$, we want to modify the values of a set of variables such that all inequalities of (10.8) are satisfied, and such that the sum of the reliability weights of the modified variables is minimal. To solve this problem Garfinkel, Kunnathur and Liepins (1988) propose a cutting plane algorithm that follows the following outline.

Algorithm 10.2:

1. Determine the edits of (10.8) that are violated by the record under consideration. If no edits are violated, we are done. Otherwise, these violated edits correspond to constraints of the associated set-covering problem, and we go to Step 2.
2. Determine an optimal cover of the constraints of the set-covering problem, i.e. a cover with a minimal sum of reliability weights. The optimal cover is denoted by $\hat{\mathbf{y}}$, and the index set of the optimal cover is denoted by $\hat{I}$, i.e. $\hat{I} = \{i \mid \hat{y}_i = 1\}$.
3. Determine a set of reduced constraints for the variables x_i $(i \in \hat{I})$. This set of reduced constraints can be written as

$$\sum_{i \in \hat{I}} a_{ij} x_i + b'_j \geq 0 \qquad (j=1,\ldots,R), \tag{10.10}$$

where

$$b'_j = \sum_{i \notin \hat{I}} a_{ij} x_i^0 + b_j \qquad (j=1,\ldots,R). \tag{10.11}$$

4. If the system (10.10) has a feasible solution then the solution to the set-covering problem corresponds to an optimal solution to the error localisation problem, and we are done. Otherwise we go to Step 5.
5. Determine an infeasible subset Ω_M of constraints of (10.10).
6. Add the following constraint to the set-covering problem:

$$\sum_{i=1}^{n} a_i y_i \geq 1, \tag{10.12}$$

where

$$a_i = \begin{cases} 0 & \text{if } i \in \hat{I}, \text{ or if variable } i \text{ is not involved in } \Omega_M \\ 1 & \text{otherwise.} \end{cases} \tag{10.13}$$

Go to Step 2.

Example 10.1:

Before we comment on this algorithm we illustrate it by applying it to Example 3.1 of Section 3.3. So, we assume that the explicit edits are given by (3.10) to (3.15). The values of the variables in a certain record are given by $T = 100$, $P = 40{,}000$, $C = 60{,}000$ and $N = 5$. The reliability weights of variables T, P and C equal 1, the reliability weight of variable N equals 2.

Edits (3.10) and (3.11) are violated. The associated set-covering problem is given by:

$$\text{Minimise } y_T + y_P + y_C + 2y_N \tag{10.14}$$

under the constraints

$$y_T + y_P + y_C \geq 1, \tag{10.15}$$

$$y_T + y_C \geq 1 \tag{10.16}$$

and

$$y_T, y_P, y_C, y_N \in \{0,1\}. \tag{10.17}$$

The optimal solution to this problem is $y_T = 1$ and $y_P = y_C = y_N = 0$. The set of reduced constraints considered in Step 3 is given by

$$T = 100{,}000, \tag{10.18}$$

$$0.5 \leq \frac{60{,}000}{T} \leq 1.1 \tag{10.19}$$

and

$$T \leq 550. \tag{10.20}$$

This clearly is an infeasible system of constraints. We select an infeasible subset of constraints, say (10.18) and (10.20). The explicit edits corresponding to these constraints generate a violated implied edit, namely

$$P + C \leq 550N. \tag{10.21}$$

The set-covering constraint corresponding to this violated implied edit is given by

$$y_P + y_C + y_N \geq 1. \tag{10.22}$$

We add this set-covering constraint to the other set-covering constraints (10.15) to (10.17), and minimise the objective function (10.14) under the updated system of set-covering constraints. The optimal solution to the new set-covering problem is $y_P = y_C = 1$ and $y_T = y_N = 0$.

The set of reduced constraints of Step 3 is now given by

$$P + C = 100, \tag{10.23}$$

$$0.5 \leq \frac{C}{100} \leq 1.1 \tag{10.24}$$

and

$$100 \leq 550. \tag{10.25}$$

This is a feasible system. A solution is for instance P = 40 and C = 60. The optimal solution to the error localisation problem is therefore: change the values of variables P and C. ■

Ragsdale and McKeown (1996) strengthen the algorithm by Garfinkel, Kunnathur and Liepins (1988) by noting that the set-covering problem does not take the direction of

change of the numerical variables into account. The above formulation only says that a variable of each violated edit should be changed. It does not consider pairs of edits. However, in order to satisfy a violated edit by changing a particular variable it may be necessary to increase its value whereas to satisfy another violated edit is may be necessary to decrease its value. Ragsdale and McKeown (1996) replace each numerical variable x_i in the violated edits by $x_i^0 + x_i^+ - x_i^-$, where x_i^0 denotes the observed value in the data set, $x_i^+ \geq 0$ a positive change in value, and $x_i^- \geq 0$ a negative change in value. This yields a set of constraints for the x_i^+ and the x_i^-. Next, they specify a modified set-covering problem involving binary variables y_i^+ and y_i^- corresponding to the x_i^+ and x_i^- rather than the x_i. Example 10.2, which is taken from Ragsdale and McKeown (1996), illustrates the modified set-covering problem.

Example 10.2:

Suppose only two edits involving three numerical variables have been specified, and that edits are given by

$$2x_1 + x_2 \leq 5 \tag{10.26}$$

and

$$-x_1 + x_3 \leq 1\,. \tag{10.27}$$

Suppose, furthermore, that an erroneous record given by $\mathbf{x_0} = (1,4,3)$ is to be edited automatically, and that the reliability weights are given by $\mathbf{w} = (1,2,3)$. Both edits are violated. The associated set-covering problem is then given by

$$\text{Minimise } y_1 + 2y_2 + 3y_3 \tag{10.28}$$

subject to

$$y_1 + y_2 \geq 1\,, \tag{10.29}$$

$$y_1 + y_3 \geq 1\,, \tag{10.30}$$

$$y_1, y_2, y_3 \in \{0,1\}\,. \tag{10.31}$$

A solution to this set-covering problem is: $\mathbf{y} = (1,0,0)$. Changing x_1 is, however, not a feasible solution to the error localisation problem.

However, when we replace x_i in the violated edits by $x_i^0 + x_i^+ - x_i^-$, we obtain the following two constraints for the x_i^+ and the x_i^-:

$$-2x_1^+ - x_2^+ + 2x_1^- + x_2^- \geq 1 \tag{10.32}$$

and

$$x_1^+ - x_3^+ - x_1^- + x_3^- \geq 1 . \tag{10.33}$$

To satisfy (10.32) at least one of the variables x_1^- and x_2^- must assume a positive value. To satisfy (10.33) at least one of the variables x_1^+ and x_3^- must assume a positive value. This leads to the following modified set-covering problem:

$$\text{Minimise } y_1^+ + y_1^- + 2y_2^- + 3y_3^- \tag{10.34}$$

subject to

$$y_1^- + y_2^- \geq 1 , \tag{10.35}$$

$$y_1^+ + y_3^- \geq 1 , \tag{10.36}$$

$$y_1^+ + y_1^- \leq 1 , \tag{10.37}$$

$$y_i^+ , y_i^- \in \{0,1\} \text{ , for } i=1,\ldots,3. \tag{10.38}$$

The constraints (10.35), (10.36) and (10.38) are the familiar set-covering constraints. Constraint (10.37) expresses that x_1 cannot be increased and decreased simultaneously. Due to such constraints the model by Ragsdale and McKeown leads to a *modified* set-covering problem instead of an ordinary set-covering problem.

The solution to the above modified set-covering problem is: $y_1^+ = y_2^- = 1$ and $y_1^- = y_3^+ = 0$. The corresponding, potential solution to the error localisation problem, change x_1 and x_2, turns out to be indeed feasible. ■

The algorithm proposed by Ragsdale and McKeown (1996) for solving the error localisation problem for continuous data is the same as the algorithm proposed by Garfinkel, Kunnathur and Liepins (1988) except for the fact that a modified set-covering problem instead of a set-covering problem is solved in Step 2.

The modified set-covering problem proposed by Ragsdale and McKeown cuts off certain infeasible solutions to the error localisation problem that are not cut off by the standard set-covering problem. In this sense the formulation by Ragsdale and McKeown is stronger than the standard set-covering formulation. The price that has to be paid is that standard methods for solving set-covering problems cannot be applied to solve the modified set-covering problem.

We now discuss the algorithms by Garfinkel, Kunnathur and Liepins, and Ragsdale and McKeown in some detail. In Step 4 we have to check whether the set of reduced constraints of Step 3 has a feasible solution. This can be done by standard linear programming techniques. The feasibility of a set of linear constraints is checked in the first step of the simplex method, for instance (see e.g. Hadley, 1962, and Chvátal, 1983).

Step 5 is the most interesting step. It corresponds to Steps 4 and 5 in Algorithm 10.1 proposed by Garfinkel, Kunnathur and Liepins (1986). Any infeasible subset of (10.10), for example the entire set (10.10), suffices to determine an optimal solution to the error localisation problem in the end, but the number of iterations needed to arrive at this solution is influenced by the selected subset of constraints of (10.10). Before we consider the problem of selecting a suitable subset of constraints of (10.10) we first show that the above algorithms Garfinkel, Kunnathur and Liepins (1986) and Ragsdale and McKeown (1996) succeeds in finding an optimal solution to the error localisation problem in a finite number of iterations.

Theorem 10.1. The above algorithms determine an optimal solution to the error localisation problem in a finite number of iterations.

Proof. When a subset Ω of the constraints of the system (10.10) is not feasible, there is apparently a contradiction between the variables not involved in the cover. According to Corollary 2 to Theorem 1 of Fellegi and Holt (1976) there is an implied edit generated by the edits in Ω in which only the variables are involved that are not part of the cover. In Step 6 the (modified) set-covering constraint corresponding to such an implied edit is added to the system of set-covering constraints. Any solution to the error localisation problem has to satisfy this (modified) set-covering constraint. Therefore, a solution to the error localisation problem is a solution to any of the (modified) set-covering problems considered in Step 2. On the other hand, a solution to a (modified) set-covering problem considered in Step 2 is a solution to the error localisation problem if and only if the variables involved in the cover can indeed be imputed, i.e. if and only if system (10.10) has a feasible solution. So, when an optimal cover of a (modified) set-covering problem is determined in Step 2 such that the variables involved in this cover can be imputed consistently then the error localisation problem has been solved optimally.

The algorithms terminate after a finite number of iterations, because the (modified) set-covering constraints added in Step 6 ensure that during each iteration a different (modified) cover is the optimal solution to the (modified) set-covering problem considered in Step 2. As there are only finitely many (modified) covers only finitely many iterations are necessary to arrive at the optimal solution to the error localisation problem. Hereby we have shown that the algorithm can be used to solve the error localisation problem to optimality. ■

The above proof differs from the one originally given by Garfinkel, Kunnathur and Liepins (1988), which was based on Farkas' Lemma. Our proof is more general as it is not restricted to continuous data, but – in principle – also applies to categorical data or even to a mix of categorical and continuous data. The advantage of our proof over the one given by Garfinkel, Kunnathur and Liepins will become apparent in the next section when we consider a more general theorem, Theorem 10.2.

There are several options for selecting a suitable subset Ω_M of constraints of (10.10) in Step 5 of the algorithm. First, we can select all violated constraints of (10.10). This is the easiest solution, but has the disadvantage that the resulting set-covering constraint can be

rather weak. As a consequence at lot of iterations may be necessary to arrive at the optimal solution. This method of determining Ω_M corresponds to not applying Steps 4 and 5 in Algorithm 10.1 proposed Garfinkel, Kunnathur and Liepins (1986) (see Section 10.2).

Second, we can apply the following simple algorithm. Consider a constraint of type (10.10) and check whether we can remove this constraint without making the remaining set of constraints feasible. If so, we remove this constraint from the set of constraints (10.10), else we retain the constraint. Next we consider another constraint which has not been checked before. While checking this other constraint we use the updated set of constraints (10.10), i.e. some constraints from the original set (10.10) may have been removed. We go on with this process until all constraints of (10.10) have been checked. In the end we are left with an infeasible set of constraints such that if any constraint is removed then the system becomes feasible. Because the system of constraints is infeasible, the corresponding explicit edits generate a violated implicit edit. This is ensured by Corollary 2 to Theorem 1 of Fellegi and Holt (1976). This method of determining Ω_M corresponds to Steps 4 and 5 of Algorithm 10.1 proposed by Garfinkel, Kunnathur and Liepins, because the set Ω_M is a prime cover of a *p*-dimensional real vector space, where *p* equals the number of elements in $\hat{I}$.

We can exploit the fact that we can choose the order in which the constraints of (10.10) are examined in the above algorithm. We suggest determining the order of the constraints in the following way. To select a constraint of (10.10) we calculate for each constraint that has not been checked already the number of variables V_n involved in that constraint that are not involved in constraints that have already been checked and that have been retained. The constraint with the highest number V_n is chosen to be checked. The idea of this approach is that for the first constraints of (10.10) that are checked it is (a bit) more likely that they are removed than for later constraints. By selecting the order of the constraints in the above manner we hope that the number of variables involved in the constraints that are retained is low. As a consequence the corresponding set-covering constraints are likely to be rather strong.

Third, we can apply the approach proposed by Garfinkel, Kunnathur and Liepins (1988). We write the set of explicit edits (10.8) as

$$A\mathbf{x} \geq \mathbf{b}\,, \tag{10.39}$$

We consider the set of reduced constraints given by (10.10) and (10.11). We write this system as

$$B\mathbf{x}_{\hat{\mathbf{i}}} + \mathbf{b}' \geq 0\,. \tag{10.40}$$

Now, we determine a solution to

$$\mathbf{p}B = 0\,, \tag{10.41}$$

$$\mathbf{p}\mathbf{b}' \leq -1 \tag{10.42}$$

and

$$\mathbf{p} \geq \mathbf{0} . \tag{10.43}$$

The edits i for which $p_i > 0$ are included in Ω_M. In fact, we construct the (modified) set-covering constraint corresponding to the implied edit

$$\mathbf{p}A\mathbf{x} + \mathbf{pb} \geq \mathbf{0} . \tag{10.44}$$

Fourth, we can construct an implied edit with a minimal number of involved variables. For this we have to solve the following integer programming problem. Minimise the objective function

$$\sum_{i \notin \hat{I}} \delta(z_i) , \tag{10.45}$$

where $\delta(z_i)$ equals 1 if $z_i \neq 0$, otherwise it equals 0, under the constraints

$$\sum_{j=1}^{R} \lambda_j = 1 , \tag{10.46}$$

$$\sum_{j=1}^{R} \lambda_j a_{ij} = 0 \quad \text{for } i \in \hat{I} , \tag{10.47}$$

$$\sum_{j=1}^{R} \lambda_j (a_{1j} v_1 + \ldots + a_{nj} v_n + b_j) < 0 , \tag{10.48}$$

$$z_i = \sum_{j=1}^{R} \lambda_j a_{ij} \quad \text{for } i \notin \hat{I} \tag{10.49}$$

and

$$\lambda_j \geq 0 \quad \text{for } j=1,\ldots,R. \tag{10.50}$$

In this way we construct an implied edit given by

$$\sum_{i \in \hat{I}} \sum_{j=1}^{R} \lambda_j a_{ij} x_i + \sum_{j=1}^{R} b_j \geq 0 . \tag{10.51}$$

This implied edit is violated by the record under consideration, because constraint (10.48) has to be satisfied. Moreover, the number of variables involved in this edit is minimal, because the objective function (10.45) is minimised.

It is important to notice that the set of constraints (10.46) to (10.50) has a feasible solution. Again this is ensured by Corollary 2 to Theorem 1 of Fellegi and Holt (1976).

The problem of minimising (10.45) under the constraints (10.46) to (10.50) can, for example, be solved by Chernikova's algorithm, or by standard techniques for solving integer programming problems such as branch-and-bound algorithms. The problems that have to be solved are relatively small. In any case they are much smaller than the error localisation problem itself. So, solving the above problem can be done relatively fast. Still,

it is not clear whether solving the above problem leads to better result than the simple heuristics discussed before.

10.4. A cutting plane algorithm for error localisation in mixed data

For a mix of categorical and continuous data we propose a combination of the two algorithms by Garfinkel, Kunnathur and Liepins (1986 and 1988), which are suitable for categorical data and continuous data, respectively.

We write an edit of type

$$\begin{aligned} &\text{IF} && v_i \in F_i^j \text{ for } i=1,\ldots,m \\ &\text{THEN} && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}, \end{aligned} \tag{10.52}$$

in the following way: an edit E^j is failed if and only if

$$v_i \in F_i^j \text{ for } i=1,\ldots,m \tag{10.53}$$

and

$$(x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j < 0\}, \tag{10.54}$$

where $F_i^j \subseteq D_i$ (D_i is the domain of variable i). The set on the right-hand side of (10.54) may be empty or the entire n-dimensional real vector space.

We assume that a record $(v_1^0,\ldots,v_m^0,x_1^0,\ldots,x_n^0)$ has to be edited. We propose the following cutting plane algorithm.

Algorithm 10.3:

1. Determine the edits given by (10.53) and (10.54) that are violated by the record under consideration. If no edits are violated, we are done. Otherwise, these violated edits correspond to the constraints of the associated set-covering problem, and we go to Step 2.
2. Determine an optimal cover of the constraints of the set-covering problem, i.e. a cover with a minimal sum of reliability weights. The optimal cover is denoted by $\hat{\mathbf{y}}$, and the index set of the optimal cover is denoted by $\hat{I}$, i.e. $\hat{I} = \{i \mid \hat{y}_i = 1\}$. Here $\hat{y}_i = 1$ (for $i = 1,\ldots,m$) indicates that the value of the i-th categorical variable should be modified, and $\hat{y}_{i+m} = 1$ (for $i = 1,\ldots,n$) indicates that the value of the i-th numerical variable should be modified. The set $\hat{I}$ can be decomposed into two subsets: a subset $\hat{I}_c$

corresponding to categorical variables only, and a subset $\hat{I}_n$ corresponding to numerical variables only.

3. Set $v_i^k = v_i^0$ for $i \in \{1,...,m\} - \hat{I}_c$, and $x_i = x_i^0$ for $i \in \{1,...,n\} - \hat{I}_n$. For every $i \in I_c$, let v_i^k assume each of the values in D_i. The index k refers to one of the $\prod_{i \in \hat{I}_c} |D_i|$ combinations of values of categorical variables constructed in this manner. For each of these combinations we can determine a set of reduced constraints for the numerical variables x_i $(i \in \hat{I}_n)$. For the k-th combination of values of categorical variables $(v_1^k,...,v_m^k)$ such a set of reduced constraints can be written as

$$\sum_{i \in \hat{I}_n} a_{ij} x_i + b'_j \geq 0 \tag{10.55}$$

for all j such that

$$v_i^k \in F_i^j \qquad \text{for } i=1,\ldots,m, \tag{10.56}$$

where

$$b'_j = \sum_{i \notin \hat{I}_n} a_{ij} x_i^0 + b_j \,. \tag{10.57}$$

The constraints given by (10.55) have to hold simultaneously.

4. If any of the sets of reduced constraints (10.55) considered in Step 3 has a feasible solution then the solution to the set-covering problem corresponds to an optimal solution to the error localisation problem, and we are done. Otherwise we go to Step 5.

5. Determine an infeasible subset Ω_M of edits given by (10.53) and (10.54). That is, determine a subset of edits such that for each combination k of values of categorical variables with indices in $\hat{I}_c$, the resulting set of reduced constraints for variables with indices in $\hat{I}_n$ is infeasible.

6. Add the following constraint to the set-covering problem:

$$\sum_{i=1}^{m+n} a_i y_i \geq 1\,, \tag{10.58}$$

where

$$a_i = \begin{cases} 0 & \text{if } i \in \hat{I}\text{, or if variable } i \text{ is not involved in } \Omega_M \\ 1 & \text{otherwise.} \end{cases} \tag{10.59}$$

Go to Step 2.

Theorem 10.2. This algorithm determines an optimal solution to the error localisation problem in a finite number of iterations.

Proof. See the proof of Theorem 10.1. ∎

Step 4 can be done by linear programming techniques. For instance, for each set of reduced constraints (10.55) we can use the first step of the simplex method to test the feasibility of this set. Step 4 can be time-consuming when the product $\prod_{i \in \hat{I}_c} |D_i|$ is large. However, if this product is large then usually $\hat{I}_c$ contains many elements. If this is the case, then the record under consideration contains many errors. Hence, the record contains little information. In such a case we may just as well remove the entire record from the data set.

Again Step 5 is the most interesting step. The problem of selecting a suitable subset of constraints Ω_M is somewhat more difficult than the similar problem discussed in Section 10.3. The difference is that in Section 10.3 we should select a suitable infeasible subset of constraints, whereas here we should select a subset of constraints Ω_M such that for each combination of values of categorical variables considered in Step 3 the corresponding set of numerical constraints given by (10.55) is infeasible. In other words, here we should select Ω_M in such a way that several subsets of constraints of Ω_M are infeasible. Nevertheless, although the problem of selecting a suitable subset of constraints Ω_M is more complicated in this case than in the case of Section 10.3 we can still use similar heuristics to determine Ω_M.

10.5. A cutting plane algorithm for continuous or categorical data based on elimination

From this section on we will use the elimination techniques developed in Sections 8.2 and 9.5 to improve upon the algorithms proposed in the previous sections of this chapter.

In this section we assume that the edits given by

$$\text{IF} \quad v_i \in F_i^j \text{ for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}, \tag{10.60a}$$

or

$$\text{IF} \quad v_i \in F_i^j \text{ for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}, \tag{10.60b}$$

are either purely continuous or purely categorical. In other words, either all numerical THEN-conditions of edits of type (10.60) always have to be satisfied irrespective of the values of the categorical variables, or the set in the THEN-condition of each edit of type (10.60) is the empty set.

Our proposed algorithm for solving the error localisation problem in continuous or categorical data is given below.

Algorithm 10.4:

0. We denote the original set of edits that has to be satisfied by Ω_0. We define $\Gamma_0 = \varnothing$, and set *s* equal to 1.

1. Determine the edits in Ω_{s-1} that are violated by the record under consideration. If no edits are violated, we are done. Otherwise, go to Step 2.

2. The violated edits of Ω_{s-1} correspond to constraints of the associated set-covering problem. For the *j*-th violated constraint E^j of Ω_{s-1}, the corresponding constraint for the set-covering problem is given by:

$$\sum_{i=1}^{m+n} d_{ij} y_i \geq 1, \tag{10.61}$$

where

$$d_{ij} = \begin{cases} 0 & \text{if variable } i \text{ is not involved in } E^j \\ 1 & \text{otherwise.} \end{cases} \tag{10.62}$$

Let $\Gamma_s := \Gamma_{s-1} \cup \{\text{new set - covering constraints}\}$. Go to Step 3.

3. Determine all optimal covers for the constraints of the set-covering problem defined by Γ_s. That is, determine all covers with a minimal sum of reliability weights. The optimal covers are denoted by $\hat{y}^k$, and the index set of the *k*-th optimal cover (k=1,..., K_s) is denoted by $\hat{I}^k$, i.e. $\hat{I}^k = \{i \mid \hat{y}_i^k = 1\}$. If the sum of the reliability weights of these solutions exceeds $N_{\max}$, the maximum (weighted) number of fields that may be modified (see Section 8.6.1), we stop: the record is considered too erroneous for automatic editing, and is discarded. Otherwise, go to Step 4.

4. For each optimal cover k=1,..., K_s, eliminate all variables in $\hat{I}^k$, i.e. all variables in this cover, from Ω_0. If the resulting set of edits, denoted by Ω_s^k, does not contain any edit violated by the original values of the remaining variables, the variables in $\hat{I}^k$ form an optimal solution to the error localisation problem.

If any optimal solution to the error localisation problem has been found in Step 4, we output all found optimal solutions and stop. If no optimal solution to the error localisation problem has been found, we go to Step 5.

5. Set $\Omega_s = \bigcup_{k=1}^{K_s} \Omega_s^k$, and let $s := s + 1$. Go to Step 1.

We have the following theorem.

Theorem 10.3. After termination of Algorithm 10.4, all optimal solutions to the error localisation problem for either continuous or categorical data have been determined. Moreover, Algorithm 10.4 is guaranteed to terminate after a finite number of iterations.

Proof. A necessary condition for a set S of variables to be a feasible solution to the error localisation problem is that after elimination of these variables the original values of the remaining variables satisfy the edits (explicit ones and implied ones obtained by elimination of variables in S) involving only these latter variables. So, none of these edits for the latter variables may be violated. This means that for each violated edit (either explicit or implied) at least one variable should be changed, i.e. at least one variable entering this edit should be part of a solution to the error localisation problem. This shows that any solution to the error localisation problem is also be a solution to the set-covering problem defined by minimising the sum of reliability weights subject to (10.61) where the constraints correspond to explicit edits plus generated implied edits. Hence, any optimal solution to the error localisation problem is also a solution to the associated set-covering problem.

It remains to show that a set S of variables is indeed a feasible solution to the error localisation problem if the original values of the remaining variables satisfy the explicit and implicit edits obtained by elimination of the variables in S from the original set of edits. This follows directly from Theorem 8.2 that says that a set of variables S' is a solution to the error localisation problem if and only if the set of relations involving no unknowns obtained by eliminating the variables in S' from the original edits and fixing the values of the remaining variables to their original values is consistent.

It is easy to see that Algorithm 10.4 terminates after a finite number of iterations as there are only finitely many different set-covering problems, each with finitely many optimal solutions. Whenever an optimal solution to the current set-covering problem does not correspond to a feasible solution to the error localisation problem, cuts are added to the set-covering problem (see Steps 2 and 5) in order make this optimal set-covering solution infeasible for all subsequent set-covering problems. ■

The proposed algorithm can be modified in two different ways. First, instead of generating and solving the standard set-covering problem associated to an instance of the error localisation problem, one could generate and solve the modified set-covering problem proposed by Ragsdale and McKeown (1996) (see also Section 10.3). The results by

Ragsdale and McKeown suggest that this modified algorithm will be faster than the algorithm formulated above.

Second, instead of determining *all* optimal solutions to the (modified) set-covering problem in Step 3, one could determine only *one* optimal solution. To fulfil our aim of finding all optimal solutions to the error localisation problem we must then perform an additional step to find all optimal solutions to the error localisation problem after we have found one. This additional step could be carried out in the following way. Once we have found an optimal solution to the error localisation problem we re-run the last iteration. This time, however, we do find all optimal solutions to the (modified) set-covering problem in Step 3, and check for all those solutions whether they are feasible for the error localisation problem.

Finding only one optimal solution to a (modified) set-covering problem in an iteration is obviously faster than determining all optimal solutions. However, in general more iterations will be needed before the algorithm terminates and, moreover, an additional step is required to find all optimal solutions to the error localisation problem. Intuitively, we would prefer to generate all optimal solutions in each iteration. However, without further research it is unclear whether finding all optimal solutions to the (modified) set-covering problem in each iteration is really better than finding only one optimal solution.

Theorem 10.3 remains valid if Algorithm 10.4 is modified in either of the two (or both) ways.

In the algorithm generated implicit edits are not stored. However, in a practical implementation one could, of course, decide to store generated implicit edits in order to avoid having to generate the same implicit edits several times.

10.6. Example

We illustrate the algorithm described in the previous section by means of a small example involving only four continuous variables. Suppose the explicitly specified edits are given by

$$T = P + C, \tag{10.63}$$

$$P \leq 0.5T, \tag{10.64}$$

$$-0.1T \leq P, \tag{10.65}$$

$$T \geq 0, \tag{10.66}$$

$$T \leq 550N, \tag{10.67}$$

where T denotes the turnover of an enterprise, P its profit, C its costs, and N the number of employees. Let us consider a specific erroneous record with values $T = 100$, $P = 40{,}000$, $C = 60{,}000$ and $N = 10$. Edits (10.65), (10.66) and (10.67) are satisfied, whereas edits (10.63) and (10.64) are violated. The reliability weights of variables T, P and C equal 1, and the reliability weight of variable N equals 2. That is, the value of variable N, the number of employees, is considered more reliable than the values of the financial variables T, P and C.

Edits (10.63) and (10.64) are violated. The associated set-covering problem is hence given by:

$$\text{Minimise } y_T + y_P + y_C + 2y_N \tag{10.68}$$

subject to the constraints

$$y_T + y_P + y_C \geq 1, \tag{10.69}$$

$$y_T + y_P \geq 1, \tag{10.70}$$

$$y_T, y_P, y_C, y_N \in \{0,1\}. \tag{10.71}$$

Constraint (10.69) says that at least one of the variables *T*, *P* or *C* should be changed, and constraint (10.70) that at least one of the variables *T* or *P* should be changed.

The optimal solutions to this problem are

a) $y_T = 1$ and $y_P = y_C = y_N = 0$;

b) $y_P = 1$ and $y_T = y_C = y_N = 0$.

Potential solutions to the error localisation problem are hence:

a) change the value of *T*;

b) change the value of *P*.

We first eliminate *T* from the explicit edits (10.63) to (10.67) using the equality-elimination rule (see Section 8.2) to check whether changing *T* is indeed a solution to the error localisation problem. We obtain the following new edits:

$$P \leq 0.5(P + C) \quad \text{(combination of (10.63) and (10.64))} \tag{10.72}$$

$$-0.1(P + C) \leq P \quad \text{(combination of (10.63) and (10.65))} \tag{10.73}$$

$$P + C \geq 0 \quad \text{(combination of (10.63) and (10.66))} \tag{10.74}$$

$$P + C \leq 550N \quad \text{(combination of (10.63) and (10.67))} \tag{10.75}$$

Edits (10.72) to (10.74) are satisfied, edit (10.75) is violated. Because edit (10.75) is violated, changing the value of *T* is not a solution to the error localisation problem.

We now eliminate *P* from the explicit edits (10.63) to (10.67) using the equality-elimination rule to check whether changing *P* is a solution to the error localisation problem. We obtain:

$$T - C \leq 0.5T, \quad \text{(combination of (10.63) and (10.64))} \tag{10.76}$$

$$-0.1T \leq T - C, \quad \text{(combination of (10.63) and (10.65))} \tag{10.77}$$

$$T \geq 0, \tag{10.78}$$

$$T \leq 550N. \tag{10.79}$$

Edit (10.77) is violated, edits (10.76), (10.78), and (10.79) are satisfied. Because edit (10.77) is violated, changing the value of P is not a solution to the error localisation problem.

The set-covering constraints corresponding to violated edits (10.75) and (10.77) are given by

$$y_P + y_C + y_N \geq 1\,, \tag{10.80}$$

respectively

$$y_T + y_C \geq 1 \tag{10.81}$$

We add these set-covering constraints to the other set-covering constraints (10.69) to (10.71), and minimise target function (10.68) subject to the updated system of set-covering constraints. The optimal solutions to the new set-covering problem are:

a) $y_P = y_C = 1$ and $y_T = y_N = 0$;

b) $y_T = y_C = 1$ and $y_P = y_N = 0$;

c) $y_P = y_T = 1$ and $y_C = y_N = 0$.

Potential optimal solutions to the error localisation problem are hence:

a) change the values of P and C;

b) change the values of T and C;

c) change the values of P and T.

We first check the first potential solution to the error localisation problem by eliminating P and C from all explicit edits (10.63) to (10.67). We first eliminate P, and again obtain (10.76) to (10.79). Now, we eliminate C from (10.76) to (10.79). We obtain

$$0.5T \leq 1.1T\,, \qquad \text{(combination of (10.76) and (10.77))} \tag{10.82}$$

$$T \geq 0\,, \tag{10.83}$$

$$T \leq 550N\,. \tag{10.84}$$

This set of edits is satisfied by the values of the remaining variables. An optimal solution to the error localisation problem is therefore: change the values of variables P and C.

Now, we check the second potential solution to the error localisation problem by eliminating T and C from all explicit edits (10.63) to (10.67). We first eliminate T, and again obtain (10.72) to (10.75). Now, we eliminate C from (10.72) to (10.75). We obtain several new edits. One of those edits is

$$2P \leq 550N\,. \qquad \text{(combination of (10.72) and (10.75))} \tag{10.85}$$

This edit is violated. Changing T and C is therefore not a solution to the error localisation problem.

Finally, we check the third potential solution to the error localisation problem by eliminating P and T from all explicit edits (10.63) to (10.67). We first eliminate P, and again obtain (10.76) to (10.79). Now, we eliminate T from (10.76) to (10.79). We obtain several new edits. One of those edits is

$$\frac{C}{1.1} \leq 550N \,. \qquad \text{(combination of (10.77) and (10.79))} \qquad (10.86)$$

This edit is violated. Changing P and T is therefore not a solution to the error localisation problem.

We conclude that there is only one optimal solution to the error localisation problem, namely: change the values of P and C.

10.7. A cutting plane algorithm for categorical and continuous data based on elimination

Thus far we have discussed an algorithm for the error localisation problem for either categorical or continuous data. We now consider the error localisation problem for a mix of categorical and continuous data. The cutting plane algorithm we propose for the error localisation problem in categorical and continuous data simultaneously, Algorithm 10.5, is similar to Algorithm 10.4. In fact, only Step 4 of Algorithm 10.5 differs from Step 4 of Algorithm 10.4.

Step 4 of Algorithm 10.5:

4. For each optimal cover k=1,..., K_s, eliminate all variables in $\hat{I}^k$, i.e. all variables in this cover, from Ω_0. For each $\hat{I}^k$ we first eliminate the continuous variables involved in this cover. For edits also involving categorical variables in $\hat{I}^k$ we then fix the remaining continuous variables to their original values, and finally eliminate the categorical variables in $\hat{I}^k$. If the resulting set of constraints, Ω_s^*, does not contain any constraint violated by the values of the remaining variables, the variables in $\hat{I}^k$ form an optimal solution to the error localisation problem.

   In case no optimal solution to the error localisation problem has been found for any cover k we define $\Omega_s^k = \varnothing$ for k=1,..., K_s. For each cover k we then add each implied edit in Ω_s^* obtained by eliminating only continuous variables, and each implied edit in Ω_s^* obtained by eliminating only categorical variables from edits Ω_0 not involving any continuous variables, to Ω_s^k.

   If a constraint in Ω_s^* has been obtained by eliminating at least one categorical variable (and possibly some continuous variables) from edits involving continuous variables, we construct the following set-covering constraint:

$$\sum_{i \in T - \hat{I}^k} y_i \geq 1, \tag{10.87}$$

where T denotes the set of variables involved in the edits that were used to eliminate the variables in $\hat{I}^k$ in order to obtain the constraint in Ω_s^* under consideration. Such a set-covering constraint says that the value of at least one variable not in $\hat{I}^k$ involved in the edits used to eliminate the variables involved in $\hat{I}^k$ in order to obtain the constraint under consideration should be changed. The set-covering constraints defined by (10.87) are added to Γ_{s+1}.

If any optimal solution to the error localisation problem has been found in Step 4, we output all found optimal solutions and stop. If no optimal solution to the error localisation problem has been found, we go to Step 5.

As Algorithm 10.4, Algorithm 10.5 can be modified in either of two ways (or both): by generating and solving a modified set-covering problem instead of a set-covering problem, and by determining only one optimal solution to the (modified) set-covering problem instead of all optimal solutions. If Algorithm 10.5 is modified in the latter way, the last iteration needs to be re-run in order to find all optimal solutions to the error localisation problem (see also Section 10.5). For (the modified versions of) Algorithm 10.5 we have a similar theorem as Theorem 10.3.

Theorem 10.4. After the termination of Algorithm 10.5, or a modified version, all optimal solutions to the error localisation problem for a mix of continuous and categorical data have been determined. Moreover, Algorithm 10.5 is guaranteed to terminate after a finite number of iterations.

This theorem can be proven in the same manner as Theorem 10.3.

10.8. A cutting plane algorithm for general data based on elimination

In this section we consider the error localisation problem for general data, i.e. a mix of categorical, continuous and integer data. Again we propose a cutting plane algorithm. This algorithm, Algorithm 10.6, is an extension of Algorithm 10.5. Again, the only difference between Algorithm 10.6 and the previous algorithms is Step 4. Below we describe Step 4 of Algorithm 10.6.

Step 4 of Algorithm 10.6:

4. For each optimal cover k=1,..., K_s, eliminate all variables in $\hat{I}^k$, i.e. all variables in this cover, from Ω_0. For each $\hat{I}^k$ we first eliminate the numerical variables in this

cover in the normal manner for continuous data. For edits also involving categorical variables in $\hat{I}^k$ we then fix the remaining numerical variables to their original values, and finally eliminate the categorical variables in $\hat{I}^k$. If the resulting set of constraints, Ω_s^*, does not contain any constraint violated by the values of the remaining variables, we know that the variables in $\hat{I}^k$ would form an optimal solution to the error localisation problem if all variables were either categorical or continuous. We then say that a solution to the categorical and continuous error localisation problem, shortly a *continuous* solution, has been found.

If such a continuous solution does not involve any integer-valued variables, it is obviously also a solution to the general error localisation problem, shortly an *integer* solution.

If a continuous solution does involve integer-valued variables, we apply the method described in Section 9.5 to check whether this continuous solution is also an integer solution.

In case no integer solution to the error localisation problem has been found we define $\Omega_s^k = \varnothing$ for k=1,..., K_s. For each cover k we now consider two cases:

a) Cover k does not correspond to a continuous solution.

   In this case we do the same as in Step 4 of Algorithm 10.5. For each cover k we then add each implied edit in Ω_s^* obtained by eliminating only continuous variables, and each implied edit in Ω_s^* obtained by eliminating only categorical variables from edits in Ω_0 not involving any continuous variables, to Ω_s^k. If a constraint has been obtained by eliminating at least one categorical variable (and possibly some continuous variables) from edits involving continuous variables, we construct a set-covering constraint of type (10.87). All above set-covering constraints defined by (10.87) are added to Γ_{s+1}.

b) Cover k corresponds to a continuous solution.

   In this case, we add the set-covering constraint

$$\sum_{i \notin \hat{I}^k} y_i \geq 1, \tag{10.88}$$

   i.e. the set-covering constraint saying that at least one of the variables not involved in optimal cover k should be modified, to Γ_{s+1}.

If any optimal integer solution has been found in Step 4, we output all found optimal solutions and stop. If no optimal integer solution to the error localisation problem has been found, we go to Step 5.

Once again, Algorithm 10.6 can be modified in either of two ways (or both): by generating and solving a modified set-covering problem, and by determining only one optimal

solution to the (modified) set-covering problem instead of all optimal solutions. Similar to the previous algorithms proposed in this chapter, if Algorithm 10.6 is modified in the latter way, the last iteration needs to be re-run in order to find all optimal solutions to the error localisation problem. For (the modified versions of) Algorithm 10.6 we have a similar theorem as Theorems 10.3 and 10.4.

Theorem 10.5. After the termination of Algorithm 10.6, or a modified version, all optimal solutions to the error localisation problem for a mix of categorical, continuous and integer data have been determined. Moreover, Algorithm 10.6 is guaranteed to terminate after a finite number of iterations.

This theorem can be proven in the same way as Theorem 10.3. The main things to notice are that the method described in Section 9.5 provides an exact test to decide whether a continuous solution is also an integer solution, and that constraint (10.88) cuts off continuous solutions that are not integer solutions.

10.9. Discussion

In this chapter cutting plane algorithms for the error localisation problem are presented. The basic algorithm of Section 10.4 is able to deal with both categorical and continuous data simultaneously. The main aspect of this algorithm that remains to be examined is the selection of Ω_M. To decide which method of selecting Ω_M is the best, computational experience with various selection methods proposed in this chapter, and possibly some other methods, should be gained.

In Sections 10.5 to 10.8 of this chapter we have extended cutting plane methods proposed by Garfinkel, Kunnathur and Liepins (1986 and 1988) and Ragsdale and McKeown (1996) for solving either the categorical or the continuous error localisation problem to cutting plane methods for solving the error localisation problem for a mix of categorical and continuous data, and even for a mix of categorical, continuous and integer-valued data. These methods seem to be more efficient than the algorithms of Sections 10.2 to 10.4 as checking the feasibility of a proposed solution to the error localisation problem and generating additional set-covering cuts in case the proposed solution turns out to be infeasible are combined into a single step.

11. Computational results

11.1. Introduction

In this chapter we compare computational results for four different algorithms for automatic error localisation on six data sets involving only numerical data. The aim of our comparison study is not to perform a comprehensive evaluation study for all possible data sets, but rather to perform a succinct evaluation study that allows us to identify the most promising algorithm(s) for a number of realistic data sets. We restrict ourselves to data sets involving exclusively numerical data for two reasons. First, automatic data editing of economic – and hence (mostly) numerical – data is a far more important subject for Statistics Netherlands than automatic data editing of social – and hence (mostly) categorical – data. Second, the complexity of the algorithms, and corresponding computer programs, increases drastically if they are to be applied to a mix of categorical and numerical data instead of only to numerical data. Due to restrictions in time and available resources we want to extent only the most promising program(s) to a mix of categorical and continuous data.

In literature some evaluation studies are already described, see Garfinkel, Kunnathur and Liepins (1986 and 1988), Kovar and Winkler (1996), and Ragsdale and McKeown (1996). Garfinkel, Kunnathur and Liepins (1986) concentrate on error localisation for purely categorical data. The other papers concentrate on error localisation for purely numerical data. It is difficult to compare our results to the described results for numerical data. First, because in most cases the actual computing speeds of the computer systems used in those studies are difficult to retrieve, and hence difficult to compare to the computing speed of present-day PC's. Second, because the used data sets with their edit rules are not publicly available.

The algorithms we examine are an algorithm based on a standard mixed integer programming (MIP) formulation that is solved by means of the commercial MIP-solver ILOG CPLEX (see Chapter 3), a vertex generation algorithm (see Chapter 5), a non-standard branch-and-bound algorithm (see Chapter 8), and a cutting plane algorithm (see Chapter 10).

The remainder of this chapter is organised as follows. In Section 11.2 we describe the data sets that we have used for our evaluation study. In Section 11.3 we provide some information regarding the implementation of the above-mentioned algorithms. The computational results are summarised in Section 11.4. Section 11.5 concludes the chapter with a brief discussion.

11.2. The data sets

For our evaluation experiments we have used realistic data. In several cases we have used actually observed data, in other cases observed data have been slightly perturbed in order to prevent disclosure of confidential information. Due to confidentiality restrictions the

values and names of the variables are not mentioned in this book. In the sequel of this section we briefly describe the characteristics of the six data sets, such as the number of variables, number of records, and the edits corresponding to each data set.

11.2.1. Data set A

Data set A contains 90 continuous variables X_1 to X_{90}, and 4,347 records. All records contain missing values, and are hence considered inconsistent. In total there are 259,838 missing values, i.e. 59.8 missing values on average per record. All variables are non-negative, i.e. $X_i \geq 0$ for i=1,…,90. Besides these non-negativity constraints, there are 8 other edits involving only the first 16 variables, which are given by:

$$-X_1 + 100\,X_5 + 100\,X_9 + X_{13} \geq 0$$

$$-X_2 + 100\,X_6 + 100\,X_{10} + X_{14} \geq 0$$

$$-X_3 + 100\,X_7 + 100\,X_{11} + X_{15} \geq 0$$

$$-X_4 + 100\,X_8 + 100\,X_{12} + X_{16} \geq 0$$

$$X_1 + 0.99\,X_5 + 0.99\,X_9 - X_{13} \geq 0$$

$$X_2 + 0.99\,X_6 + 0.99\,X_{10} - X_{14} \geq 0$$

$$X_3 + 0.99\,X_7 + 0.99\,X_{11} - X_{15} \geq 0$$

$$X_4 + 0.99\,X_8 + 0.99\,X_{12} - X_{16} \geq 0.$$

For the other 74 variables only the non-negativity constraints have to be satisfied. The first 16 variables contain 18,555 missing values in total.

11.2.2. Data set B

Data set B contains 76 continuous variables X_1 to X_{76}, and only 274 records of which 157 are inconsistent, i.e. do not satisfy all edits, and 117 consistent. The data set contains no missing values, because missing values were imputed with the value zero before the data set was delivered to the author. The reason for this pre-processing step is that in reality most missing values in this data set should be zeros. All variables, except X_9, X_{14}, X_{15}, X_{16}, X_{21}, and X_{27}, are non-negative. Besides the non-negativity restrictions, the following edits should be satisfied.

$$X_{14} + X_{15} - X_{16} = 0$$

$$X_9 + X_{10} - X_{11} + X_{12} + X_{13} - X_{14} = 0$$

$$X_1 - X_8 - X_9 = 0$$

$$X_2 + X_3 + X_4 + X_5 + X_6 + X_7 - X_8 = 0$$

$X_1 - X_{28} = 0$

$X_{55} + X_{57} + X_{59} + X_{61} + X_{63} + X_{65} + X_{67} + X_{69} - X_{71} = 0$

$X_{56} + X_{58} + X_{60} + X_{62} + X_{64} + X_{66} + X_{68} - X_{70} = 0$

$X_{51} - X_{52} - X_{53} = 0$

$X_{46} + X_{48} + X_{49} + X_{50} - X_{51} = 0$

$X_{40} + X_{41} + X_{42} + X_{43} + X_{44} - X_{45} = 0$

$X_{35} + X_{38} - X_{39} = 0$

$X_{36} + X_{37} - X_{38} = 0$

$X_{29} + X_{30} + X_{31} + X_{32} + X_{33} + X_{34} - X_{35} = 0$

$X_{20} + X_{21} + X_{22} + X_{23} + X_{24} + X_{25} + X_{26} + X_{27} - X_{28} = 0$

$X_{17} + X_{18} + X_{19} - X_{20} = 0$

$X_7 - X_{45} = 0$

$X_5 - X_{39} = 0$

$X_{72} + X_{73} - X_{74} = 0$

$-X_{53}+X_{54} \leq 0$

$-X_{46}+X_{47} \leq 0.$

Note that the above set of edits can be split into several sets of edits involving disjoint sets of variables. The error localisation problem could be solved for each of these edit sets separately. This would generally lead to reduced computing times for our algorithms. We have, however, not performed this pre-processing step. We have neither split the edit set for any of the other data sets.

11.2.3. Data set C

Data set C contains 53 continuous variables X_1 to X_{53}, and 1,480 records of which 1,404 are inconsistent and 76 consistent. The data set contains no missing values, because missing values were replaced by zeros before the data set was delivered to the author. All variables, except X_2, X_3, X_{15}, X_{16}, X_{17}, X_{19}, X_{20}, X_{24}, X_{25}, X_{28}, X_{31}, X_{34}, X_{35}, X_{39}, X_{41}, X_{50}, and X_{51}, are non-negative. Besides the non-negativity restrictions, the following edits should be satisfied.

$-X_{25} - X_{28} - X_{31} - X_{34} + X_{35} = 0$

$X_{18} - X_{47} = 0$

$X_{12} - X_{38} = 0$

$X_{16} - X_{41} = 0$

$X_{23} - X_{53} = 0$

$X_{14} - X_{40} = 0$

$X_{13} - X_{39} = 0$

$X_{20} - X_{51} = 0$

$-X_{42} - X_{43} - X_{44} - X_{45} - X_{46} + X_{47} = 0$

$-X_{32} + X_{33} + X_{34} = 0$

$-X_{29} + X_{30} + X_{31} = 0$

$-X_{26} + X_{27} + X_{28} = 0$

$-X_{19} + X_{24} + X_{25} = 0$

$-X_{20} - X_{22} - X_{23} + X_{24} = 0$

$-X_{15} - X_{16} - X_{17} - X_{18} + X_{19} = 0$

$-X_{12} + X_{13} + X_{14} + X_{15} = 0$

$-X_{36} - X_{37} + X_{38} = 0$

$-X_{7} + X_{8} + X_{9} + X_{10} + X_{11} = 0$

$-X_{1} - X_{2} + X_{3} = 0$

$X_{3} - X_{4} - X_{5} - X_{6} = 0$

$X_{12} - X_{37} \geq 0$

$33\ X_{3} - X_{22} \geq 0$

$0.2\ X_{7} - X_{10} \geq 0$

$0.2\ X_{7} - X_{9} \geq 0$

$0.2\ X_{7} - X_{8} \geq 0$

$-0.25\ X_{11} + X_{12} - X_{14} \geq 0$

$-0.09\ X_{11} - X_{21} + X_{50} \geq 0$

$-0.02\ X_{12} + X_{23} \geq 0$

$-100\ X_{3} + X_{11} \geq 0$

$-20\ X_{3} + X_{20} \geq 0$

$0.33\ X_{12} - X_{23} \geq 0$

$110\ X_{3} - X_{20} \geq 0$

$0.7\ X_{12} - X_{13} - 0.7\ X_{14} \geq 0$

$270\ X_{3} - X_{11} \geq 0$

$2.4\ X_{11} - X_{12} + X_{14} \geq 0$

$0.47\ X_{11} + X_{21} - X_{50} \geq 0.$

11.2.4. Data set D

Data set D contains 4,217 records, of which 2,152 are inconsistent and 2,065 consistent. Data set D contains 51 continuous variables X_1 to X_{51}. The data set contains no missing values, because as in data sets B and C missing values were replaced by zeros before the data set was delivered to the author. All variables, except X_{42} and X_{43}, are non-negative. Besides the non-negativity restrictions, the following edits should be satisfied.

$X_{11} - X_{16} - X_{17} + X_{18} - X_{19} + X_{20} - X_{21} + X_{22} + X_{26} - X_{30} - X_{34} - X_{35} + X_{36} - X_{37} + X_{38} - X_{39} + X_{40} - X_{41} - X_{43} = 0$

$X_{36} - X_{37} + X_{38} - X_{39} + X_{40} - X_{41} - X_{42} = 0$

$X_{44} + X_{45} + X_{46} + X_{47} + X_{48} + X_{49} - X_{50} = 0$

$X_{31} + X_{32} + X_{33} - X_{34} = 0$

$X_{27} + X_{28} + X_{29} - X_{30} = 0$

$X_{23} + X_{24} + X_{25} - X_{26} = 0$

$X_{12} + X_{13} + X_{14} + X_{15} - X_{16} = 0$

$X_4 + X_6 + X_8 + X_{10} - X_{11} = 0$

$-X_{15} - X_{21} + X_{22} \leq 0$

$-X_{12} - X_{13} - X_{17} + X_{18} \leq 0$

$X_9 - X_{10} \leq 0$

$X_7 - X_8 \leq 0$

$X_5 - X_6 \leq 0$

$X_3 - X_4 \leq 0$

$-X_1 + X_2 \leq 0.$

11.2.5. Data set E: the EPE data set

Data set E, the EPE (Environmental Protection Expenditures) data set, is a data set from the Swiss Federal Statistical Office[1]. The data set has been used in the EUREDIT project, an international research project on data editing and imputation that was partly funded by the European Commission, for evaluation purposes.

[1] On certain conditions the EPE data set may be obtained from the Swiss Federal Statistical Office.

The data set contains 54 continuous variables X_1 to X_{54} that are involved in edits, and 1,039 records. Of these 1,039 records, 378 are inconsistent and 661 consistent. The data set contains 2,230 missing values, i.e. 5.9 missing values on average per inconsistent record. All variables are non-negative. Besides these non-negativity edits we have the following 21 balance edits.

$$-X_1 - X_5 - X_9 + X_{13} = 0$$
$$-X_2 - X_6 - X_{10} + X_{14} = 0$$
$$-X_3 - X_7 - X_{11} + X_{15} = 0$$
$$-X_4 - X_8 - X_{12} + X_{16} = 0$$
$$-X_{17} - X_{19} - X_{21} + X_{23} = 0$$
$$-X_1 - X_2 - X_3 - X_4 - X_{17} + X_{18} = 0$$
$$-X_5 - X_6 - X_7 - X_8 - X_{19} + X_{20} = 0$$
$$-X_9 - X_{10} - X_{11} - X_{12} - X_{21} + X_{22} = 0$$
$$-X_{18} - X_{20} - X_{22} + X_{24} = 0$$
$$-X_{13} - X_{14} - X_{15} - X_{16} - X_{23} + X_{24} = 0$$
$$-X_{25} - X_{31} + X_{37} = 0$$
$$-X_{26} - X_{32} + X_{38} = 0$$
$$-X_{27} - X_{33} + X_{39} = 0$$
$$-X_{28} - X_{34} + X_{40} = 0$$
$$-X_{29} - X_{35} + X_{41} = 0$$
$$-X_{25} - X_{26} - X_{27} - X_{28} - X_{29} + X_{30} = 0$$
$$-X_{31} - X_{32} - X_{33} - X_{34} - X_{35} + X_{36} = 0$$
$$-X_{37} - X_{38} - X_{39} - X_{40} - X_{41} + X_{42} = 0$$
$$-X_{30} - X_{36} + X_{42} = 0$$
$$-X_{43} - X_{44} - X_{45} - X_{46} - X_{47} + X_{48} = 0$$
$$-X_{49} - X_{50} - X_{51} - X_{52} - X_{53} + X_{54} = 0.$$

11.2.6. Data set F: the ABI data set

Data set F is a subset of the ABI (Annual Business Inquiry) data set that has been used in the EUREDIT project[2]. Data set F consists of the records of the businesses in the ABI data

[2] On certain conditions the ABI data set may be obtained from the Office for National Statistics (UK).

set that have filled in the so-called long questionnaire, have a registered turnover of less than 1,000,000 British pounds, and have at least one employee. The data set contains 1,425 records of which 1,141 are inconsistent, and 284 are consistent. The data set contains 195 missing values, i.e. 0.2 missing values on average per inconsistent record. Data set F contains 26 continuous variables X_1 to X_{26}. All variables, except X_{21} X_{22}, X_{23}, and X_{24}, are non-negative. Besides the non-negativity restrictions, the following edits should be satisfied.

$X_{16} - 0.3\ X_{17} \leq 0$

$X_8 - 0.3\ X_{17} \leq 0$

$-X_1 + X_{24} \leq 0$

$-X_1 + X_{23} \leq 0$

$-X_1 + X_{22} \leq 0$

$-X_1 + X_{21} \leq 0$

$0.25\ X_1 - X_{20} \geq 1$

$X_7 - 0.4\ X_{17} \leq 0$

$X_6 \geq 1$

$X_6 - 60\ X_{25} \leq 0$

$-X_6 + 4\ X_{25} \leq 0$

$-0.1\ X_2 + X_3 \leq 0$

$0.03\ X_2 - X_3 \leq 0$

$X_1 - 2\ X_{26} \leq 0$

$-20\ X_1 + X_{26} \leq 0$

$X_{18} + X_{19} - X_{20} = 0$

$X_7 + X_8 + X_9 + X_{10} + X_{11} + X_{12} + X_{13} + X_{14} + X_{15} + X_{16} - X_{17} = 0$

$X_2 + X_3 + X_4 + X_5 - X_6 = 0.$

11.2.7. Summary of the data sets

In Table 11.1 below we give a summary of the characteristics of the six data sets. In the table the number of variables, the number of non-negativity constraints, the number of edits (excluding the non-negativity constraints), the total number of records, the number of inconsistent records, and the total number of missing values are listed. Besides we present the number of records with more than 6 erroneous fields or missing values. For the purpose of our evaluation study we define these records to be 'highly erroneous' ones. In Section 11.4 we compare the computing time required for the records that are not highly erroneous

to the computing time that is required for all records for two of the evaluated algorithms. Finally, we list the average number of errors per inconsistent record (excluding the missing values) and the average number of optimal solutions per inconsistent record.

The figures in the last two rows of Table 11.1 were hard to establish, because all implemented programs suffer to some extent from numerical problems (see also Section 11.3). They have been obtained by carefully comparing and combining the results of the four programs we have applied to each other. For instance, for each record we have assumed that the best solution determined by any of the four programs is indeed the optimal solution, i.e. the solution with the fewest possible changes. The number of changes in this optimal solution is assumed to be the actual number of errors in the record under consideration. To determine the number of optimal solutions for each inconsistent record we have examined how many optimal solutions are determined by the programs based on vertex generation, non-standard branch-and-bound and cutting planes. If two or three programs found the same number of optimal solutions (this is usually the case), that number is assumed to be the actual number of optimal solutions. In the very rare cases where each program determined a different number of optimal solutions, we have assumed that the highest number is the actual number of optimal solutions.

Table 11.1. Characteristics of the data sets

	Data set A	Data set B	Data set C	Data set D	Data set E	Data set F
Number of variables	90	76	53	51	54	26
Number of non-negativity constraints	90	70	36	49	54	22
Number of edits[1]	8	20	36	15	21	18
Total number of records	4,347	274	1,480	4,217	1,039	1,425
Number of inconsistent records	4,347	157	1,404	2,152	378	1,141
Total number of missing values	259,838	0	0	0	2,230	195
Number of records with more than 6 errors or missing values	4,346	7	117	16	136	8
Average number of errors per inconsistent record[2]	0.2	2.5	2.6	1.6	5.8	3.0
Average number of optimal solutions per inconsistent record	6.1	12.0	6.9	23.3	1.2	11.6

[1] Excluding non-negativity constraints.

[2] Excluding missing values.

The number of variables, edits and records are in most of the six data sets quite realistic. Exceptions are data set A, where the number of edits other than non-negativity edits is very small, and data set B, where the number of records is very small. At Statistics Netherlands, a very large and complex data set to be edited automatically may involve slightly more than 100 variables, about 100 edits, and a few thousand records. These numbers are somewhat higher than for the data sets in Table 11.1, but for such large data sets the value of many variables equals zero. This simplifies the error localisation problem to some extent, for example, because this justifies replacing missing values by zeros in a pre-processing step.

The six data sets used in our evaluation tests were not selected because of certain characteristics they possess, but because either raw (unedited) and clean (manually edited) data were both available, or because we were requested to edit the data set automatically. The availability of both raw and clean data enables us to compare the population figures estimated using on automatically edited data to population figures estimated using manually edited data. This in turn enables us to assess the quality of automatically edited

data. The results of these experiments related to the statistical quality of automatic editing are not reported in this book.

The six data sets come from a wide range of business surveys, namely a survey on labour costs, a structural business survey on enterprises in the photographic sector, a structural business survey on enterprises in the building and construction industry, a structural business survey on the retail sector, and a survey on environmental expenditures. Besides these data sets we have also used the ABI data set. Due to confidentiality reasons the sector to which the businesses in this data set belong has not been made public.

As far as we are able to tell, the six test data sets are not essentially different from other data sets arising in practice. In other words, to the best of our knowledge these data sets are representative for other data sets from business surveys. A good performance on the six data sets hence suggests that the performance on other data sets arising in practice will be acceptable.

This is confirmed by practical experience at Statistics Netherlands, where nowadays almost all annual structural business surveys are treated by a combination of selective editing (for details on this implementation of selective editing see Hoogland, 2002) and automatic editing. For an overview of this approach for annual structural business surveys at Statistics Netherlands we refer to De Jong (2002). Automatic editing for annual structural business surveys is carried out by means of SLICE (version 1.0), which is based on the vertex generation approach of Chapter 5. Because extensive use of time-consuming COM-components is made in the software architecture of SLICE, the computing times are of a higher order than those mentioned in Tables 11.2 and 11.3 in Section 11.4 below. Nevertheless, all involved structural business surveys can be treated by SLICE 1.0 within a reasonable amount of time. Obviously, computing times vary over data sets of different surveys, but no data set with an exceedingly high computing time has been encountered so far. Our practical experience hence suggests that computational results for our test data sets can be carried over to other business data sets.

11.3. Implementation of the algorithms

The four algorithms we examine in this chapter have been implemented in four computer programs. We briefly discuss the implementation details of these programs in this section. The first algorithm, based on a standard MIP formulation (see Chapter 3), we consider has been implemented by Van Riessen (Van Riessen, 2002), a student at the Hogeschool van Amsterdam (College of Amsterdam), while doing an internship at Statistics Netherlands. This algorithm has been implemented in Visual C++ 6.0, and calls routines of ILOG CPLEX (version 7.5), a well-known commercial MIP-solver, to actually solve the MIP problems involved (see *ILOG CPLEX 7.5 Reference Manual*, 2001). We refer to Van Riessen's program as ERR_CPLEX in the remainder of this chapter.

In contrast to all other error localisation programs we consider in this chapter, ERR_CPLEX finds only one optimal solution to each instance of the error localisation problem. To find all optimal solutions we could – once an optimal solution to the current MIP problem has been determined – iteratively add an additional constraint, which basically states that the present optimal solution is excluded but other optimal solutions to the current MIP problem remain feasible, and solve the new MIP problem. This process of

determining an optimal solution to the current MIP problem and adding an additional constraint to obtain a new MIP problem goes on until all optimal solutions to the error localisation problem have been found. We have not implemented this option, however. Resolving the problem from scratch for each solution would be very time-consuming. The alternative is to use a hot restart, where information generated to obtain an optimal solution to an MIP problem is utilised to obtain an optimal solution to a slightly modified MIP problem quickly. A problem with this possibility is that experiences at Statistics Netherlands with ILOG CPLEX so far, on linear programming (LP) problems arising in statistical disclosure control, show that ILOG CPLEX becomes numerical unstable if too many hot restarts in a row are applied.

The results of ERR_CPLEX are therefore only indicative. If the algorithms we have developed ourselves were clearly outperformed by ERR_CPLEX, this would suggest that standard MIP-solvers may be preferable to our algorithms. In that case, further studies with an extended version of ERR_CPLEX that aims to find all optimal solutions to the error localisation problem instead of only one would still be needed, however.

ERR_CPLEX, or more precise the MIP-solver of ILOG CPLEX, suffers from some numerical problems. These problems arise because in (erroneous) records the largest values may be a factor 10^9 or more larger than the smallest values. Due to these numerical problems ERR_CPLEX occasionally generates suboptimal solutions containing too many variables. In some other cases it does not find a solution at all. The value of M (see (3.33) to (3.36) in Section 3.5) in ERR_CPLEX was set to 10,000. Lower values for M led to too many incorrect results, whereas higher values for M led to additional numerical problems.

The second algorithm, based on vertex generation (see Chapter 5), has been implemented by the author of this book. This program, CherryPi, has been developed in Delphi 3. The implemented algorithm is the adapted version of Chernikova's algorithm described in Sections 5.3 to 5.5 and Section 5.9 of this book. All mentioned improvements due to Rubin (1975 and 1977), Sande (1978a), Schiopu-Kratina and Kovar (1989), and Fillion and Schiopu-Kratina (1993) on the original algorithm by Chernikova (1964 and 1965) have been implemented in CherryPi. The possibly more efficient algorithm due to Duffin has not been implemented.

The adapted version of Chernikova's algorithm uses a matrix to solve the error localisation problem (see Sections 5.3 to 5.5 and Section 5.9 for more details regarding this matrix). The number of rows of this matrix is implied by the number of edits and the number of variables. The number of columns is determined dynamically. Due to memory and speed restrictions a maximum for the allowed number of columns is set in CherryPi. If the actual number of columns exceeds the allowed maximum, certain columns are deleted. This influences the solutions that are found by CherryPi. Due to this pragmatic rule in some cases only non-optimal solutions may be found, and – even worse – in some other cases no solutions at all may be found. Another effect of this pragmatic rule is that if columns have been deleted in order to arrive at solutions to an instance of the error localisation problem, the optimality of the found solutions is not guaranteed. The higher the allowed number of columns, the better the quality of the solutions found by CherryPi, but also the slower the speed of the program. Practical experience has taught us that in many instances setting the allowed number of columns to 4,000 gives an acceptable trade-off between the quality of the found solutions and the computing time of the program. In the version of CherryPi that

was used for the comparison study the allowed number of columns was therefore set to 4,000. Besides the above-mentioned memory problems, CherryPi occasionally suffers from numerical problems, for the same reason as ERR_CPLEX.

The third algorithm, based on a non-standard branch-and-bound approach (see Chapter 8), has originally been implemented by Quere, a post-graduate student at Eindhoven University of Technology, while doing an internship at Statistics Netherlands. At Statistics Netherlands Quere was supervised by the author of this book. Later the prototype computer program, Leo, was modified by mainly Van den Broeke, and to a lesser extent by the author of this book. Leo has been developed in Delphi 3. The program requires that a maximum cardinality for the optimal solutions must be specified beforehand. Only optimal solutions with at most the specified maximum cardinality are determined.

In Leo the following rule to select a branching variable has been implemented: first select the variables that are involved in at least one failed edit and a minimum number of satisfied edits, and then select the variable from this set of variables that occurs most often in the failed edits. If there are several 'best' variables to branch on, one of them is chosen randomly. Considering that in Leo variables are first eliminated and later fixed, i.e. the opposite order in comparison to Daalmans' implementation of a similar algorithm for categorical data (see Section 8.6.4 and Daalmans, 2000), Daalmans' work suggests that Leo's method of selecting the branching variable should be quite reasonable. Other orders of treating variables in Leo remain to be tested.

In Leo, the equality-elimination rule described at the end of Section 8.2 has not been implemented. On two data sets, the data sets for which the computing times of Leo are comparatively bad, we have applied a special, alternative version of Leo in which the equality-elimination rule has been implemented.

Leo sometimes suffers from memory problems, especially for records with many errors, because too many nodes with too many edits need to be stored. For records for which Leo suffers from memory problems, it cannot determine an optimal solution. Leo occasionally suffers from numerical problems, for the same reason as ERR_CPLEX and CherryPi.

The fourth and last algorithm, based on the cutting plane algorithm of Section 10.5, has been implemented by Coutinho, while working temporarily at Statistics Netherlands. Coutinho was supervised by the author of this book. We refer to Coutinho's program as CUTTING in this chapter. CUTTING has been developed in Delphi 3. In this implementation a modified set-covering problem (see Ragsdale and McKeown (1996) and Section 10.5 of this book) instead of an ordinary set-covering problem is generated and solved in each iteration. In each iteration CUTTING determines all optimal solutions to the corresponding modified set-covering problem. A fundamental part of the program is a solver for modified set-covering problems. Using well-known ideas from literature, we have developed this solver ourselves based on a recursive branch-and-bound algorithm. We did not spend much time on optimising the performance of this solver. It may, therefore, be improved upon. CUTTING can, if desired, determine only optimal solutions up to a user-specified maximum cardinality. The program can also work without such a maximum cardinality. Like Leo, CUTTING suffers from memory problems for some records. For such records, it cannot determine an optimal solution. CUTTING occasionally suffers from numerical problems, for the same reason as the other three programs.

The computing times of ERR_CPLEX and CherryPi may possibly be improved upon if we include a restriction on the number of variables that may be changed, like we do for Leo and CUTTING. The stricter this restriction, the faster each program is likely to be. Including such a restriction in CherryPi will probably have less effect than for Leo and CUTTING, because the search process of CherryPi is based on manipulating edits rather than on treating variables directly. The effect of including a restriction on the number of variables that may be changed in ERR_CPLEX is not entirely clear. On the one hand, in order to include such a restriction an additional integer constraint would be required, which would slightly increase the computing time. On the other hand, the search process would be shortened because certain possible solutions would not have to be examined. Considering the two opposite effects, we expect that including a restriction on the number of variables that may be changed in ERR_CPLEX leads to a reduced computing time, but this remains to be tested.

We could use a restriction on the number of variables that may be changed to iteratively check the records: first we try to solve the error localisation problem for all inconsistent records with the allowed maximum number of errors set to one, for the remaining records we then try to solve the error localisation problem with the allowed maximum number of errors set to two, etc. until the error localisation problem for each record has been solved. This approach is likely to lead to a reduced computing time. We have not implemented this approach for any of the four programs, however.

An important aspect in the evaluation of an algorithm is the time required to implement it in a computer program. The easiest algorithm/program to implement is ERR_CPLEX. The Visual C++ program only has to transform data and user-specified metadata, such as edits, into optimisation problems in a format that can be interpreted by ILOG CPLEX. To solve these optimisation problems routines from ILOG CPLEX are used. A bit more complicated is CUTTING. The two most important steps are the elimination of variables and solving modified set-covering problems. Both steps are actually quite simple to implement. Implementing CUTTING required about two months for a non-professional programmer. Slightly more complicated is Leo as this involves implementing a recursive algorithm, which is difficult to debug. The most complicated program to implement is CherryPi as several “tricks” (see Chapter 5) need to be implemented in order to make this program sufficiently fast. To implement CherryPi about three to four months were required for a non-professional programmer.

11.4. Computational results

For Leo and CUTTING we have performed two kinds of experiments per data set. In the first kind of experiments we have set the maximum cardinality N_{max} to 6. For many realistic data sets setting N_{max} to 6 is a good option as for records containing more than 6 errors it is unlikely that automatic error localisation will lead to data of sufficiently high statistical quality. Possible exceptions are data sets that contain many missing values, such as data set A. In the second kind of experiments for Leo we have set N_{max} as high as possible without encountering memory problems for many, i.e. 20 or more records. In the second kind of experiments for CUTTING we have removed a maximum cardinality all together. For ERR_CPLEX and CherryPi we have only performed experiments without a specified maximum cardinality.

The experiments have been performed on a 1500 MHz PC with 256 MB of RAM. This PC is connected to a local area network. Computing times may therefore be influenced by the amount of data that was transmitted through the network at the time of the experiments. To reduce and to estimate this influence we have performed five experiments per data set at various moments during the day. In Table 11.2 we have mentioned the average computing times of these experiments for the entire data sets, and between brackets the standard deviation of these computing times over the corresponding five experiments. Note that some programs, such as Leo, have a random aspect that also influences the computing time. This random aspect is reflected in a relatively high standard deviation. In all experiments, the reliability weight of each variable was set to 1.

Table 11.2. Average computing times of the error localisation algorithms for entire data sets (between brackets the standard deviation of these computing times)

	Data set A	Data set B	Data set C	Data set D	Data set E	Data set F
ERR_CPLEX[1]	233 (1)	10 (0)	86 (1)	93 (9)	13 (0)	35 (0)
CherryPi	570 (38)	96 (1)	540 (7)	498 (30)	622 (3)	79 (0)
CUTTING	601 (17)	513 (12)	1913 (7)	1101 (20)	90 (1)	94 (2)
CUTTING ($N_{max} = 6$)	156 (10)	395 (23)	695 (31)	1036 (137)	50 (2)	92 (5)
Leo[2]	18 (0)	308 (10)	531 (4)	21 (1)	59 (34)	7 (0)
Leo ($N_{max} = 6$)	7 (0)	51 (1)	94 (2)	19 (0)	4 (1)	8 (1)

[1] These tests were performed on a special server. On this PC the only fully licensed version of ILOG CPLEX at Statistics Netherlands has been installed. For comparison reasons we have also used CherryPi for data set A on this machine. The average computing time for data set A on this machine is 528 seconds (compared to 570 seconds on the usual PC) with a standard deviation of 0 seconds. To compare the computing times of ERR_CPLEX to those of the other programs, we have therefore multiplied the original computing times on the special server by a factor of 570/528 = 1.08.

[2] To find the results for Leo for $N_{max} > 6$, we have set N_{max} equal to 90 for data set A, to 8 for data sets B and C, and to 12 for data sets D, E and F.

Data set F contains only 8 records for which 6 or more changes are required. The computing times of Leo and CUTTING are therefore almost equal to the computing times of Leo_6, respectively CUTTING_6 (i.e. Leo, respectively CUTTING with $N_{max} = 6$). In fact, due to the stochastic variability in the computing times Leo even outperformed Leo_6 in our experiments. Taking the standard deviation of the experiments into account, Leo and Leo_6 are about equally fast.

The computing time of Leo_8 for data set B was high due to one record. On average Leo_8 spent 242 seconds (of the total of 308 seconds on the average) to conclude that it could not find a solution for this record. Even when we set the maximum to 10, Leo_10 could not

find a solution. It took Leo_10 1,398 seconds to come to this conclusion. According to ERR_CPLEX this particular record contains 11 errors.

In Table 11.3 below we present for each data set the average computing time per processed record, i.e. the corresponding number of Table 11.2 divided by the total number of processed records. Between brackets we mention the average computing time per processed erroneous record (assuming that it does not take any computing time to process the consistent records), i.e. the corresponding number of Table 11.2 divided by the total number of processed erroneous records. Records that require too much computer memory for Leo are excluded from the results for Leo. For data sets for which such records occur, the average for Leo is hence taken over fewer records than for the other programs. This introduces some bias in favour of Leo, as such records are usually rather time-consuming to solve.

Table 11.3. Average computing times of the error localisation algorithms per erroneous record in milliseconds (between brackets the average computing time per processed erroneous record)

	Data set A	Data set B	Data set C	Data set D	Data set E	Data set F
ERR_CPLEX[1]	54 (54)	36 (57)	53 (66)	26 (50)	13 (35)	24 (30)
CherryPi	131 (131)	350 (611)	365 (385)	118 (231)	599 (1,646)	55 (69)
CUTTING	138 (138)	1,872 (3,268)	1,293 (1,363)	261 (512)	87 (238)	66 (82)
CUTTING ($N_{max} = 6$)	36 (36)	1,442 (2,516)	470 (495)	246 (481)	48 (132)	65 (81)
Leo[2]	4 (4)	1,124 (1,962)	361[3] (381[4])	5 (10)	58[5] (162[6])	5 (6)
Leo ($N_{max} = 6$)	2 (2)	186 (325)	64 (67)	5 (9)	4 (11)	6 (7)

[1] Tests performed on a special server. To compare the computing times of ERR_CPLEX to those of the other programs, they have been multiplied by a factor of 1.08.

[2] To find the results for Leo for $N_{max} > 6$, we have set N_{max} equal to 90 for data set A, to 8 for data sets B and C, and to 12 for data sets D, E and F.

[3] Average taken over 1,479 processed records (averages of the other programs taken over 1,480 processed records).

[4] Average taken over 1,395 processed erroneous records (averages of the other programs taken over 1,404 processed erroneous records).

[5] Average taken over 1,026 processed records (averages of the other programs taken over 1,039 processed records).

[6] Average taken over 365 processed erroneous records (averages of the other programs taken over 378 processed erroneous records).

Due to numerical and memory problems, the programs could not always determine solutions. None of the programs can guarantee to find (all) optimal solutions for all records. For ERR_CPLEX, CherryPi, Leo, and CUTTING we have listed in Table 11.4 below for each data set the number of records for which these programs could not determine solutions to the error localisation problem. For all data sets, Leo_6 and CUTTING_6 found all optimal solutions for all records requiring 6 or less changes. Especially for data set A, this was very easy as there is only one record in data set A that has 6 or fewer errors or missing values (see Table 11.1)

Table 11.4. Number of records for which no solution could be found

	Data set A	Data set B	Data set C	Data set D	Data set E	Data set F
ERR_CPLEX	3	0	0	0	0	0
CherryPi	0	2	11	8	2	7
CUTTING	0	1	93	0	0	2
Leo[1]	0	1	58	0	53	0

[1] To find the results for Leo, we have set N_{max} equal to 90 for data set A, to 8 for data sets B and C, and to 12 for data sets D, E and F.

For 9 of the 58 records of data set C for which Leo_8 could not find a solution and for 13 of the 53 records of data set E for which Leo_12 could not find a solution, Leo suffered from memory problems. Those 9, respectively 13 records were excluded from the computational results for Leo in Tables 11.2 and 11.3 above. As far as we have been able to determine, excluding these records from the computational results does not have a large effect and does not change the overall picture. Leo_8, respectively Leo_12, could not find solutions for the other records referred to in Table 11.4, because more than 8, respectively 12, changes were required.

Comparing the evaluation results of the various programs to each other is a complex task. If we rank the algorithms according to their computing times, and compare ERR_CPLEX, CherryPi, Leo (with $N_{max} > 6$) and CUTTING with each other we see that ERR_CPLEX performs best for 3 out of the 6 data sets and second best for the other 3 data sets. Leo (with $N_{max} > 6$) performs best for 3 out of 6 data sets and second best for 2 data sets. So, one might conclude that – purely looking at of the computing times – ERR_CPLEX is slightly better than Leo. Clearly worst is CUTTING.

Now, if we compare the versions of Leo and CUTTING with $N_{max} = 6$ (i.e. Leo_6 and CUTTING_6) to ERR_CPLEX and CherryPi, and again rank the programs according to their computing times, we see that ERR_CPLEX performs best for 2 out of the 6 data sets and second best for 3 data sets. Leo_6 performs best for 4 out of 6 data sets and second best for the other 2 data sets. Here one might conclude that – purely looking at of the computing times – Leo_6 is better than ERR_CPLEX. The performances of CherryPi and CUTTING_6 are about equally good.

As we already mentioned in Section 11.3, the equality-elimination rule described at the end of Section 8.2 has not been implemented in Leo. For the two data sets for which ERR_CPLEX is faster than Leo_6, data sets B and C, we have applied a special version of Leo in which this rule has been implemented. The results are given in Table 11.5. In this table we have mentioned the average computing times for the entire data sets, and between brackets the standard deviation of these computing times.

Table 11.5. Average computing times for Leo with equality-elimination rule (between brackets the standard deviation of these computing times)

	Data set B	Data set C
Leo[1]	308 (10)	531 (4)
Leo with equality-elimination[1]	14 (2)	77 (1)
Leo ($N_{max} = 6$)	51 (1)	94 (2)
Leo with equality-elimination ($N_{max} = 6$)	4 (1)	19 (1)

[1] To find the results for Leo (both with and without the equality-elimination rule) for $N_{max} > 6$, we have set N_{max} equal to 8 for both data sets.

For data sets B and C, we present in Table 11.6 the average computing time per processed record for the special version of Leo with equality-elimination, and between brackets the average computing time per processed erroneous record (assuming that it does not take any computing time to process the consistent records). As no records require too much computer memory for the version of Leo with equality-elimination the averages are taken over all records, respectively over all erroneous records.

Table 11.6. Average computing times for Leo with equality-elimination rule per erroneous record in milliseconds (between brackets the average computing time per processed erroneous record)

	Data set B	Data set C
Leo[1]	1,124 (1,962)	361[2] (381[3])
Leo with equality-elimination[1]	51 (89)	52 (55)
Leo ($N_{max} = 6$)	186 (325)	64 (67)
Leo with equality-elimination ($N_{max} = 6$)	15 (25)	13 (14)

[1] To find the results for Leo (both with and without the equality-elimination rule) for $N_{max} > 6$, we have set N_{max} equal to 8 for both data sets.

[2] Average taken over 1,479 processed records (averages of the other versions taken over 1,480 processed records).

[3] Average taken over 1,395 processed erroneous records (averages of the other versions taken over 1,404 processed erroneous records).

For data set B the version of Leo_8 with equality-elimination could not find a solution for one record, the one record that requires 11 changes. For data set C the version of Leo_8 with equality-elimination could not find optimal solutions for 58 records, just like the

standard version of Leo. The version of Leo with the equality-elimination rule did not suffer from memory problems, however.

Examining the results of Tables 11.2, 11.3, 11.5 and 11.6, we can conclude that as far as computing speed is concerned ERR_CPLEX and Leo (either with $N_{max} = 6$ or with $N_{max} > 6$) are the best programs. We note at the same time, however, that this conclusion is not completely justified as ERR_CPLEX determines only one optimal solution whereas the other programs (aim to) determine all optimal solutions.

Comparing the results of Tables 11.5 and 11.6 to Tables 11.2 and 11.3 we see that the equality-elimination rule described at the end of Section 8.2 leads to a substantial reduction in computing time, at least for the two data sets examined. With this rule, Leo_6 is clearly faster than ERR_CPLEX for all data sets.

One might expect ERR_CPLEX to be relatively fast if there are many optimal solutions per record on the average as this program only determines one solution per record, whereas the other algorithms determine (many) more. Surprisingly, this is not the case. For instance, for data set D, with 23.3 optimal solutions per erroneous record on the average, ERR_CPLEX is relatively slow. For data set C with 'only' 6.9 optimal solutions per erroneous record on the average, ERR_CPLEX is relatively fast.

Besides computing speed other aspects are, of course, important too. We note that all programs, even the commercially available ILOG CPLEX, suffer from numerical problems. In addition, Leo sometimes suffers from memory problems. Due to its matrix with a fixed maximum number of columns, CherryPi does not always determine optimal solutions. Instead, it sometimes determines a less good, suboptimal solution. Summarising, it is hard to give a verdict on the quality of the solutions found by the programs as the programs suffer from a diversity of problems.

11.5. Discussion

McKeown (1981), in the context of Special Transportation Problems and Pure Fixed Charge Transportation Problems, remarks that 'It is unclear in any of these contexts as to what makes a problem "easy" or "difficult" to solve'. This remark has again been confirmed in the context of the error localisation problem. From the characteristics of the data sets it is hard to establish beforehand whether the corresponding instances of the error localisation problem will be "easy" or "hard". We can even extend the remark of McKeown to the following: it is unclear in our context as to what makes an algorithm a "good" or "bad" one. All algorithms we have examined have their good and bad aspects. In the end, the algorithm one favours is to some extent a subjective choice.

From our own developed algorithms, we consider the branch-and-bound algorithm described in Chapter 8 the most promising one for solving the error localisation problem. The main reason for our choice is the excellent performance of Leo for records with up to 6 errors. For such records it determines all optimal solutions very fast. We admit that for records with more than 6 errors the results of Leo become less good, just like the other algorithms. The program begins to suffer from memory problems, and the computing time increases. To some extent these problems can be overcome by implementing the equality-elimination rule described at the end of Section 8.2. Besides, as we argued before in this

book, we feel that records with many errors should not be edited in an automatic manner, but in a manual manner. That is, we feel that records with more than, say, 6 errors should be rejected for automatic editing. Given this point of view, Leo seems to be an excellent choice.

In addition, it is not very complex to extend the branch-and-bound algorithm of Leo to a mix of categorical and continuous data. Statistics Netherlands has therefore decided to implement this algorithm in a module of version 1.5 (and future versions) of the SLICE system (see e.g. De Waal, 2001b). This version reads an upper bound for the number of missing values per record as well as a separate upper bound for the number of errors (excluding missing values) per record. The former number is allowed to be quite high, say 50 or more, whereas the latter number is allowed to be moderate, say 10 or less. If the number of missing values or the number of errors (excluding missing values) in a record exceeds either of these upper bounds, this record is rejected for automatic editing. The new module is suitable for a mix of categorical and continuous data, and includes the equality-elimination rule. In addition, it contains a heuristic to handle integer data based on the methodology of Chapter 9. The new module replaces the CherryPi-module, based on vertex generation, of SLICE 1.0.

One may argue that some users of SLICE will want to edit records with many erroneous fields, say 10 or more, automatically despite our arguments against editing such extremely contaminated records. Such users might then be disappointed, because the new module is not able to handle such records. To overcome this problem, we propose to opt for a simple heuristic treatment of these extremely erroneous records instead of applying the new module.

There are many simple heuristic treatments possible. As an example we mention how we were able to handle the 53 'difficult' records of data set E by means of Leo. To edit these 53 difficult records, i.e. the 13 records that required too much computer memory and the 40 records for which no solutions involving 12 or less variables exist, by means of Leo we first replaced the missing values by zero. This is quite a standard action as most missing values in this data set should equal zero in any case. Subsequently, we tried to find all solutions for these adapted records that involve at most 8 variables. Over 5 experiments, this took 40 seconds on the average for the entire set of 53 records (with a standard deviation of 13 seconds). For only one record we were unable to find a solution at all by means of Leo.

Another simple possibility is to split the set of edits into two subsets. First, we can apply the branch-and-bound algorithm on one of these subsets. One of the optimal solutions for this subset is chosen, and the corresponding fields are set to missing. Subsequently, we apply the branch-and-bound algorithm on the newly created record with possibly some additional missing values in comparison to the original record, using all edits. The solutions obtained in this way are, possibly suboptimal, solutions to the error localisation problem for the original record. This approach utilises the fact that the branch-and-bound algorithm works quite well for records with missing values.

A similar approach, which utilises the same fact, for solving the error localisation problem for a record with many errors is to first determine a number of implausible values in a heuristic manner. These implausible values are set to missing. Subsequently, we apply the

branch-and-bound algorithm on the newly created record with some additional missing values in comparison to the original record. The solutions obtained in this way are again, possibly suboptimal, solutions to the error localisation problem for the original record. Chung, a trainee at Statistics Netherlands, has studied this heuristic approach to solving extremely erroneous records (see Chung, 2003).

Finally, for records containing many errors one could resort to solving an LP approximation for the error localisation problem. This LP approximation is described in Section 13.2. To determine extremely erroneous records beforehand, Van der Laar at Statistics Netherlands has developed algorithms and software. The accuracy of these algorithms remains to be tested, however.

All in all we are confident that records with many errors do not pose a threat for us if we apply the branch-and-bound algorithm in practice. The methodology by Van der Laar (hopefully) allows us to identify the extremely erroneous records, which can then be treated by any of the above-mentioned heuristics. We are willing to admit that our choice for the branch-and-bound algorithm is to some extent a subjective choice, but we feel that our choice is a justifiable one.

12. Imputation

12.1. Introduction

Imputation is the process of estimating missing values and filling in these estimates in the data set. The "holes" in the data set are filled in this way. These holes may be due to data there were originally missing in the data set, or due to values that were set to missing because they were considered implausible during the error localisation phase.

Imputation is a very important topic for both statistical offices and universities. In the last two decades a lot of scientific research has been devoted to imputation. In the present chapter we restrict ourselves to discussing the current imputation module of SLICE, the general software framework for editing and imputation that Statistics Netherlands is developing (see De Waal and Wings, 1999; De Waal, 2000d and 2001b), and an imputation program, WAID, that has been developed within a European project. The author of this book was overall co-ordinator of this project. The reasons for limiting the discussion on imputation methods to only this module of SLICE and WAID is that, first, too many imputation methods have been developed to discuss them all in this chapter, and, second, the author's personal involvement with the mentioned module and program. The imputation module of SLICE is discussed in Section 12.2. The imputation program WAID is discussed in Section 12.3. That latter section is based on De Waal (2001c). For two overview papers on imputation techniques we refer to Kalton and Kasprzyk (1986) and Kovar and Whitridge (1995). For an overview of imputation software we refer to Hox (1999), and Chambers et al. (2001b). For a brief overview of imputation methods that are applied at Statistics Netherlands see De Waal (2000e).

The imputation module of SLICE mentioned above uses a so-called regression imputation method to impute for missing data (see Section 12.2 for more details), and WAID a so-called donor imputation method (see Section 12.3). Taking the edits into account with such imputation methods is a non-trivial matter. For simplicity, edits are therefore usually not taken into account while imputing. So, after imputation the edits may still be violated. However, if the (generalised) Fellegi-Holt paradigm has been used to localise the errors in the data, we know that it is possible to satisfy all edits by changing the values of the fields that were identified as being erroneous. Therefore, after the imputation step we slightly modify the imputed values in such a way that all edits become satisfied. Note that if we apply the (generalised) Fellegi-Holt paradigm to localise the errors, we only have to modify the imputed values. The not imputed, original values, which were considered correct during the error localisation phase, do not have to be modified. As we assume that the imputations are carried out according to a "good" statistical model, which implies that an imputed record will generally be of acceptable quality from a statistical point of view, we aim to change the imputed values as little as possible while making sure that the resultant record passes all edits. This problem of obtaining consistent records is explored in Section 12.4, which is based on De Waal (2001a). We propose an algorithm, similar to the algorithm proposed in Chapters 8 and 9, for solving the problem. We have developed the

basic ideas of the algorithm. Later a post-graduate student from Delft University of Technology, Kartika, implemented the algorithm and in the course of that work filled in many implementation details. Section 12.5 concludes this chapter with a brief discussion.

12.2. Regression imputation in SLICE

The imputation module of the current version of SLICE (version 1.5) uses regression imputation to impute for missing values. For each variable a statistical analyst has to select a number of predictor variables. A predictor variable is a variable that is used to estimate the missing values of the variable under consideration. The statistical analyst can also indicate whether he would like to use an intercept, i.e. a constant term, in the regression equation or not. Suppose the statistical analyst indicates that a constant term and s predictor variables should be used. The missing value of the variable under consideration Y in a certain record r is then estimated by

$$y_r = \alpha + \beta_1 x_1 + ... + \beta_s x_s , \qquad (12.1)$$

where x_i (i=1,…,s) indicates the value of the i-th predictor variable, β_i (i=1,…,s) the value of the i-th regression coefficient, and α the intercept. The underlying regression model is given by

$$y_r = \alpha + \beta_1 x_1 + ... + \beta_s x_s + e_r , \qquad (12.2)$$

where e_r is an unobservable random variable.

If at most one predictor variable has been specified, and possibly also a constant term, the imputation module of SLICE can automatically compute the value of regression coefficient (and the value of the constant term). To do this, the module uses the records in a reference data set for which we have both the value of the variable to be imputed as well as the value of the predictor variable. The module then maximises the likelihood of the regression coefficient and the intercept α, under the assumption that e_r is normally distributed.

If two or more predictor variables have been specified, the statistical analyst has to specify the values of regression coefficients and the intercept himself.

12.3. WAID

Together with University of Southampton, Office for National Statistics, Statistics Finland and Instituto Nacional de Estatística de Portugal, Statistics Netherlands has participated in a European project called AUTIMP. This project was partly financed by the 4th Framework Programme of the European Commission.

One of the aims of the AUTIMP project was the development of software for automatic imputation. The developed prototype imputation software is a stand-alone program, and can be used under Windows 95/98 and Windows NT. The software is based on automatic interaction detection (AID) trees (cf. Sonquist, Baker and Morgan, 1971). Because the developed algorithm gives lower weights to outliers while constructing the tree, the

technique is referred to as weighted automatic interaction detection (WAID). The developed imputation software is also called WAID.

A WAID-tree, or generally a tree-based model, classifies the data in terms of the values of a set of categorical predictor variables. It is a binary tree that is generated by successively splitting a "training" data set into smaller subsets. These subsets are increasingly more homogeneous with respect to a selected response variable. This response variable may be either categorical or numerical. The process of recursively splitting the data set into two subsets continues until a stopping criterion is met. The terminal nodes in this tree form homogeneous clusters.

In the WAID program, homogeneity of a cluster can be measured in several ways. If the response variable is categorical, homogeneity is measured by the so-called Gini index. If the response variable is numerical, homogeneity can be measured by OLS (Ordinary Least Squares), or by outlier robust measures. These outlier robust measures are: Tukey's BiWeight, Huber's Min/Max, and Andrew's Sine. See the report by Tsai and Chambers in Chambers et al. (2001a) for more details on these homogeneity measures.

The developed imputation software can impute for missing values of both categorical and numerical variables. Several missing categorical values may be imputed simultaneously. Technically, a single compound categorical variable based on the categorical variables involved in this missing data pattern is then constructed. A numerical variable cannot be imputed simultaneously with other variables (neither with numerical ones nor with categorical ones).

To use the WAID methodology to impute for missing values in a data set, the software first determines missing data patterns in this data set. For each of these missing data patterns, or parts of missing data patterns, the user has to select a set of categorical predictor variables.

Next, for each (part of a) missing data pattern, WAID-trees are grown using a complete "training" data set. This "training" data set may be the subset of complete records of the data set to be imputed, but it may also be a different data set. The terminal nodes of the generated WAID-trees form clusters of records that are as homogeneous as possible with respect to the variables involved in this (part of a) missing data pattern. These homogeneous clusters themselves are, however, not used by the computer program. Only the classification rules that define these homogenous clusters are used. In this way, we can use a "training" data set to determine the classification rules, and later use another data set with donor records to actually impute for missing values.

After generation of the WAID-trees, and hence generation of the classification rules for constructing homogeneous clusters of records, the data set with missing values is again supplied to the computer program. We also supply a data set with donor records to the computer program. This data set may be the same as the data set to be imputed, but it may also be a different data set. In the data set with donor records we apply the generated classification rules to construct homogeneous clusters of donor records.

To impute for missing values in a certain record, we determine which WAID-trees correspond to the missing data pattern of this record. More than one WAID-tree, and hence more than one homogeneous cluster, may correspond to a particular record, because

separate WAID-trees may have been generated for different parts of its missing data pattern. Subsequently, we determine the homogeneous clusters corresponding to this record by using the classification rules to classify the values of the predictor variables. The records in the data set with donor records corresponding to those clusters are used to impute for the missing data in the record under consideration.

The imputation software supports several imputation methods, for example, the nearest neighbour or a random donor record may be selected from a homogeneous cluster. For more information about the developed imputation software we refer to De Waal (2001c), and to the manual of WAID (see De Waal, De Waard and Plomp, 2001). For more information about the underlying methodology we refer to the report by Tsai and Chambers in Chambers et al. (2001a).

The developed software has been tested extensively. We refer to Chambers et al. (2001a) for evaluation reports. The main conclusion that can be drawn from these reports is that the WAID methodology generally leads to imputed data sets of acceptable quality.

12.4. Consistent imputation

As we already described in the introduction to this chapter, automatic edit and imputation can be carried out in a number of subsequent steps. During the error localisation phase firstly all optimal solutions to the error localisation problem are determined. Subsequently, one optimal solution is selected using a secondary criterion. The values of the variables involved in this solution are set to missing.

After the error localisation phase the missing values (both the values that were missing in the original record and the values that have been set to missing in the error localisation phase) are imputed. In this imputation step imputation methods can be used that preserve the statistical properties as well as possible. During this step the edits are not taken into account. As a result some edits may still be failed.

Finally, the *imputed* values are modified slightly such that all edits become satisfied. The non-imputed, original values are not modified. Because we try to keep the final values as close as possible to the imputed values, we hope that the resulting, consistent, record preserve the statistical properties of the data as well as possible.

The problem we have described in the paragraph above is the so-called *consistent imputation* problem. This is a non-trivial problem, because although the number of suspect fields may be small, the number of missing values in the original record may be high.

12.4.1. Formulation of the consistent imputation problem

If an imputed record, i.e. a record after the imputation step, does not satisfy an edit given by

$$\text{IF} \quad v_i \in F_i^j \text{ for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}, \qquad (12.3a)$$

or

$$\text{IF} \quad v_i \in F_i^j \text{ for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}, \tag{12.3b}$$

its *imputed* values have to be modified in such a way that (12.3) becomes satisfied. The resulting record should be as close as possible to the imputed record. We assume that the imputed values can indeed be modified in such a way that a consistent record results. This is, for instance, the case if the implausible values in a record are determined according to the Fellegi-Holt paradigm, these values are subsequently set to missing, and the missing values – both the original missing values as well as the values that were considered implausible – are imputed.

To measure how close a record is to another record, a suitable distance function has to be defined. In this book we will consider distance functions of the type

$$\sum_{i=1}^{m} w_i a(v_i, \tilde{v}_i) + \sum_{i=1}^{n} w_{m+i} \mid x_i - \tilde{x}_i \mid, \tag{12.4}$$

where the record after the imputation phase is given by $(v_1,\ldots,v_m,x_1,\ldots,x_n)$, the final record by $(\tilde{v}_1,\ldots,\tilde{v}_m,\tilde{x}_1,\ldots,\tilde{x}_n)$, the w_i are user-specified weights, and $a(v_i,\tilde{v}_i)$ is a non-negative matrix satisfying $a(v_i,\tilde{v}_i)=0$ if $v_i=\tilde{v}_i$. Note that $\tilde{v}_i = v_i$ and $\tilde{x}_i = x_i$ for variables (categorical and numerical, respectively) that have not been imputed in the imputation step, because we only modify the imputed values.

Note that for purely continuous data (12.4) reduces to

$$\sum_{i=1}^{n} w_i \mid x_i - \tilde{x}_i \mid. \tag{12.5}$$

The consistent imputation problem can be formulated as

> Minimise (12.4) by modifying the imputed values so that (12.3) becomes satisfied for all edits $j = 1,\ldots,J$, x_i is integer for $i \in I$, and the remaining x_i are continuous,

where I denotes the index set of the integer variables.

12.4.2. A heuristic algorithm

The problem of minimising (12.4) subject to the constraint that all edits (12.3) become satisfied while all integer variables indeed attain integer values can be formulated as a mixed integer programming problem (see Kartika, 2001). This mixed integer programming

problem may be solved by using standard software. Unfortunately, this mixed integer programming problem is usually rather large, because for each category of each involved categorical variable a 0-1 variable is required. Solving the integer programming problem by means of standard mixed integer programming software is therefore likely to be rather time-consuming, especially during application in the day-to-day routine at a statistical office.

In this book we will not make an attempt to solve the consistent imputation problem to optimality, and restrict ourselves to describing a heuristic that is likely to give acceptable results in practice. In any case the heuristic will lead to consistent data that satisfy all edits.

Given a set of variables that have been imputed S, we first fill in the original values for all other variables in the set of explicit edits. This leads to a set of reduced edits involving only the imputed variables. We eliminate the imputed variables in S from the set of reduced edits by applying the algorithm described in Section 9.5. Because we assume that the variables in S can indeed be modified in such a way that the resulting record passes all edits, at least one of the subproblems, which may, for example, arise due to splintering, is guaranteed to have a solution. We select one of those subproblems, and keep track of the corresponding sets of (implicit) edits after i variables in S have been eliminated (i=0,…,|S|). We denote the set of (implicit) edits after i variables in S have been eliminated by Φ_i. The set of (implicit) edits for i=0, Φ_0, is the set of reduced explicit edits.

After all s=|S| variables in S have been eliminated, the set Φ_s of relations not involving any unknowns is consistent (Φ_s may be the empty set, which is consistent by definition). Hence, by the lifting principle (for either categorical, continuous, or integer data; see Theorems 8.1 and 9.3), there is a value $\tilde{v}_s$ for the s-th variable that has been eliminated such that Φ_{s-1} is consistent if we fill in this value. If the s-th variable is categorical and we have several possibilities for $\tilde{v}_s$, we choose $\tilde{v}_s$ such that $a(v_s,\tilde{v}_s)$ is minimal. If the s-th variable is integer-valued and we have several possibilities for $\tilde{x}_s$, we choose $\tilde{x}_s$ such that it is integer and $|\tilde{x}_s - x_s|$ is minimal. For the (s-1)-th variable we apply the same approach, etc.

We continue this process until all values of imputed integer-valued and categorical variables have been modified in the above way. We are then left with a set of imputed continuous variables (if any) and a current set of (implicit) edits involving only these variables. The final values for these variables are then found by minimising (12.5) subject to the constraint that the current set of implicit edits is satisfied. This minimisation problem can simply be formulated as a linear programming (LP) problem, and can, for example, be solved by means of the simplex algorithm (see e.g. Hadley, 1962, and Chvátal, 1983).

Theorem 12.1. The heuristic described above leads to a record that satisfies all edits.

Proof. The various lifting principles (see also Theorem 8.1 and Theorem 9.3) say Φ_i can be satisfied if and only if Φ_{i-1} can be satisfied. If we fill in a value for a categorical or integer-valued variable that satisfies all edits in Φ_i, we hence know that all edits in Φ_{i-1} can also be satisfied by filling in appropriate values for the imputed values that have not yet been modified. After the imputed values of the categorical and integer variables have been modified, the lifting principles also say that the remaining imputed continuous variables can be modified such that all explicit edits become satisfied. The problem of finding modified continuous values that are as close as possible, in the sense of (12.5), to the imputed values can be found by solving an LP problem. ■

As an alternative to solving an LP problem for the continuous variables one could also treat the continuous variables in the same manner as the integer ones, i.e. each time we choose a value $\tilde{x}_i$ for a continuous variable, we select it so that Φ_{i-1} becomes satisfied and $|\tilde{x}_i - x_i|$ is minimal.

Note that if the variables are imputed sequentially, we are not obliged to select the value for which the increase in the objective value is minimal. Instead, we may select any feasible value. Such a feasible value could, for example, also be selected by applying a random draw mechanism using an appropriate probability distribution.

To illustrate the algorithm we give two examples: one involving only categorical variables, the other involving only continuous variables. Example 12.1, involving only categorical variables, below is taken from Kartika (2001).

Example 12.1:
Suppose we have four imputed, categorical variables with domains: $D_1 = \{1,2,3,4\}$, $D_2 = \{1,2,3\}$, $D_3 = \{1,2,3\}$ and $D_4 = \{1,2\}$, and no imputed numerical variables. Suppose also that the reduced edit set is given by:

$$\text{IF } (v_2 = 3) \text{ AND } (v_3 \in \{1,2\}) \text{ AND } (v_4 = 1) \text{ THEN } \varnothing, \tag{12.6}$$

$$\text{IF } (v_2 \in \{2,3\}) \text{ AND } (v_4 = 2) \text{ THEN } \varnothing, \tag{12.7}$$

$$\text{IF } (v_1 \in \{1,2,4\}) \text{ AND } (v_2 \in \{1,3\}) \text{ AND } (v_3 \in \{2,3\}) \text{ THEN } \varnothing \tag{12.8}$$

and

$$\text{IF } (v_1 = 3) \text{ AND } (v_3 \in \{2,3\}) \text{ AND } (v_4 = 1) \text{ THEN } \varnothing. \tag{12.9}$$

Here we use the convention that if a categorical variable is not mentioned in an IF-condition, this variable may take any value. The matrix element $a(v_i, \tilde{v}_i)$ in the objective function (12.4) equals 1 if $v_i \neq \tilde{v}_i$, and 0 otherwise. Suppose that the vector of imputed values is given by $\mathbf{v_0}$=(3, 3, 2, 2). This vector fails edit (12.7).

We apply our algorithm to obtain a consistent record. We start by selecting a variable, say v_1. We eliminate this variable and obtain a set of implicit edits without v_1. This set of implicit edits is given by (12.6), (12.7) and

$$\text{IF } (v_2 \in \{1,3\}) \text{ AND } (v_3 \in \{2,3\}) \text{ AND } (v_4 = 1) \text{ THEN } \varnothing. \tag{12.10}$$

We again select a variable, say v_2, and eliminate this variable from the current set of edits. As a result, we obtain an empty set of implicit edits. This means that we may assign arbitrary values to v_3 and v_4. Because our aim is to keep the final record close to the imputed record, we assign to both variables their original imputed values, i.e. 2. Now, a value has to be assigned to v_2 such that (12.6), (12.7) and (12.10) become satisfied given that to both the third and the fourth variable the value 2 has been assigned. Filling in the value 2 for both the third and fourth variable in (12.6), (12.7) and (12.10) gives the edit

$$\text{IF } (v_2 \in \{2,3\}) \text{ THEN } \varnothing. \tag{12.11}$$

The only possibility to satisfy (12.11) is to assign the value 1 to v_2. Finally, we assign a value to v_1 such that (12.6) to (12.9) are satisfied given the values that have already been assigned earlier. Filling in the values assigned to v_2, v_3 and v_4 in (12.6) to (12.9) gives the edit

$$\text{IF } (v_1 \in \{1,2,4\}) \text{ THEN } \varnothing. \tag{12.12}$$

The only way to satisfy (12.12) is to assign the value 3 to v_1, which happens to be its original imputed value. So, we obtain a new record $\tilde{\mathbf{v}}_0$=(3, 1, 2, 2) with target value $\sum_{i=1}^{m} a(v_i, \tilde{v}_i) = 1$.

If the variables were eliminated in a different order, one might arrive at a different solution with a different target value. To illustrate this we now assume that we start by eliminating v_4 instead of v_1. The set of implicit edits is then given by (12.8),

$$\text{IF } (v_2 = 3) \text{ AND } (v_3 \in \{1,2\}) \text{ THEN } \varnothing \tag{12.13}$$

and

$$\text{IF } (v_1 = 3) \text{ AND } (v_2 \in \{2,3\}) \text{ AND } (v_3 \in \{2,3\}) \text{ THEN } \varnothing. \tag{12.14}$$

We now eliminate variable v_1. The set of implicit edits is given by (12.13) and

$$\text{IF } (v_2 = 3) \text{ AND } (v_3 \in \{2,3\}) \text{ THEN } \varnothing. \tag{12.15}$$

We eliminate v_3, and obtain

$$\text{IF } (v_2 = 3) \text{ THEN } \varnothing. \tag{12.16}$$

as the only implicit edit. We eliminate v_2 and obtain the empty set as the set of implicit edits, which is consistent by definition.

To satisfy (12.16) we have to change the value of v_2. We make v_2 equal to the feasible value nearest to its original imputed value, i.e. to 2. We now have to satisfy (12.13) and (12.15) given the value assigned to v_2. For this we do not have to change the value of v_3. Next, we have to satisfy (12.8), (12.13) and (12.14) by changing the value v_1 given the values already assigned. We make v_1 equal to one of the feasible values nearest to its original imputed value, say to 4. Finally, we have to satisfy (12.6) to (12.9) by changing the value of v_4 given the values already assigned. We make v_4 equal to the only feasible value, i.e. to 1.

So, we obtain a new record $\tilde{\mathbf{v}}_0$=(4, 2, 2, 1) with target value $\sum_{i=1}^{m} a(v_i, \tilde{v}_i) = 3$. This solution is clearly not optimal. ■

Example 12.2 below, involving only continuous variables is taken from De Waal (2002b). It demonstrates how error localisation and consistent imputation are related. It also demonstrates the alternative approach to solving an LP problem.

Example 12.2:

Suppose that the original set of edits is given by

$$T = P + C, \tag{12.17}$$

$$P \leq 0.5T, \tag{12.18}$$

$$-0.1T \leq P, \tag{12.19}$$

$$T \geq 0, \tag{12.20}$$

and

$$T \leq 550N, \tag{12.21}$$

where T denotes the turnover of an enterprise, P its profit, C its costs, and N the number of employees. All variables are considered continuous. Let us consider a specific, erroneous record with values T = 100, P = 40,000, C = 60,000 and N = 5. Edits (12.19), (12.20) and (12.21) are satisfied, whereas edits (12.17) and (12.18) are violated. The reliability weights of variables T, P and C equal 1, and the reliability weight of variable N equals 2. The value of variable N, the number of employees, is hence considered more reliable than the values of the financial variables T, P and C. The only optimal solution to the error localisation problem is: change the values of P and C. We assume that the values of variables P and C indeed have been imputed, say the imputed value of P equals 60, and the imputed value of C equals 90.

We start by filling in the values for the variables that have not been imputed, i.e. T and N. We then obtain the following set of reduced edits:

$$100 = P + C\,, \tag{12.22}$$

$$P \leq 50\,, \tag{12.23}$$

$$-10 \leq P\,, \tag{12.24}$$

$$100 \geq 0\,, \tag{12.25}$$

$$100 \leq 2750\,. \tag{12.26}$$

Edits (12.25) and (12.26) are satisfied and can obviously be discarded.

In this example we will not solve an LP problem as this is not very interesting to describe. More interesting to describe is the alternative approach where we first eliminate all variables and then apply back-substitution to find suitable values.

We select a variable occurring in (12.22) to (12.24), say C. We eliminate this variable from these edits. As edit (12.22) cannot be combined with the other two edits to eliminate C, we only have to copy (12.23) and (12.24) to the new set of edits. We obtain the system given by (12.23) and (12.24).

To check whether the set of edits (12.17) to (12.21) can be imputed consistently by modifying the values of P and C, we could eliminate P from (12.23) and (12.24). The edit we would obtain, $-10 \leq 50$, is satisfied, which shows that (12.17) to (12.21) can indeed be imputed consistently by modifying the values of P and C.

We now select a value for P such that (12.23) and (12.24) become satisfied, i.e. we select a value for P between -10 and 50. As we have mentioned before, any value within this interval may be selected. Here we try to keep the final value of P as close as possible to the imputed value. We therefore select $P = 50$.

Given this value for P, the set of edits (12.22) to (12.26), reduces to

$$100 = 50 + C\,, \tag{12.27}$$

$$50 \leq 50\,, \tag{12.28}$$

$$-10 \leq 50\,, \tag{12.29}$$

$$100 \geq 0\,, \tag{12.30}$$

$$100 \leq 2750\,. \tag{12.31}$$

For the final value of C we select a value that satisfies (12.27) to (12.31). In this case there is only one allowed value, namely $C = 50$. We therefore select this value. The resulting record passes all edits. ■

When only continuous variables have been imputed, our method can, if desired, solve the consistent imputation problem to optimality. When categorical or integer-valued variables have been imputed, optimality of the method is not guaranteed, because the optimal

modified value is sequentially determined for each *individual* categorical and integer-valued variable separately. Optimality of the method would only have been guaranteed if the optimal modified values were determined for all variables simultaneously. However, as we have already mentioned, this is a very difficult problem. The method described is "only" a heuristic. It is, however, much simpler and faster than an optimal method.

12.5. Discussion

The quality of the solutions found by the heuristic described in the previous section seems likely to be acceptable, because all categorical and integer-valued variables are given optimal values when considered separately and the continuous variables are simultaneously given optimal values. To confirm this conjecture about the quality of the found solutions, these solutions have to be compared to the optimal values. These optimal values may be found by formulating the problem of minimising (12.4) subject to the constraint that all edits (12.3) are satisfied while all integer variables indeed attain integer values as a mixed integer programming problem. Such a mixed integer programming may be solved using standard software.

In Section 12.4.2 we have not described the order in which the imputed variables are modified. This order influences the quality of the method, i.e. the difference between the value of (12.4) found by the heuristic and the optimal value, as well as the computing time of the method. For a mix of categorical and continuous data a prototype computer program has been developed implementing the above heuristic. This program has been used to test a number of possible orders. The program has been developed by Kartika, a post-graduate student from Delft University of Technology, who also carried out the evaluation tests of this program. Kartika was supervised by Dr. Kraaikamp at Delft University of Technology, and by the author of this book at Statistics Netherlands. Kartika (2001) gives the results of the evaluation tests, and also describes details of the implemented algorithm. For the data set used by Kartika both the quality of the obtained solutions and the computing time were acceptable for Statistics Netherlands. Due to these evaluation results Statistics Netherlands decided to implement the algorithm of Section 12.4 in a module of our SLICE system.

13. Practical Issues of Error Localisation

13.1. Introduction

In this chapter we discuss three practical issues. The first issue is how to handle records that contain many errors. The error localisation algorithms of the previous chapters, which in most cases aim to find all optimal solutions to the error localisation problem, are generally too time-consuming and memory-consuming for such records. In Section 13.2 we describe a simple and fast approach to obtain a, possibly non-optimal, solution to the error localisation problem. This approach is based on the heuristic for consistent imputation described in Section 12.4.2.

The second issue is related to designing a "good" set of edits. This is a non-trivial exercise. Designing a good set of edits is a complicated process that requires a lot of experience and subject-matter knowledge. Furthermore, many statistical analyses may have to be carried out to arrive at a good set of edits. We do not attempt to describe these statistical analyses here, let alone the entire process of developing a set of explicit edits. We have a much more limited aim in this chapter, and only describe a basic test for a set of edits. A prerequisite for a good set of edits is that the edits do not contradict each other, i.e. that the set of edits is consistent. In Section 13.3 we sketch how the consistency of a set of edits for mixed data can be tested. In the same section we also briefly describe a method to detect redundant explicit edits. This may be valuable information for a developer of the edit set.

The third issue we address in this chapter is how subject-matter knowledge can be used in combination with a system based on the Fellegi-Holt paradigm. In Section 13.4 we describe various ways in which subject-matter knowledge can influence the results of such a system. Section 13.4 is based on De Waal (2000c). Section 13.5 ends this chapter, and the part of this book on statistical data editing, by describing the impact of the developed methodology on practice at Statistics Netherlands.

13.2. Handling records with many errors

As we mentioned in the introduction to this chapter, solving the error localisation problem for records containing many errors is generally too demanding in terms of computing time and computer memory for the exact error localisation algorithms of earlier chapters. To handle such records we suggest a much simpler and faster approach. This approach does not aim to find (all) optimal solutions to the error localisation problem. In fact, it does not try to solve the mathematical error localisation problem as formulated in Section 3.2. Instead, we try to solve a special instance of the consistent imputation problem of Section 12.4.

In this special instance, we first fill in arbitrary values for the missing values. Subsequently, we try to minimise

$$\sum_{i=1}^{m} w_i a(v_i^0, \tilde{v}_i) + \sum_{i=1}^{n} w_{m+i} \mid x_i^0 - \tilde{x}_i \mid , \qquad (13.1)$$

where the original record (after filling in arbitrary values for the missing values) is given by $(v_1,...,v_m,x_1,...,x_n)$, and the final record by $(\tilde{v}_1,...,\tilde{v}_m,\tilde{x}_1,...,\tilde{x}_n)$. As in Section 12.4. the w_i are user-specified weights, and $a(v_i,\tilde{v}_i)$ is a non-negative matrix satisfying $a(v_i,\tilde{v}_i)=0$ if $v_i=\tilde{v}_i$. We try to minimise the objective function (13.1) subject to the constraint that all edits (12.3) become satisfied. We can do this by applying the heuristic of Section 12.4.2.

After completion of the heuristic the variables for which the final values differ from the values of the original record (after filling in arbitrary values for the missing values) plus the variables for which the values were originally missing are considered to be erroneous. These erroneous may be imputed by a suitable imputation method. As the objective function (13.1) is minimised subject to the constraint that all edits (12.3) become satisfied, it is obviously guaranteed that the variables considered to be erroneous can indeed be imputed in a consistent manner, i.e. in such a way that all edits become satisfied.

The objective function (13.1) does not measure the (weighted) number of fields that need to be changed. Instead, it measures the distance between the values of the original record (after filling in arbitrary values for the missing values) and a consistent synthetic record. The closest – in terms of (13.1) – synthetic record that satisfies all edits defines the fields that are considered erroneous.

For purely numerical data the problem of minimising (13.1) subject to the constraint that all edits become satisfied reduces to a linear programming (LP) problem. This LP problem can usually be solved in a mere fraction of the time required for the corresponding instance of the mathematical error localisation problem defined in Chapter 3.

Harte, a student from the Hogeschool of Amsterdam (College of Amsterdam), has developed a prototype computer program based on the above approach. He has tested the program and approach on Data set C described in Chapter 11 (for a slightly different set of edits). His conclusion was that the LP approach is indeed much faster than any of the algorithms evaluated in Chapter 11. As was to be expected, the LP approach did require more fields to be changed than the algorithms based on the Fellegi-Holt paradigm. Whereas the algorithms based on the Fellegi-Holt paradigm required 3,361 fields in total to be changed, the LP approach required 4,139 fields to be changed, i.e. about 23% more. After imputation and solving the consistent modification problem of Section 12.4 the publication figures obtained by Harte were comparable to the results obtained by applying the Fellegi-Holt paradigm, and in some cases the former were even a bit better. For more details regarding the LP approach and the results of the evaluation study we refer to Harte (2000).

For records containing many errors the LP approach is our preferred approach. The Fellegi-Holt paradigm was designed to identify the fields that are most likely to be erroneous. For records containing only a few errors, application of the Fellegi-Holt paradigm is frequently successful, i.e. the identified fields are indeed incorrect. For records containing many errors, application of the Fellegi-Holt paradigm hardly ever achieves its

goal. The fields identified as being erroneous are hardly ever really incorrect. For such highly erroneous records, one could just as well resort to the LP approach. In our opinion, it is not very important that a record that could be made consistent by changing, say, 8 fields is in fact made consistent by changing, say, 10 fields. Generally, the probability that either the 8 fields or the 10 fields are indeed the erroneous ones is almost equal to zero. In contrast, we would object to making a record consistent by changing four fields if this record could be made consistent by changing only two fields as it is generally much more likely that the latter two fields are indeed erroneous than the former four.

13.3. Testing the set of explicit edits

To test whether a set of explicit edits is consistent or not we can again apply the algorithm described in Chapters 8 and 9. Namely, to test the consistency of a set of edits we simply eliminate all variables. If only categorical and continuous variables are involved in the edits, we obtain a set of relations without any unknowns. If this set of relations is consistent, the set of explicit edits is consistent. Otherwise, the set of explicit edits is inconsistent. This immediately follows from Theorem 8.2. If categorical, continuous and integer variables are involved in the edits, we obtain several sets of relations without any unknowns. These sets correspond to the dark shadow and "splinters" due to elimination of the integer variables (see Chapter 9). If any of these sets of relations without unknowns is consistent, the set of explicit edits is consistent. Otherwise, the set of explicit edits is inconsistent. This follows from Theorem 9.3.

Note that a consistency test, such as the one mentioned above, can also be used to test whether a given set S of variables can be imputed consistently. Fill in the values of the variables not contained in S into the original edits. This results in a set of reduced edits involving only variables in S. It is possible that some of the edits in the set of reduced edits involve no unknowns anymore. If any such an edit is not satisfied, the variables in S cannot be imputed consistently. Otherwise, test the set of reduced edits for consistency. If it is indeed consistent, the variables in S can be imputed consistently. If the set of reduced edits is inconsistent, the variables in S cannot be imputed consistently. This method for testing whether it is possible to impute a specific set of variables consistently might, for example, be implemented in a computer program suited for interactive editing such as Blaise. After a subject-matter specialist has specified which variables he is planning to modify, the computer program could then test whether these variables can indeed be imputed in a manner consistent with the edits.

Testing whether a certain edit is redundant, i.e. can be deleted without altering the feasible region described by the edits, can be re-written as a consistency test for a certain collection of edits. We suppose that we start with a consistent edit set.

We want to test whether an edit E^j given by

$$\text{IF} \quad v_i \in F_i^j \text{ for } i=1,\ldots,m$$

$$\text{THEN} \quad (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j \geq 0\}, \qquad (13.2a)$$

or

$$\begin{aligned} &\text{IF} \quad && v_i \in F_i^j \ \text{ for } i=1,\ldots,m \\ &\text{THEN} \quad && (x_1,\ldots,x_n) \in \{\mathbf{x} \mid a_{1j}x_1 + \ldots + a_{nj}x_n + b_j = 0\}\,, \end{aligned} \tag{13.2b}$$

is redundant. To this end we first negate the edit. If E^j is an inequality, its negation is given by

$$\overline{E}_1^j \ \text{AND} \ \overline{E}_2^j \ \text{AND} \ \ldots \ \text{AND} \ \overline{E}_m^j \ \text{AND} \ \overline{E}_*^j\,, \tag{13.3}$$

where $\overline{E}_i^j$ is given by

$$\text{IF} \ \ v_i \in D_i - F_i^j, v_k \in D_k \ \ (\text{for } k \neq i) \ \ \text{THEN} \ \varnothing \tag{13.4}$$

and $\overline{E}_*^j$ is given by

$$\sum_{i=1}^{n} a_{ij}x_i + b_j < 0\,. \tag{13.5}$$

The AND-operators in (13.3) mean that all these edits have to be satisfied simultaneously.

If E^j is an equality, its negation is given by

$$\begin{aligned} &(\,\overline{E}_1^j \ \text{AND} \ \overline{E}_2^j \ \text{AND} \ \ldots \ \text{AND} \ \overline{E}_m^j \ \text{AND} \ \overline{E}_*^j\,) \\ &\text{OR} \\ &(\,\overline{E}_1^j \ \text{AND} \ \overline{E}_2^j \ \text{AND} \ \ldots \ \text{AND} \ \overline{E}_m^j \ \text{AND} \ \overline{E}_{**}^j), \end{aligned} \tag{13.6}$$

where the $\overline{E}_i^j$ (i=1,…,m) are given by (13.4), $\overline{E}_*^j$ by (13.5), and $\overline{E}_{**}^j$ by

$$\sum_{i=1}^{n} a_{ij}x_i + b_j > 0\,. \tag{13.7}$$

In (13.6) the OR-operator means that at least one of the statements between brackets has to hold true. The AND-operators mean that all edits between brackets have to be satisfied simultaneously for a compound statement between brackets to hold true.

The edits (13.5) and (13.7) are not yet in standard form. Therefore, we re-write the THEN-condition (13.5) as

$$\sum_{i=1}^{n} a_{ij}x_i + b_j \leq -\varepsilon\,, \tag{13.8}$$

and the THEN-condition (13.7) as

$$\sum_{i=1}^{n} a_{ij} x_i + b_j \geq \varepsilon \tag{13.9}$$

for a sufficiently small number ε. We can hence express the negation of an edit as one or two sets of edits in our standard type (see (13.2)).

Now, in the case that the edit under consideration E^j is an inequality, if the other edits in combination with (13.3) form an inconsistent edit set, we can conclude that all possible records that satisfy the other edits also satisfy E^j. In other words, if the other edits in combination with (13.3) form an inconsistent edit set, E^j is redundant and can be deleted. Similarly, in the case that E^j is an equality, if the other edits in combination with (13.6) form an inconsistent edit set, E^j is redundant and can be deleted.

We can give a formal proof that the above method to determine redundant edits is correct using simple propositional logic (for an introduction to propositional logic we refer to Ben-Ari, 2001). Edit E^j is by definition redundant if and only if

$$(\underset{i \neq j}{\wedge} E^i) \rightarrow E^j \tag{13.10}$$

is a valid formula. This is equivalent to

$$\neg(\underset{i \neq j}{\wedge} E^i) \vee E^j \tag{13.11}$$

being a valid formula, where "$\neg$" denotes negation. In turn this is equivalent to

$$(\underset{i \neq j}{\wedge} E^i) \wedge \neg E^j \tag{13.12}$$

being an unsatisfiable formula, i.e. $\neg E^j$ and the edits E^i contradict each other. It remains to find the negation of E^j. For ease of notation we write E^j as

$$E^j : (C_1^j \wedge \ldots \wedge C_m^j) \rightarrow N(x), \tag{13.13}$$

where C_k^j denotes $v_k \in F_k^j$, and $N(x)$ the numerical condition $\sum_{i=1}^{n} a_{ij} x_i + b_j \geq 0$, or $\sum_{i=1}^{n} a_{ij} x_i + b_j = 0$. Alternatively, we can write (13.13) as

$$E^j : \neg(C_1^j \wedge \ldots \wedge C_m^j) \vee N(x). \tag{13.14}$$

The negation of E^j can hence be written as

$$\neg E^j : (C_1^j \wedge \ldots \wedge C_m^j) \wedge \neg N(x). \tag{13.15}$$

Finally, we rewrite the C_k^j so they are in our standard form (13.2). C_k^i is equivalent to

$$\neg(\neg C_k^j) \vee false\,, \tag{13.16}$$

which is equivalent to

$$(\neg C_k^j) \rightarrow false\,. \tag{13.17}$$

This is exactly the condition given by (13.4). ■

A direct interpretation of (13.3) and (13.6) is easy to provide. The purely categorical edits given by the $\overline{E}_i^j$, which have to be met simultaneously, say that edit E^j is redundant if its IF-condition cannot be satisfied given the other explicit edits. The mixed edit given by $\overline{E}_*^j$, in combination with the mixed edit $\overline{E}_{**}^j$ in the case that E^j is an equality, says that E^j is redundant if the numerical variables cannot attain values outside the region described by the THEN-condition of E^j.

Note that the negation of a purely categorical edit, i.e. an edit for which the THEN-condition is given by $\{\mathbf{x} \mid \mathbf{x} \in \varnothing\} \equiv false$, is a set of purely categorical edits. The only, potentially, numerical edit (see (13.15)) would be $\neg false \equiv true$. The negation of a purely numerical edit, i.e. an edit for which the sets $F_i^j = D_i$ (for all i=1,...,m), is again a purely numerical edit. All categorical edits that arise due to the negation of E^j are satisfied because $D_i - F_i^j = \varnothing$ for some i=1,...,m (see (13.4)).

In Van den Broeke (2001) an alternative method to detect inconsistency and redundant edits in a set of explicit edits is proposed. This method is based on solving certain LP-problems for many combinations of categorical values. It is more powerful, because it can also detect for which combinations of categorical values the corresponding numerical constraints form an inconsistent system. However, the method is bound to be much more time-consuming than the simple method proposed in this section.

13.4. Using subject-matter knowledge in a Fellegi-Holt program

13.4.1. Introduction

An ideal system for automatic edit and imputation would focus on the statistical distribution of the final data, while taking into account that the errors in the raw data may have been introduced by several different error generation mechanisms. In discussions several persons have expressed their doubts whether a Fellegi-Holt (FH) system, i.e. a computer program for automatic editing based on the Fellegi-Holt paradigm, offers sufficiently flexibility for it to come even close to an ideal system. Their criticism focuses on the following aspects of an FH system:

- an FH system cannot make a distinction between hard edits, which have to be satisfied for every record, and soft – or statistical – edits, which have to be satisfied for only most of the records;

- an FH system makes a too rigid assumption about the error generating mechanism, viz. such a system assumes that the errors in each data set can be identified by minimising a weighted number of fields that need to be changed in order to satisfy the edits;
- an FH system is not guided by the final statistical distribution of the data, but only by the edits that have to be satisfied.

The first two points express a doubt whether an FH system can take different error generation mechanisms into account. The third point expresses a doubt whether an FH system can be so tuned that it succeeds in preserving the statistical distribution of correct data. We have to admit that this criticism is partly justified. An FH system is, for example, only indirectly guided by the final statistical distribution of the data. An FH system is not an ideal system for automatic editing and imputation, but in this section we argue that by using subject-matter knowledge as much as possible an FH system can come close to an ideal one. In any case, we hope to show in this section that an FH system is quite flexible, and offers a lot of opportunities to utilise available subject-matter knowledge. The use of subject-matter knowledge can to a considerable extent overcome the above-mentioned criticism.

Subsections 13.4.2 to 13.4.5 describe various ways of providing subject-matter knowledge to an FH system. Subsection 13.4.2 is devoted to the edits, and Subsection 13.4.3 to reliability weights. These two subsections explain how an FH system may take different error generation mechanisms into account. Subsection 13.4.4 is devoted to imputation, and Subsection 13.4.5 to the selection of fields to be modified in the case that the FH paradigm results in optimal ties. Those two subsections explain how an FH system may aim to preserve the statistical distribution of the data rather than simply aim to change as few fields as possible. The final subsection, Subsection 13.4.6, concludes this section with a very brief discussion.

Application of the (generalised) FH paradigm may yield several optimal ways, i.e. sets of fields with a minimum sum of reliability weights, to modify the data. In this section we will refer to such a set of fields as an FH-optimal set of fields.

13.4.2. Edits

The most obvious way of supplying subject-matter knowledge to an FH program is by means of the edits. Subject-matter knowledge can be used to specify which records are allowed, and which records are not allowed.

The edits used during automatic editing often differ from the edits that are used during computer-assisted editing. Some of the edits that are used during computer-assisted editing are logical, or hard, edits. These edits logically have to be satisfied by each consistent record. Other edits are statistical, or soft, edits. These edits do not have to be satisfied by each consistent record, but are satisfied by the vast majority of these records. During computer-assisted editing humans may decide to overrule the specified edits. They may decide not to change a record that violates one or more edits, they may also decide to change a record that satisfies all edits.

When records are edited automatically, all records that do not satisfy the edits and none of the records that satisfy the edits are modified. This means that if the program uses the same edits as during computer-assisted editing, some records may be edited too much, whereas others may not be edited enough.

It seems very natural to demand that an FH program ensures that all logical edits become satisfied after application of the program. The statistical edits, however, usually have to be adapted in order for them to be useful for a computer program for automatic editing. Subject-matter knowledge and experience with such a computer program are both required to arrive at an optimal set of edits for automatic editing.

There are two ways to deal with edits specified by subject-matter specialists. The first approach is to modify a record when any edit (either a logical one or a statistical one) is violated. The second approach is to modify a record when either a logical edit is violated or when the statistical edits are violated to a substantial extent.

In the computer programs for automatic editing that we know of the first approach is usually taken. The second approach is more flexible, however, and would be our preferred option. If a record does not violate the logical edits and the statistical edits are only violated to a limited extent, the record is not modified. In other words, the violated statistical edits will remain violated for that record. Once it is decided that a certain record will be modified automatically, we propose to take the statistical edits into account. For such a record all edits, both logical and statistical ones, will become satisfied.

To determine the extent in which the statistical edits are violated one could, for instance, consider the following: the number of violated statistical edits, the deviations of the variables and the edits from their usual values, the raising weight of the record, and possibly weights that indicate how "soft" each edit is. Alternatively, an outlier detection method could be used to measure the violation of the statistical edits. A record that only violates statistical edits is then modified automatically if it is considered to be an outlier, else it is not modified.

Balance edits, i.e. edits saying that the sum of a number of variables should be equal to the sum of some other variables, are usually treated as logical edits in an FH system. However, small deviations are quite common in practice, and can naturally arise due to rounding errors. Therefore, a better approach would be to treat balance edits as statistical edits rather than as logical edits as long as the violation is small. Large violations should never be allowed, and should be prohibited by hard edits. Subject-matter knowledge is necessary to specify up to which point balance edits should be treated as statistical edits.

13.4.3. Reliability weights

The reliability weight that is provided by the subject-matter specialists for a certain variable reflects to what extent they a priori trust the values of this variable. Variables of which it is known that their values are frequently incorrect should hence be given a low weight. Variables of which it is known that their values are rarely incorrect should be given a high weight. In the extreme case, the reliability weight of a variable of which the value cannot be incorrect equals infinity.

The quality of data that have been edited automatically is often strongly influenced by the reliability weights assigned to the variables. A badly chosen set of weights may have a catastrophical influence on the quality of the resulting data. Correct values may then be changed, while incorrect values are not changed.

In most of the computer programs for automatic editing the reliability weights of the variables are fixed before the program starts. CherryPi is an exception: for each record the user-specified reliability weights are dynamically adjusted to reflect the occurrence of potential typing errors (see Van de Pol, Bakker and De Waal, 1997). To determine whether a value is the result of a typing error the balance edits are used. If such an edit is violated and can be satisfied by, for example, swapping two digits of a certain value, the reliability weight of this variable is dynamically lowered. The value of that variable is then more likely to be changed.

This idea can clearly be extended. Each record could be pre-processed to dynamically determine the set of reliability weights. The user-specified weights could be used as the initial set of weights. The subject-matter specialist could specify certain rules to adjust these weights. For instance, for certain types of records it may be well known that a particular error pattern occurs very frequently. The variables involved in this error pattern may then be given low reliability weights. An example of such a regularly occurring error is a factor 1,000 error, where a financial figure has to be filled in in thousands of Euros, but the respondent mistakenly answered in Euros. Such a potential error may often be easily detected. The reliability weight of the corresponding variable can then be given a low value.

Information on the occurrence of certain error patterns may be obtained during the application of a computer program for error localisation. That is, previously edited records may provide valuable information about the errors in the records remaining to be edited automatically.

One can, for example, also compare the value of a variable to a “normal” value of this variable. If there is a big difference, the reliability weight of this variable may be given a low value. How a “normal” value and a “big” difference are defined has to be specified by a subject-matter specialist, or have to be based on statistical analyses.

A final example of how reliability weights may be dynamically adjusted is when a total differs from the sum of the original values of the constituent variables. In some cases one may then want to give the total a relatively high reliability weight so that its value is unlikely to be changed. In other cases, one might distrust the value of the total. One should then give the total a relatively low reliability weight so its value is likely to be modified. What value should be assigned to the reliability weight of a total should depend on subject-matter knowledge.

Based on subject-matter knowledge methods for dynamically adjusting reliability weights, such as in the examples mentioned above, have to be chosen. Also the parameters that should be used for these methods should be based on subject-matter knowledge or on statistical analyses.

13.4.4. Imputation

In this subsection we first discuss possible imputation models in an FH system, and then discuss how consistent imputation can be obtained.

Imputation model

A strong aspect of an FH system is that – to a large extent – the user can specify which imputation methods he would like to use. In contrast, a system like NIM uses only a hot-deck donor imputation method (see e.g. Bankier, 1999; Bankier et al., 2000). In an FH system the user can choose from a much larger class of techniques, ranging from very simple imputation methods, such as imputing the median/modal value of a variable, to advanced imputation techniques, such as imputation based on the EM algorithm or imputation based on clustering algorithms. In Chapter 12 of this book we discussed imputation techniques to a limited extent; for a more thorough discussion of these techniques we refer to Kalton and Kasprzyk (1986), and Kovar and Whitridge (1995).

Subject-matter knowledge may be used to specify an appropriate imputation technique for the data set under consideration. For some data sets, for example data from social surveys, hot-deck imputation may be a very appropriate technique, whereas for other data sets, for example data from establishment surveys, this technique may be inappropriate. For such surveys it is often more advisable to use regression imputation.

Of course, subject-matter knowledge may not only be used when choosing the most appropriate imputation technique for a given survey, but also when determining suitable parameters for the imputation techniques. For instance, when regression imputation is used, subject-matter knowledge may be used to determine suitable auxiliary variables. Regression coefficients can subsequently be calculated easily by means of a computer.

An FH system may not only be used to edit data, but also to impute for item-nonresponse. However, in, for instance, an establishment survey it is often unclear whether a certain value is missing or not. Values that equal zero are often not filled in by the respondents. Subject-matter knowledge should be used to decide for a missing quantitative item whether the value is really missing or whether it equals zero.

Selecting an appropriate imputation technique and associated parameters is essential for the quality of data that have been edited by an FH system. In this selection process, subject-matter knowledge plays an essential role. The role of the computer is restricted to calculating certain parameters, and to actually imputing the data set.

Consistent imputation

Specifying the imputation model in such a way that all resulting records will be consistent is often too difficult in practice. Therefore, an alternative approach is frequently used. This approach consists of two steps. In the first step the missing and implausible values in a record are simply imputed by using an imputation model without taking the edits into account. In the second step the imputed values are modified slightly in such a way that the resultant record will be consistent, and will be as close as possible to the record obtained after the first step. This second step can be done by solving a mathematical optimisation

problem where the objective is to change the record after the first imputation step as little as possible subject to the constraint that the final record satisfies all edits (see Section 12.4 for more details).

If this approach is used, a distance function has to be specified that measures how close two records are. For practical reasons, the distance functions will have to be restricted to certain classes. For instance, for records consisting only of numerical data, the distance between two records might be measured by the Euclidean distance function, or by the sum of the absolute values of the differences between the values of the variables. Within a class of distance functions the particular distance function that is used may be specified by a subject-matter specialist. For instance, numerical data may first be scaled in a certain user-specified way.

By specifying parameters of the distance function, the subject-matter specialist may force that the values of certain variables will be changed relatively less than the values of other variables during the second step of the imputation process. For example, one may demand that variables that can be imputed very accurately during the first step of the imputation process are changed less than variables for which no good imputation model can be specified.

The use of subject-matter knowledge as described in this subsection is admittedly rather abstract. Apart from subject-matter knowledge a considerable amount of experience and mathematical understanding is required to obtain the best possible results.

13.4.5. Selection of FH-optimal set of fields to be modified

If there are several FH-optimal sets of fields, one of these sets has to be selected. For this, additional criteria may be used. In the FH based automatic editing programs we know of one FH-optimal set of fields is selected before the FH-optimal sets of fields have been imputed. In some of FH systems this is done implicitly, because only one FH-optimal set of variables is generated; in other systems all FH-optimal sets of variables are generated and then one set is explicitly selected. The selected variables are subsequently imputed. A possibly better approach would be to first impute all FH-optimal sets of fields and only then select one of these sets. This approach would allow the use of more advanced additional criteria to select an FH-optimal set of fields.

In this subsection we give examples of the kinds of subject-matter knowledge that can be used for both possible approaches.

In both cases subject-matter knowledge is used to help the computer identify the set of variables with incorrect values. This subject-matter knowledge has to be translated into a secondary optimality criterion. Supplementing the FH paradigm – the primary optimality criterion – by such subject-matter knowledge in this way can lead to an improved algorithm for finding the incorrect fields.

Selection of fields to be modified before imputation

If the fields that have to be modified are selected before the FH-optimal sets are imputed, we have, for example, the following two possibilities for the secondary optimality criterion:

- Construct a ranking list of the most common error patterns. The FH-optimal set of fields that has the highest rank is then selected.
- Compare the (original) values of the variables in each FH-optimal set of fields to "normal" values for these variables. The FH-optimal set for which the original values deviate most from the "normal" values is then selected. "Normal" values may be determined in many ways, for example by taking median/modal values.

The fields of the selected FH-optimal set will be imputed. The use of subject-matter knowledge allows one to select the FH-optimal set of fields that most likely is the actual set of incorrect fields.

Other kinds of secondary optimality criteria may also be constructed. It depends on the data set under consideration, and hence on subject-matter knowledge, which criterion is the best one.

Selection of fields to be modified after imputation

If the fields that have to be modified are selected after the FH-optimal sets are imputed, the most logical criterion to select an FH-optimal set of fields to be modified is to compare the resulting records with "normal" records. The FH-optimal set of fields that leads to the most "normal" record may then be selected as the set of fields to be modified.

For such a comparison a distance function should be specified, and "normal" records should be defined. A distance function should be dependent on the data under consideration and the purpose of the editing process. "Normal" records could again be defined by taking median/modal values, for instance. It should be clear that a distance function and the concept of "normal" records should be based on subject-matter knowledge. The use of subject-matter knowledge allows one to select the FH-optimal set of fields that, once imputed, leads to the record that is most likely to be, or closest to, the correct one.

Of course, other kinds of secondary optimality criteria may also be constructed. Again, it depends on the data set and on subject-matter knowledge which criterion is the best one.

13.4.6. Discussion on using subject-matter knowledge in a Fellegi-Holt program

The tasks that have to be performed in order to edit a data set can be split into conceptually difficult tasks and conceptually easy tasks. Humans are better in solving the former kind of tasks than computers. The latter tasks may be conceptually easy, but may at the same time require considerable computing power to solve. Computers are therefore better in solving these tasks.

An FH based computer program enables one to have the best of both worlds. For the conceptually easy tasks of identifying the FH-optimal sets of fields and imputation of better values for an FH-optimal set, the computer is perfectly suited. Humans are much less good in executing these tasks, because both tasks require many accurate computations. For the conceptual hard tasks, such as designing a set of edits, designing imputation models, and determining (rules for computing) reliability weights, human input is essential. In this book we have indicated several possibilities for such human input.

The final form of human input that has not yet been mentioned in this chapter, but that is of utmost importance, is the decision whether to accept the data that have been edited automatically or not. It is up to the human subject-matter specialist to decide whether the data that have been edited automatically are of acceptable quality.

The combined use of subject-matter knowledge and computing power of PC's can lead to high quality data that have been edited automatically. Unfortunately, the present-day FH systems for automatic editing only allow the use of subject-matter knowledge to a rather limited extent. The main conclusion that can be drawn from this chapter is that FH based computer systems for automatic editing should become more flexible in order to allow the maximal input of subject-matter knowledge.

In the author's opinion, SLICE, the general software framework for editing and imputation that is currently being developed by Statistics Netherlands (see e.g. De Waal and Wings, 1999; De Waal, 2000d and 2001b), offers great potential in this respect. SLICE itself is planned to become an add-on module of Blaise, the integrated survey processing system that has been developed by Statistics Netherlands (see *Blaise Reference Manual,* 2002, and the *Blaise Developer's Guide,* 2002). Over the years Blaise has become the de facto standard for survey processing. Many modules such a Manipula (for manipulating data sets) and Bascula (for calculating raising weights) have been added to the original core that was meant for questionnaire design and interviewing. Nowadays, Blaise is used by over 80 different statistical institutes all over the world. By connecting SLICE to the Blaise system, SLICE will be able to read and write data files and corresponding meta-data in Blaise format. Moreover, the syntax of SLICE is largely based on the Blaise language. The flexibility offered by the Blaise system will enable subject-matter knowledge to be used during various stages of the (automatic) editing process.

13.5. The impact of the developed methodology on practice at Statistics Netherlands

In this last section on statistical data editing, we consider the impact of the developed methodology described in this book on the daily practice at Statistics Netherlands. First of all, we are glad to say here that the developed methodology has indeed had an impact. In fact, when Statistics Netherlands was being re-organised in 1999-2000 the availability of methodology and software tools for efficient statistical data editing, such as the methodology and tools described in this book, was explicitly mentioned as one of the reasons for this re-organisation.

In 1995 we started our research on automatic editing. In those days obtaining a license for a commercial solver for mixed integer programming problems was not even contemplated. Instead, we therefore studied the methodology of Chapter 4. Implementation of the

algorithm described in Chapter 4 for continuous data was quite simple. Unfortunately, tests quickly showed that the developed prototype software was far too slow for practical use.

The first version of software for automatic error localisation at Statistics Netherlands that could be applied in practice, CherryPi based on the vertex generation approach of Chapter 5, was finished early 1996. That version ran under DOS and was suited for purely continuous data only. In subsequent years, CherryPi was tested and converted to a Windows version. Early 1998 it was decided to integrate CherryPi into SLICE (version 1.0), a framework for automatic editing and imputation that was then to be developed. In those years CherryPi or SLICE were not yet used in practice at Statistics Netherlands.

The breakthrough came later when the Division of Business Statistics decided to revise its production processes for annual structural business surveys and design a completely new uniform computer system to collect and process data for such surveys. After many discussions and several tests, it was decided to use a combination of selective editing (for details on this implementation of selective editing see Hoogland, 2002) and automatic editing to produce clean data. As software framework for automatic editing and imputation it was decided to use SLICE 1.0. For a general overview of the approach for annual structural business surveys at Statistics Netherlands and a brief description of the software system we refer to De Jong (2002).

After the first Windows-based version of CherryPi was finished we continued with methodological research on automatic error localisation in order to be able to handle more general data, namely a mix of categorical and continuous data, and edits. The first attempts, the extension to categorical data of the vertex generation approach (cf. Chapter 5), the formulation as a dynamic disjunctive-facet problem (cf. Chapter 6), and the heuristic developed by Pergamentsev (cf. Chapter 7), were unsuccessful, mainly because the developed algorithms were considered too complicated to implement and maintain. The branch-and-bound approach of Chapter 8 was, however, sufficiently simple to implement and maintain at a statistical bureau. Moreover, the tests of Chapter 11 show that the approach is also sufficiently fast for practical application. The cutting plane algorithms described in Chapter 10 were developed later in order to examine whether these approaches would perform better with respect to computing speed than the algorithm described in Chapter 8. The computational results of Chapter 11 show otherwise, however. The cutting plane algorithm described in Section 10.5 to 10.7 has the benefit that it is even simpler to implement and maintain than the algorithm of Chapter 8. For statistical offices looking for an easy to implement algorithm for automatic error localisation and with a low budget, that algorithm would be an excellent starting point. Statistical offices with a higher budget could also consider obtaining a license for a commercial solver for mixed integer programming problems and use that solver to solve the problem formulated in Section 3.4. That approach is also quite simple to implement and maintain.

As the algorithm described in Chapter 8 is easy to implement and has a good performance, we therefore decided to implement that algorithm in SLICE (version 1.5). The implemented algorithm contains a heuristic to handle integer-valued data based on the algorithm of Chapter 9. The complete, exact algorithm of Chapter 9 has not been implemented: again because it would be too complicated to implement and maintain. The beta version of SLICE 1.5 has been released for testing purposes early 2003. The Division of Business Statistics at Statistics Netherlands is planning to use SLICE 1.5, and its new

module for automatic error localisation (CherryPie), in its production processes in the near future.

Version 1.5 of SLICE contains a heuristic to handle records containing many errors. At the moment, the favoured method is the heuristic described in Section 13.2. In future this heuristic may, perhaps, be replaced by an alternative heuristic developed by Chung (2003) that is based on setting suspect values to missing and only then use CherryPie to correct the thus created record.

SLICE also contains a module for consistent imputation. Version 1.0 contained a simple module based on solving a linear programming problem. Version 1.5 contains an implementation of the heuristic described in Section 12.4. This heuristic is quite easy to implement and maintain.

Methodology for testing the set of explicit edits (see Section 13.3) has not (yet) been implemented in SLICE, although it would not be very complicated to do so. The reason for not implementing such methodology is that this has low priority. It is not clear whether implementing such methodology has many benefits as it only allows one to detect inconsistent and redundant edits, but not more subtle mistakes in the specified edits.

WAID, the software package for imputation sketched in Section 12.3, has made a slight impact on the production processes at Statistics Netherlands as it is currently being applied in practice by a limited number of users.

14. A View on Statistical Disclosure Control

14.1. Introduction

The aim of statistical disclosure control (SDC) is to prevent sensitive information about individual respondents, or small groups of respondents, from being disclosed. SDC is becoming increasingly important due to the growing demand for information provided by statistical offices. The information released by these statistical offices can be divided into two major parts: tabular data and microdata. Whereas tables have been released traditionally by statistical offices, microdata sets are released only since fairly recently. In the past the users of data usually did not have the tools to analyse these microdata sets properly themselves. Nowadays every serious researcher is in possession of a powerful personal computer. Analysing microdata is therefore no longer a privilege of the statistical office. The users of data can and want to analyse these microdata themselves. This creates non-trivial SDC-problems.

As absolute prevention of disclosure of sensitive information about individual respondents, or small groups of respondents, can only be guaranteed if no or hardly any information is released, this aim would be far too restrictive for statistical offices. A more realistic aim is to *limit* the probability that sensitive information about individual respondents or small groups of respondents can be disclosed. In this book we therefore concentrate on limiting the probability that sensitive information about individual respondents can be disclosed.

Sensitive information about an individual respondent might be disclosed if the respondent were re-identified by an attacker. For example, suppose we have data on the residence of respondents, their occupation, their sex and their income. Suppose also that the income of an individual respondent is considered sensitive information, as is the case in the Netherlands. In a record with the following values *Residence* = *Amsterdam*, *Occupation* = *Mayor* and *Sex* = *Male*, sensitive information on the income of this, easily re-identifiable, person could immediately be disclosed. In fact, this person could even be re-identified without knowledge on his or her sex. To limit the risk of disclosure one could decide not to release the income of this person, for instance. Another example of a record from which sensitive information might be disclosed if we are not careful is a record with the following values *Residence* = *Urk*, *Occupation* = *Statistician* and *Sex* = *Female*. Urk is a small village in the Netherlands. In addition, in the Netherlands there are not many female statisticians. It is hence quite likely that there is only one female resident in Urk whose occupation is statistician. It would be quite easy to re-identify this person and to disclose her income in the case that income were released. Even if there were more than one female resident in Urk whose occupation is statistician, it is still very likely that there would only be a few. By guessing, an attacker would then have a relatively good chance to re-identify the correct person. We conclude that it is necessary to protect this record against disclosure.

A key problem in the theory of SDC for microdata is therefore the determination of the probability that a record in a released microdata set is re-identified. In order to estimate

this probability a number of different approaches have been attempted. The aim of these attempts differs considerably. In some publications the aim is to gain a qualitative insight into the probability of re-identification of an unspecified record from a microdata set. In other publications the aim is set much higher, namely to obtain the probability that a specific record is re-identified. These are, of course, extreme cases. The former case is comparatively easy to solve, although still difficult. The latter case is more difficult and may be impossible to solve.

The concept of the probability of re-identification of respondents is rather subtle, as we are not always dealing with repeatable experiments. The common, traditional definition of the probability of an event as the fraction of times this event would occur if experiments with the same random mechanism were repeated an infinite number of times can therefore not always be used. This is, for instance, the case if we wish to determine the probability that a certain respondent has unique identifying characteristics in the population. In a given population a respondent either has unique characteristics or not. In a given population the probability that a certain respondent has unique identifying characteristics in the population is therefore a useless concept. In such cases we resort to using a superpopulation model, where the population itself is assumed to be generated by means of a certain random mechanism. That random mechanism then provides the probability we are interested in, i.e. in our example the probability that a certain respondent has unique identifying characteristics in the population. In other cases the traditional definition of a probability can be used, for example to determine the probability that a certain person is drawn in a survey sample.

In this chapter we give an overview of the problems for which Statistics Netherlands has attempted to provide a solution. We consider the problems and the outline of their solutions, while technical points are skipped. The choice of the problems and the possible solutions we consider is heavily influenced by the experiences of Statistics Netherlands in the field of SDC.

The approach of Statistics Netherlands to limit the disclosure risk of microdata of social surveys, the only kind of microdata Statistics Netherlands currently releases, differs from approaches that are used by several other statistical offices. In particular, the North-American statistical offices generally use another approach than Statistics Netherlands. These statistical offices usually perturb their microdata of both social and business surveys by adding noise before these microdata are released. Statistics Netherlands, however, usually applies a combination of recoding potentially identifying variables and suppressing remaining potentially dangerous values for its microdata of social surveys. At present it is unclear which approach will prevail in the long run.

The rest of this chapter is organised as follows. Basic concepts are defined in Section 14.2. Preliminaries on SDC for microdata are the subject of Section 14.3. Our basic philosophy of SDC for microdata is discussed in Section 14.4. In Section 14.5 we describe the ideal situation for microdata: in this case we can accurately calculate a probability for each record that this specific record can be re-identified. A somewhat less ideal situation is described in Section 14.6: in this case we can accurately calculate a probability for a data set that an unspecified record can be re-identified. In Section 14.7 we have to face reality: at the moment we do not have a sufficiently good disclosure risk model yet and we have to be satisfied with heuristic arguments. Section 14.8 briefly discusses SDC for tabular data.

In Section 14.9 we discuss some conclusions we can draw and suggest possibilities for future research.

We would like to point out here that our philosophy for microdata as described in this chapter is mainly applicable to microdata of social surveys. For microdata of business data our philosophy is less appropriate (see also Franconi, 1999).

Part of this chapter has appeared in Survey Methodology (De Waal and Willenborg, 1996). This article has been supplemented by material from De Waal and Willenborg (1994c).

For a more complete overview of statistical disclosure control we refer to the following books (in chronological order) Willenborg and De Waal (1996 and 2001), Doyle et al. (2001), and Domingo-Ferrer (2002). Interesting general reports and articles on confidentiality include Citteur and Willenborg (1993), Dalenius (1981), Duncan and Lambert (1986, 1989), Fienberg (1994), Lambert (1993), Marsh, Dale and Skinner (1994), Paass (1988), Skinner et al. (1994), and the 1993 special issue of the Journal of Official Statistics on confidentiality and data access.

14.2. Basic concepts

In this section a number of basic concepts are defined. We will assume that the statistical office wants to release a microdata set containing records of a sample of the population. Each record contains information about an individual entity. Such an entity could be a person, a household or a business enterprise. In the rest of this chapter we will usually consider the individual entity to be a person, although this is not essential.

The two most important concepts in the field of SDC are re-identification and disclosure. Re-identification is said to occur if an attacker establishes a one-to-one relationship between a microdata record and a target individual with a sufficient degree of confidence. Following Skinner (1992) we distinguish between two kinds of disclosure. Re-identification disclosure occurs if the attacker is able to deduce the value of a sensitive variable for the target individual after this individual has been re-identified. Prediction disclosure (or attribute disclosure) occurs if the microdata enable the attacker to predict the value of a sensitive variable for some target individual with a sufficient degree of confidence. For prediction disclosure it is not necessary that re-identification has taken place. Most research so far has concentrated on re-identification disclosure. In this chapter we will use the term disclosure to indicate re-identification disclosure unless stated otherwise.

Now, let us define what is meant by an identifying variable. A variable is called identifying if it can serve, alone or in combination with other variables, to re-identify some respondents by some user of the data. Examples of identifying variables are residence, sex, nationality, age, occupation and education. A subset of the set of identifying variables is the set of direct (or formal) identifiers. Examples of direct identifiers are name, address and public identification numbers. Direct identifiers must have been removed from a microdata set before it is released as else re-identification is very easy. Other identifiers in most cases do not have to be removed from the microdata set. A combination of identifying variables is called a key. The identifying variables that together constitute a key

are also called key variables. A key value is a combination of scores on the identifying variables that together constitute the key.

In the first example given in Section 14.1 the combination of scores *Residence* = *Amsterdam*, *Occupation* = *Mayor* and *Sex* = *Male* forms the key value, in the second example the combination of scores *Residence* = *Urk*, *Occupation* = *Statistician* and *Sex* = *Female* forms the key value.

In practice, determining whether or not a variable is identifying is a problem that can only be solved by sound judgement. No limitative list of intrinsically identifying variables exists, nor, for that matter, an unambiguous and well-defined set of rules to determine such variables. Selecting a set of identifying variables, and therefore of keys, is generally based on subjective assumptions about the population. Statistics Netherlands applies some criteria, like the visibility of the categories of a variable, to determine whether or not a variable is identifying, but these criteria do not provide a definite answer to this problem for all variables. Whether or not a variable is considered identifying is essentially a matter of judgement. In the remainder of this chapter we will assume, however, that a set of keys has been determined.

The counterparts of identifying variables are the sensitive (or confidential) variables. A variable is called sensitive (or confidential) if some of the values represent characteristics a respondent would not like to be revealed about him. In principle, Statistics Netherlands considers all variables sensitive, but in practice some variables are considered more sensitive than others. As in the case of identifying variables, determining whether or not a variable is sensitive can be solved only by sound judgement in practice. The variables *Sexual Behaviour* and *Criminal Past* are generally considered sensitive, but for other variables this may depend on, for instance, cultural background. Keller and Bethlehem (1992) give as an example the variable *Income*. In the Netherlands income is considered sensitive, whereas in Sweden it is not. Moreover, there are variables that should be considered both identifying and sensitive. An example of such a variable is *Ethnic Membership*. However, in the literature it is usually assumed that the identifying and sensitive variables can be divided into disjoint sets. In the remainder of this chapter we will also assume that a set of sensitive variables has been determined which is disjoint from the set of identifying variables.

To end this section, we give a definition of SDC. Statistical disclosure control aims to reduce the risk that sensitive information of individual persons can be disclosed to an acceptable level. What is acceptable depends on the policy of the data releaser. In order to reduce the risk of disclosure an estimate for the risk of disclosure would be very helpful although it is not a necessary requisite (cf. Section 14.7). Ample research has been devoted to defining and estimating this risk of disclosure.

14.3. Preliminaries on SDC for microdata

As a customer of a statistical office, the user of a microdata set should be satisfied with its quality. The user is usually not interested in individual records, but only in statistical results which can be drawn from the total set of records. For instance, he wants to examine tables he has produced himself from the microdata set.

Because a microdata set is meant for statistical analysis it is not necessary that each record in the set is correct. The statistical office has the possibility to perturb records, for example, by adding noise or by swapping parts of records between different records, in order to reduce the risk of re-identification. By perturbing records the risk of re--identification is reduced because even when a correct re-identification takes place the information which is disclosed may be incorrect. In any case the attacker cannot be sure that the disclosed information is correct. The statistical office 'only' has to guarantee that the statistical quality of, for instance, the tables the user wants to examine is high enough. This may be quite complicated to achieve in practice, however.

When local suppression is applied some values of variables in some records are set to 'missing', i.e., deleted from the microdata set. When global recoding is applied some variables are given a coarser categorisation. In a first step, we try to protect a microdata set by means of global recoding. However, if protecting a microdata set entirely by means of global recodings resulted in a considerable information loss, we would apply local suppressions as well. In this way we try to avoid too much information from being lost. It should be clear that local suppressions should only be applied parsimoniously.

An advantage of local suppression and global recoding is that these techniques preserve the integrity of the data in the sense that protected records can satisfy edit rules by imputing appropriate values. A disadvantage of local suppression is that it introduces biased results, because extreme values will be locally suppressed. However, when local suppressions are only applied parsimoniously, this bias will be small.

From the SDC point of view a user of the data should also be looked upon as a potential attacker. Hence, it is useful to consider the ways in which disclosure can take place. An attacker tries to match records from the microdata set with records from an identification file or with individuals from his circle of acquaintances. An identification file is a file containing records with values on direct identifiers and values on some other identifiers in the microdata set. The latter identifiers may be used to match records from the released microdata set with records from the identification file. After matching, the direct identifiers in the identification file can be used to determine whose record has been matched, and the sensitive variables in the released microdata set can be used to disclose information about this person. A circle of acquaintances is the set of persons in the population for which the attacker knows the values on a certain key from the microdata set. So, a circle of acquaintances could actually be an identification file, and vice versa. In the rest of this chapter we will therefore use the terms 'identification file' and 'circle of acquaintances' interchangeably.

In order for re-identification of a record of an individual to occur the following conditions have to be satisfied:

C_1. The individual is unique on a particular key value K.

C_2. The individual belongs to an identification file or a circle of acquaintances of the attacker.

C_3. The individual is an element of the sample.

C_4. The attacker knows that the record is unique in the population on the key K.

C_5. The attacker comes across the record in the microdata set.

C_6. The attacker recognises the record of the individual.

Whenever one of the conditions C_1 to C_6 does not hold, re-identification cannot be accomplished with absolute certainty. If either condition C_1 or C_4 does not hold, then a matching can be made but the attacker cannot be sure that this leads to a correct re-identification. Of course, even in this case there is still a non-zero probability that the attacker successfully re-identifies a respondent.

It is clear from the conditions C_1 to C_6 that a 'good' model for the risk of re-identification should incorporate aspects of both the data set and the user. When a Dutch microdata set is used by someone in, say, China who is essentially unfamiliar with the Dutch population, then the risk of re-identification is negligible. In order to re-identify someone in a microdata set it is necessary to acquire sufficient knowledge about the population. The amount of work that should be done to acquire this knowledge is proportional to the safety of the microdata set.

14.4. A philosophy of SDC for microdata

It seems likely that the attention of a potential attacker is drawn by combinations of identifying variables that are rare in the sample or in the population. Combinations that occur quite often are less likely to trigger his curiosity. If he tries to match records deliberately, then he will probably try to do this for key values that occur only a few times. If the user does not try to match records deliberately, but he knows an acquaintance with a rare key value then a record with that particular key value may trigger him to consider the possibility that this record belongs to this acquaintance. Moreover, the probability of a correct match is higher if the number of persons that score on the matching key value is smaller. Finally, it is also very likely that among the persons that score on a rare key value there are many uniques if the key is augmented with an additional variable. Records that score on such rare combinations of identifying variables are therefore more likely to be re-identified.

In particular key values that occur only once in the population, i.e., uniques in the population, can lead to re-identification. In the past emphasis was placed almost exclusively on uniqueness. It should be noted, however, that uniqueness is neither sufficient nor necessary for re-identification. If a person is unique in the population on certain key variables, but nobody realises this, then this person may never be re-identified. If on the other hand this person is not unique in the population, but there is only one other person in the population with the same key, then this other person is, in principle, able to re-identify him. Furthermore, suppose a person is not unique, but belongs to a small group of people. Suppose also that the attacker happens to know information about him which is not considered to be identifying by the statistical office, but which is contained in the released microdata set, then it is very well possible that he is unique on the key combined with the new information. So, it is possible that a person is re-identified although he is not unique on the keys of identifying variables in the population. Finally, prediction disclosure may occur. That is, if a person is not unique in the population, but belongs to a group of people with (almost) the same score on a particular sensitive variable, then sensitive information can be disclosed about this individual without actual re-identification.

Prediction disclosure is not discussed further in this chapter. For more information on prediction disclosure we refer to Skinner (1992), US Department of Commerce (1978), Duncan and Lambert (1986), and Cox (1986).

SDC should concentrate on key values that are rare in the population. A probability that information from a particular respondent, whose data are included in a microdata set, is disclosed should reflect the 'rareness' of the key value of this respondent's record. A probability for the event that information from an arbitrary respondent is disclosed should reflect the 'overall rareness' of the records in the data set. If there are many records in a microdata set of which the key value is rare, then the probability of disclosure for this data set should be high. In the next sections we will examine some attempts to incorporate these ideas within a mathematical framework.

14.5. Re-identification risk per record

In an ideal world (as far as SDC is concerned) a releaser of microdata would be able to determine a risk of re-identification for each record, i.e., a probability that the respondent of this record can be re-identified. Such a risk per record would enable us to adopt the following strategy. First, order the records according to their risk of re-identification with respect to a single key. Second, select a maximum risk the statistical office is willing to accept. Finally, modify all the records for which the risk of re-identification with respect to the key chosen is too high. Repeat this procedure for each key if there are more keys.

Unfortunately, we do not live in such an ideal world at the moment. However, steps towards the ideal situation have been made by Paass and Wauschkuhn (1985) and Fuller (1993), for instance. In Paass and Wauschkuhn (1985) it is assumed that a potential attacker has both a microdata file, released by a statistical office, and an identification file at his disposal. Between both files there may be many data incompatibilities. These data incompatibilities may be caused by for example, coding errors, by different definitions of categories or by 'noise' in the data. By assuming a probability distribution for these data incompatibilities and a disclosure scenario Paass and Wauschkuhn develop a sophisticated model to estimate the probability that a specific record from the microdata file is re-identified. The type of distribution of the errors that caused the data incompatibilities is assumed to be known to the attacker. The variance of the errors is assumed unknown to him. A potential attacker has to estimate this variance, on the basis of the (assumed) knowledge of the statistical production process. The model of Paass and Wauschkuhn is essentially based on discriminant analysis and cluster analysis.

Paass and Wauschkuhn distinguish between six different scenarios. Each scenario corresponds to a special kind of attacker. The number of records in the identification file and the information content of the identification file depend on the chosen scenario. An example of such a scenario is the journalist scenario, where a journalist selects records with extreme attribute combinations in order to re-identify respondents with the aim of showing that the statistical office fails to secure the privacy of its respondents.

Paass and Wauschkuhn apply their method to match records from the identification file with records from the microdata file. If the probability that a specific record from the identification file belongs to a specific record from the microdata set is high enough, then

these two records are matched. This probability is the probability of re-identification per record, conditional on a particular disclosure scenario.

Müller et al. (1991) and Blien, Wirth and Müller (1992) apply the method recommended in Paass and Wauschkuhn (1985) to real data. When compared to simple matching, i.e., a record is considered re-identified by an attacker if he succeeds in finding a unique value set in the microdata file which is identical to a value set in the identification file, the method suggested by Paass and Wauschkuhn does not turn out to be superior. Apparently, the number of correctly matched records when applying the method by Paass and Wauschkuhn is in disagreement with the actual probability of re-identification per record.

In the context of masking procedures, i.e., procedures for microdata disclosure limitation by adding noise to the microdata, Fuller (1993) obtains an expression for the probability that a specific record in the released microdata set is the same as a specific target record from an identification file. That is, an expression for the re-identification probability per record is derived. To derive this expression several assumptions are made. It is assumed that the data, the noise and errors in the data are normally distributed. Moreover, it is assumed that the covariance matrices of both the noise and the errors in the data are known to an attacker. Finally, it is assumed that the data have been obtained by simple random sampling. These assumptions allow Fuller (1993) to derive his expression for the re-identification probability by means of probability theoretical considerations. Unfortunately, the approach by Fuller has not been tested on real data yet. Hence, it is hard judge the applicability of this approach. For a comment on the approach by Fuller see Willenborg (1993).

Paass and Wauschkuhn (1985), and Fuller (1993) are mainly interested in the effects of noise that has (unintentionally and intentionally, respectively) been added to the data on the disclosure risk. A weak point of their respective approaches is the, implicit, assumption that the key is a high-dimensional one. Assuming a high-dimensional key implies that (almost) everyone in the population is unique. The probability that a combination or key value occurs more than once in the population is negligible. This makes the computation of the probability of re-identification per record considerably easier. On the other hand, in the case of low-dimensional keys it is not unlikely that certain key values occur many times in the population. Therefore, deriving a probability of re-identification per record for low-dimensional keys is much harder than for high-dimensional keys, because for high-dimensional keys the probability of statistical twins in the population is almost zero.

The last few years determining the re-identification risk per record has attracted the attention of other researchers, such as Skinner and Holmes (1998), and Benedetti and Franconi (1998). Despite this research we feel, however, that the research on the re-identification risk is still not mature enough for practical use. In our opinion, a good model for the re-identification risk per record does not appear to exist at the moment. In Section 14.6 we therefore consider less ambitious models, namely models for the re-identification risk per file.

14.6. Re-identification risk per file

In a somewhat less ideal world a releaser of microdata would not be able to determine the risk of re-identification for each record, but he would be able to determine the risk that an

unspecified record from the microdata set is re-identified. In this case, the statistical office should decide on the maximal risk it is willing to take when releasing a microdata set. If the actual risk is less than the maximal risk, then the microdata set can be released. If the actual risk is higher than the maximal risk, then the microdata set has to be modified. Determining which records have to be modified remains a problem, however.

A basic model to determine the probability that an arbitrary record from a microdata set is re-identified has been proposed by Mokken et al. (1989, 1992). In Mokken, Pannekoek and Willenborg (1989) only the case where there is a single researcher, an unstratified population and a single key is considered. It has been extended to include the cases of subpopulations, multiple researchers and multiple keys (cf. Willenborg 1990a; Willenborg 1990b; Mokken et al. 1992). The model of Mokken et al. (1992) takes three probabilities into account. The first probability, f, is equal to the sampling fraction. In other words, f, is the probability that a randomly chosen person from the population has been selected in the sample. The second probability, f_a, is the probability that a specific researcher who has access to the microdata knows the values of a randomly chosen person from the population on a particular key. The third probability, f_u, is the probability that a randomly chosen person from the population is unique in the population on a particular key. Combining these three probabilities, f, f_a and f_u, the probability that a record from a microdata set is re-identified can be evaluated.

For each sample element a number of variables is measured. The values obtained by these measurements (scores) are collected in records, one for each sample element. It is assumed that the variables in the key are either categorical variables or variables for which the measurements fall into a finite number of categories.

Together, the records constitute a data set S that will be made available to a researcher R. We recall that whenever we use the term disclosure in fact re-identification disclosure is meant. The model of Mokken et al. (1989, 1992) does not take prediction disclosure into account.

In terms of the Paass and Wauschkuhn (1985) set-up f_a and f_u together reflect the *Informationsgehalt der Überschneidungsmerkmale,* i.e., the information content of the matching values. The various scenarios they consider differ in terms of f_a and f_u. In particular, f_u is influenced by the number of variables and the information content of these variables, i.e., their categorisation, an attacker has at his disposal to re-identify a record. The parameter f_a is determined by the number of records that are contained in the identification file.

With respect to researcher R and key K there is a circle of acquaintances A. Obviously, A and its size $|A|$ will depend on the particular researcher R as well as on the key K and the variables as registered and coded in the data set.

It is assumed that if conditions C_1, C_2 and C_3 of the conditions for re-identification given in Section 14.3 hold, then conditions C_4, C_5 and C_6 hold too. Condition C_4 is a rather exacting one, but it can be introduced as an assumption for the sake of convenience in formulating a disclosure risk model. Note that it then yields a worst-case situation, in the sense that fallible perception and memory or other sources of ignorance, confusion and uncertainty for a potential intruder are excluded. Taken as an assumption together with C_5 and C_6 the implication is that the occurrence of any unique acquaintance of R in data set S is

equivalent to re-identification by *R*. It is assumed that re-identification of a record implies disclosure of confidential information. Thus re-identification can be treated as equivalent to disclosure. Implicitly, it is assumed in this model that the link between the identifying variables and the sensitive variables has not been disturbed by a technique such as data swapping.

Furthermore, it is assumed that both the identifying and the confidential information are free of error or noise to researcher *R,* contrary to for example, Paass and Wauschkuhn (1985), and Fuller (1993). Clearly, this assumption is unrealistic for most microdata sets.

The disclosure risk D_R for a certain microdata set *S* with respect to a certain researcher *R* and a certain key *K,* is defined to be the probability that the researcher makes at least one disclosure of a record in *S* on the basis of *K*. In order to apply a criterion based on the disclosure risk, the value of this quantity for a given data set has to be determined. An expression for this quantity can be derived on the basis of a set of assumptions.

In the model of Mokken et al. the following assumptions are made in addition to C_1 -C_6:

A_1. The circle of acquaintances *A* can be considered as a random sample from the population.

A_2. The data set *S* is a random sample from the population.

Assumption A_1 serves to imply that the probability that a randomly chosen element from the population is an acquaintance of *R* is $f_a = |A|/N$, where *N* is the size of the population. As a consequence the expected number of unique elements in *A*, $|U_a|$, is equal to $f_a |U| = |A| f_u$ where *U* is the set of unique persons in the population and $|U|$ its size. Obviously assumption A_2 implies that the probability that a specific unique element is selected in the sample is *f*. These assumptions allow one to obtain a very simple expression for the disclosure risk D_R in terms of *f*, f_a, and f_u, namely

$$D_R = 1 - \exp(-Nff_a f_u). \qquad (14.1)$$

Two of the parameters in the model of Mokken et al. (1989, 1992), f_a and f_u, are unknown. The parameter f_a can be 'guestimated', i.e., obtained by inspired guesswork, by assuming different scenarios an attacker may follow. A number of such scenarios has been described in Paass and Wauschkuhn (1985) and Paass (1988). Evaluating f_a seems difficult, however. In order to estimate the other parameter, f_u, a number of models has been proposed in the literature. Models to estimate the number of uniques in the population, and hence f_u, that have been proposed include the Poisson-gamma model (Bethlehem, Keller and Pannekoek, 1989; Mokken, Pannekoek and Willenborg, 1989; Willenborg, Mokken and Pannekoek, 1990; De Jonge, 1990; Rinott, 2003), the negative binomial superpopulation model (Skinner et al, 1990; Benedetti and Franconi, 1998; Rinott, 2003), the Poisson-lognormal model (Skinner and Holmes, 1992; Hoogland, 1994), models based on equivalence classes (Greenberg and Zayatz, 1992) and models based on modified negative binomial-gamma functions (Crescenzi, 1992; Coccia, 1992). As we have remarked in Section 14.4 not only the number of population uniques is important, but the numbers of cells with two, three, etc. persons are important as well. The Poisson-gamma model, the Poisson-lognormal model and the negative binomial superpopulation model can be applied to estimate the

number of cells with two, three, etc. persons as well. It seems that the other models mentioned above can also be extended in order to estimate these numbers. A major drawback is that the results are not very reliable in many cases.

From the model by Mokken et al. (1989, 1992) it is clear that the statistical office that disseminates the data is able to influence the risk of re-identification. The statistical office basically has two ways to do this. First of all, the size of the data set can be reduced, i.e., the sampling fraction f can be reduced. A reduction of f implies a reduction of the risk. However, lowering f is generally undesirable, because usually f has to be reduced substantially to be effective. This implies that only a small part of the data available can be released. The second way in which the statistical office can influence the re-identification risk is by reducing the number of population uniques, i.e., by reducing f_u, The fraction f_u depends on the information provided by the key variables. The less information the key variables provide, the less uniques there are in the population. In order words, f_u can be reduced by collapsing categories (global recoding) and by replacing values by missings (local suppression). Collapsing categories is a global action, because it generally affects many records; replacing values by missings is a local action because it affects only a few individual records. Usually, the loss in information when reducing f_u is considerably less than the loss in information when reducing f. Therefore, a statistical office will usually choose to control the re-identification risk by reducing f_u rather than reducing f. The third possibility of controlling the re-identification risk, i.e., by reducing f_a, is not applied in practice, because f_a is difficult to model.

Although the model by Mokken et al. (1989, 1992) provides some insight in how to reduce the disclosure risk it can hardly be used as a basis for the protection of microdata sets. The reason for this is that the two parameters of the model, f_u and f_a, are often difficult to evaluate. Usually there is insufficient data available to estimate f_u and f_a accurately. We conclude that even a model for a re-identification risk for an entire microdata set is difficult to apply in practice at the moment. In Section 14.7 we therefore face reality in which we have no satisfactory model (yet) for either the re-identification risk per record or re-identification risk for an entire microdata set.

De Waal (1994a) examines the worst-case populations, i.e. the populations with the highest expected number of population-uniques, using a simple urn model. This model provides some insight into what populations are the worst as far as statistical disclosure is concerned. The model is explored in detail in Chapter 15.

We end this section by noting that record linkage techniques have also been used to assess the re-identification risk per file. We refer the interested reader to, for example, Winkler (1998), and Domingo-Ferrer, Mateo-Sanz and Torra (2001).

14.7. Intuitive re-identification risk

In reality we are, unfortunately, still forced to base SDC on heuristic arguments rather than on a solid theoretical basis. The SDC rules mentioned in this section all reduce the re-identification risk. It is, however, not possible to evaluate this reduction of the re-identification risk. At Statistics Netherlands, rules for SDC of microdata of social surveys are based on testing whether scores on certain keys occur frequently enough in the population. Microdata sets of business surveys are not released by Statistics Netherlands at

the moment. A few problems arising for protecting microdata of social surveys are the determination of the keys that have to be examined, the way to estimate the number of persons in the population that score on a certain key, to make the meaning of the phrase 'frequently enough' operational by determining for example, (a) threshold value(s), and how to determine appropriate SDC-measures.

Statistics Netherlands distinguishes between two kinds of microdata sets for social surveys. The first kind is a so-called public use file. A public use file can be obtained by everybody. The keys that have to be examined for a public use file are all combinations of two identifying variables. The number of identifying variables is limited, and certain identifying variables, such as place of residence are not included in a public use file. Moreover, sampling weights have to be examined before they can be included in a public use file, because there are many situations in which weights can give additional information (see Chapter 19, and De Waal and Willenborg, 1995a, 1997). For instance, when a certain subpopulation is oversampled then this subpopulation can be recognised by the low weights associated with its members in the sample. Weights may only be published when they do not provide additional information that can be used for disclosure purposes. In case sampling weights are not considered suited for publication, SDC measures should be taken, such as subsampling the units with a low weight in order to get a subsample in which all units have approximately the same weight. Because the weights are then approximately equal, assuming that they are exactly equal would introduce only a small error.

The second kind of microdata set for social surveys released by Statistics Netherlands is a so-called microdata set for research. A microdata set for research can only be obtained by well-respected (statistical) research offices. The information content of a microdata set for research is much higher than that of a public use file. The number of identifying variables is not limited and an identifying variable such as place of residence may be included in a microdata set for research. Because of the high information content of a microdata set for research, researchers have to sign a declaration stating that they will protect any information about an individual respondent that might be disclosed by them. The keys that have to be examined for a microdata set for research consist of three-way combinations of variables describing a region with variables describing the sex, ethnic group or nationality of a respondent with an ordinary identifying variable.

The rules Statistics Netherlands applies for SDC of microdata of social surveys are based on the following idea: a key value, i.e., a combination of scores on the identifying variables that together constitute the key, is considered safe for release if the frequency of this key value in the population is more than a certain threshold value d_0. This value d_0 was chosen after a careful and extensive search considering many different values and comparing the records that have to be modified for each value of d_0. The value that leads to the 'most likely' set of records that have to be modified has been chosen to be the value of d_0. Which records are considered to be the 'most likely' ones to be modified is a matter of personal judgement.

When applying the above rules for either public-use files or microdata for research we are generally posed with the problem that we do not know the number of times that a key value occurs in the population. We only have the sample available to us. The population

frequency of a key value has to be estimated based upon the sample. For large regions it is possible to use an interval estimator to test whether or not a key value occurs often enough in a region. This interval estimator is based on the assumption that the number of times that a key value occurs in the population is Poisson distributed (cf. Pannekoek, 1999). However, for relatively small regions the number of respondents is low, which causes the estimator to have a high variance which in turn causes a lot of records to be modified. To estimate the number of times that a key value occurs in a small region we therefore suggest applying a point estimator. We will now discuss some possibilities for such an estimator.

A simple point estimator for the number of times that a certain key value occurs in a region is the direct point estimator. The fraction of a key value in a region *i* is estimated by the sample frequency of this key value in region *i* divided by the number of respondents in region *i.* The population frequency is then estimated by this estimated fraction multiplied by the number of inhabitants in region *i.* When the number of respondents in region *i* is low, which is often the case, the direct estimator is unreliable. Another point estimator is based on the assumption that the persons who score on a certain key value are distributed homogeneously over the population. In this case the fraction of a key value in region *i* can be estimated by the fraction in the entire sample. The advantage of this, so-called, synthetic, estimator is that the variance is much smaller than the variance of the direct estimator. Unfortunately, the homogeneity assumption is usually not satisfied which causes the estimator to be biased. However, a combined estimator can be constructed with both an acceptable variance and an acceptable bias by using a convex combination of the direct estimator and the synthetic estimator. Such a combined estimator has been tested by Pannekoek and de Waal (1995 and 1998). See Chapter 16 for more information.

Another practical problem that deserves attention is top-coding of extreme values of continuous (sensitive) variables. These extreme values may lead to re-identification because these values are rare in the population. At the moment Statistics Netherlands uses an interval estimator to test whether there is a sufficient number of individuals in the population who score on a 'comparable' value of the continuous variable (cf. Pannekoek, 1992). If this is the case, then the extreme value may be published, otherwise the extreme value must be suppressed. In order to apply this method in practice it remains to specify what is meant by 'sufficient' and by 'comparable'.

Some important practical problems occur when determining which protection measures should be taken when a microdata set appears to be unsafe. In that case the original data set must be modified in such a way that the information loss due to SDC-measures is as low as possible while the resultant data set is considered safe. In De Waal and Willenborg (1994a) and De Waal and Willenborg (1995b, 1998) a model for determining the optimal local suppressions is presented. See Chapter 17 for more details regarding this model. Determining the optimal global recodings is much more difficult. Comparing the information loss due to global recodings to the information loss due to local suppressions is already a problem. In De Waal and Willenborg (1995c) and De Waal and Willenborg (1999b) this latter problem is solved by using methods based on entropy concepts. See Chapter 18 for more information on measuring the information loss due to local suppression, global recoding and data perturbation.

14.8. Statistical disclosure control for tables

Between statistical disclosure control for microdata and tables there are many similarities. For instance, when trying to reduce the risk of disclosure one usually starts by modifying the identifying variables. In the case of microdata one collapses the categories of an identifying variable; in the case of tabular data one collapses two columns or rows of the table. After the global modifications have been made local modifications must be made. In the case of microdata of social surveys values of identifying variables in some records have to be set to 'missing', in the case of tabular data values of sensitive cells have to be set to 'missing'. Here we also see a striking difference between statistical disclosure control for microdata of social surveys and tabular data. In the case of microdata of social surveys values of *identifying variables* are suppressed, whereas in the case of tabular data values of *sensitive variables* are suppressed. Another important difference between disclosure control for microdata and tabular data is the number of identifying variables that is involved. In the case of microdata there are generally many identifying variables, whereas in the case of tabular data there are only a few identifying variables.

In the literature on statistical disclosure control for tabular data it is usually assumed that the tables that are published are based on an observation of the entire population. The disclosure problem of tabular data in the case that only a sample of the population is observed is hardly discussed. Some thoughts on this subject would be welcome. In the sequel we will, however, make the usual assumption that the tables are based on an observation of the entire population.

After some columns and/or rows have been collapsed it is necessary to make some local modifications. A well-known technique to modify data in a table in order to safeguard this table against disclosure is cell suppression. First of all, this technique tries to identify which cells in a table contain information that is to be considered sensitive.

The most common way to determine whether a cell is considered sensitive is by means of a dominance rule. A dominance rule states that if the values of the data of a certain number of respondents, say 3, constitute more than a certain percentage, say 75%, of the total value of the cell, then this cell has to be suppressed. The main idea on which this approach is based is the following. If a cell is dominated by the value of one respondent, then his contribution can be estimated fairly accurately. In particular, if there is only one respondent then his contribution can be disclosed exactly. If the value of a cell is dominated by the contributions of two respondents, then each of these respondents is able to estimate the value of the contribution of the other one quite accurately. In particular, if there are exactly two respondents then these respondents can disclose the contribution of the other. If there are n respondents then n-1 of them, pooling their information, can disclose information about the value of the data of the remaining respondent. For small n, say, 2, 3 and 4, this poses a problem.

Apart from dominance rules other rules for determining sensitive cells have been suggested and applied in practice. An example of such a rule is the prior-posterior rule (cf. Cox, 1981; Geurts, 1992). This rule uses two parameters, p and q with $p < q$. It is assumed that every respondent can a priori estimate the contribution of each other respondent to within q percent of its respective value. After a table has been published the information of the respondents changes and they may be able to make a better (a posteriori) estimate of

the contribution of another respondent. A cell is considered sensitive if it is possible to estimate the contribution of an individual respondent to that cell to within *p* percent of the original value. A cell that is considered sensitive has to be suppressed.

The suppression of a cell value because the content of this cell is considered sensitive according to, for example, a dominance rule is called primary suppression. Primary suppression alone is in many cases not sufficient to obtain a table that is safe for release. In a table the marginal totals are often published as well as the values of the internal cells. A cell value that has been suppressed can then be computed by means of the marginal totals. Therefore, other cell values have to be suppressed in order to avoid this possibility. This is called secondary suppression. Secondary suppression can be done in many different ways. Usually secondary suppression aims to optimise some target function. For instance, one could try to minimise the number of respondents whose data are suppressed in the table or one could try to minimise the total value of the data that are suppressed. Selecting the 'best' target function is very hard and is partly based on subjective considerations.

Secondary suppression causes other problems as well. Although it might be impossible to compute the exact values of suppressed cell values in a table after secondary suppression, it is still possible to compute ranges in which the values of the cells lie (cf. Geurts, 1992). To compute such ranges one can use all available information about the table, such as, for example, that the cell values of the table at hand are all non-negative. If these ranges of feasible values are small, then an attacker is able to obtain good estimates for the suppressed cell values. Therefore, secondary suppression must be done in such a way that the ranges in which the suppressed cell values lie are not too small.

Additional information is another problem when reducing the risk of disclosure for a particular table. For example, suppose that the following dominance rule is used: a cell is suppressed if at least 80% of the value is the combined result of the data of 2 companies. Suppose, furthermore, that all the companies are in the sample. Now, suppose that there are three companies contributing to a certain cell and that the data of the largest company constitutes 50% of the value of this cell. If the cell is not suppressed, then this company can deduce that the data of the second largest company constitutes between 25% and 30% of the total value of the cell. On the other hand, if a cell value is primarily suppressed and the largest company contributing to that cell happens to know that the value was suppressed because it was considered sensitive and also happens to know the parameters of the dominance rule, then the largest company can deduce that the data of the second largest company constitutes between 30% and 50% of the total value of the cell. Absolute secrecy about the parameters of the dominance rules is the first step to avoid these problems.

Three- and higher-dimensional tables and 'linked' tables pose a lot of theoretical problems (cf. De Vries, 1993, and De Waal and Willenborg, 1999a). The theory for these tables is much more difficult than for ordinary two-dimensional tables. For secondary suppression in such tables several heuristics and exact algorithms have been proposed, but further research remains to be carried out in order to perfect these algorithms. For the latest, state-of-the-art exact algorithms we refer to Fischetti and Salazar-González (1998b, 2000).

Another well-know technique which is also applied at Statistics Netherlands to protect a table against disclosure is rounding. In our opinion, the most interesting way of rounding is controlled rounding (cf. Fellegi, 1975; Cox and Ernst, 1983; Fischetti and Salazar-

González, 1998a). The main advantages of controlled rounding compared to conventional rounding and random rounding is that the additivity of the tables is preserved, i.e. after rounding the rows and columns still add up to their rounded marginal totals, and that the user can choose an information measure which will be minimised. At the moment controlled rounding for two-dimensional tables does not provide serious problems any more (cf. Cox and Ernst, 1983). However, controlled rounding of higher-dimensional tables is a difficult problem. In some cases the problem is impossible to solve (cf. Cox, 1987). Some heuristics to deal with three-way tables have been developed at the U.S. Bureau of the Census (see Fagan, Greenberg and Hemming, 1988).

Note that rounding is a formalised way of adding noise to the data. Only for the 'sensitive' cells is it necessary to add this noise. For some other cells noise has to be added in order to preserve the additivity of the table. In other words, noise is added to those cells as a way to compensate for the noise that has been added to the sensitive cells. For still other cells it is not necessary at all to add noise. So, when we apply rounding not all the cells have to be rounded. Apart from the sensitive cells we are free to choose which cells we are going to round. This remark shows that we have a lot of freedom to create a satisfactory heuristic for rounding. This freedom to choose which additional cells have to be rounded apart from the sensitive cells is similar to the freedom when choosing cells for secondary suppression.

14.9. Discussion

There is one important conclusion one can draw from this chapter: SDC still offers a lot of possibilities for future research, despite the considerable amount of research that has been carried out to date. An excellent overview of current research problems on SDC is given by Giessing and Hundepool (2001). Most of the research topics mentioned in that paper are tackled in the so-called CASC project, a large international research project on SDC. The list of research topics given below is partly based on Giessing and Hundepool (2001).

The theory of SDC for microdata has a number of gaps. Among the technical problems that remain to be solved are the following:

- The determination of the number of uniques, or more generally the number of rare frequencies, in the population (for a worst-case scenario, see Chapter 15). Some of the models proposed in Section 14.6 appear to be acceptable, but can probably be improved upon. An alternative approach is to estimate which elements in the sample are unique in the population (see e.g. Verboon, 1994). Extending the model by Mokken et al. (1989, 1992) to estimate the risk of re-identification for a file is yet another subject to be tackled. This extension should take into account that measurement errors have been made and that population uniqueness is not necessary in order to disclose information. This problem is not examined in the rest of this book although this topic has attracted considerable attention during the last few years, see, for example, Winkler (1998), Franconi (1999), Domingo-Ferrer, Mateo-Sanz and Torra (2001), Rinott (2003), and Shlomo (2003)
- The construction of an estimator for the number of times that a key value occurs in small areas. Such an estimator is difficult to construct, although the preliminary results obtained at Statistics Netherlands seem encouraging (see Chapter 16 for the construction of such an estimator).

- The determination of appropriate global recodings and local suppressions. We refer to Hurkens and Tiourine (1998b) and Chapter 17 for more information.
- The calculation of information loss due to local suppressions and global recodings (see Chapter 18 for an information loss model based on entropy).
- The estimation of the re-identification risk per record. In fact, this would yield a sound criterion to judge the safety of a microdata set. This criterion can guide one in producing safe microdata sets by applying SDC-measures such as global recoding and local suppression. This problem is not examined in the remainder of this book. Some interesting recent papers are Skinner and Holmes (1998), and Benedetti and Franconi (1998).
- The protection of microdata of business surveys by means of perturbative techniques. Microdata of business surveys differ substantially from microdata of social surveys, for example, because microdata of business surveys are often skewly distributed. They cannot be protecting in the same manner as microdata of social surveys. Instead one has to resort to perturbative techniques. In recent years, ample research has been dedicated to such perturbative techniques, both at Statistics Netherlands and in the rest of the world. Some important research papers on this topic are Kim (1986), Sullivan (1989), and Moore (1996a, 1996b). At Statistics Netherlands, research on perturbative techniques for microdata has focussed on the so-called Post RAndomisation Method (PRAM). We refer to Kooiman, Willenborg, and Gouweleeuw (1997), De Wolf, Gouweleeuw, Kooiman and Willenborg (1997), and Gouweleeuw et al. (1998) for more information on PRAM. Perturbative techniques for microdata are not considered in this book.
- The protection of microdata of business surveys by means of microaggregation. Microaggregation is a technique to combine several individual records into a single, micro-aggregated record. Microaggregation can be carried out either univariately or multivariately. It offers an alternative for the above-mentioned perturbative techniques. For information on microaggregation we refer to, for example, Defays and Nanopoulos (1993), Domingo-Ferrer and Mateo-Sanz (2002), and Sande (2001 and 2002). Microaggregation is not examined in this book.
- The construction and release of synthetic microdata. In order to avoid disclosure of sensitive information several researchers have proposed to release carefully constructed synthetic microdata instead of the original microdata (see for example Rubin, 1993; Fienberg, 1998; Dandekar, Cohen and Kirkendall, 2001). The feasibility of such an approach has been doubted by other researchers (see Kooiman, 1998). The construction of synthetic microdata is outside the scope of this book.

For tabular data there are a number of important problems to be solved:

- The development of alternative techniques to cell suppression. Fischetti and Salazar-González (1998c) propose to publish intervals instead of suppressing values. This simplifies the mathematical problem of protecting a table against disclosure considerably. This approach is not examined in this book.

- The extension of the theory of statistical disclosure control for single two-dimensional tables to higher-dimensional tables and 'linked' tables against disclosure. For this purpose heuristics should be developed. We refer to De Wolf (1999) for a heuristic to protect high-dimensional tables against disclosure by means of cell suppression. High-dimensional tables are not explicitly considered in this book.
- The development of cell suppression algorithms that lead to absolutely safe tables according to the sensitivity measure that is applied (see for example Fischetti and Salazar-González, 1998b, 2000, and Chapter 20 of the present book).

Finally, more research should be carried out with respect to the possibility of remote access to statistical data. With remote access, the data, either microdata or tabular data, remain at the statistical office and users request certain statistical analyses to be performed, for example by supplying an SPSS script to the statistical office. Such scripts are screened to detect whether they would provide the user with too much sensitive and identifying information, given the information this user has already received from the statistical office. Remote access is not examined in this book. We refer the reader to, for example, De Boer, Schouten and Willenborg (2000), Duncan and Mukherjee (2000), and Shlomo (2003).

The algorithms for protecting microdata and tabular data should be incorporated into a software framework that can deal with the time-consuming calculations. For microdata, software must be developed to indicate which records and variables must be modified, and how they should be modified, when applying a particular disclosure rule. For tabular data software must be developed to perform cell suppression and (controlled) rounding when a dominance rule or another kind of cell sensitivity rule is specified. Statistics Netherlands has developed such a software framework called ARGUS. In fact, ARGUS consists of two separate parts: μ-ARGUS for protection of microdata and τ-ARGUS for protection of tabular data. We will occasionally refer to ARGUS in subsequent chapters. μ-ARGUS is based on the framework described in Section 14.7, τ-ARGUS on the framework of Section 14.8. For more information on ARGUS see for example De Jong (1992), De Waal and Willenborg (1994b), Van Gelderen (1995), Pieters and De Waa1 (1995), De Waal and Pieters (1995), Giessing and Hundepool (2001), and Hundepool et al. (2002a and 2002b).

15. The Maximum Expected Number of Unique Individuals in a Population

15.1. Introduction

A central problem in the theory of statistical disclosure control is the determination of the probability that an individual with certain identifying features is unique in the population. The reason for this is that individuals with unique identifying features are (relatively) easy to recognise. Therefore, to estimate the risk of disclosure that is involved when microdata are released it is useful to estimate the number of unique individuals in the population. In this chapter an upper bound on the expected number of unique individuals in a population is derived for a given disclosure key.

For convenience the problem is stated in terms of urns and balls instead of identifying features and individuals. Individuals correspond to balls and identifying features to urns. Each individual of the population has a probability p_j to have identifying feature j (i.e. a specific score on a disclosure key). In terms of balls and urns: each ball has a probability p_j to be assigned to urn j. The balls are assigned to the urns independently. Our problem is to find the probability distribution for which the expected number of individuals with unique identifying features is maximal. In other words, we want to find the probability distribution for which the expected number of urns with exactly one ball is maximal.

Urn models are quite common in statistical disclosure control to assess the disclosure risk of a microdata set. For examples of such urn models in literature we refer to Chen and Keller-McNully (1998), Samuels (1998) and Fienberg and Makov (2001). We are the first to consider the specific problem of this chapter, and hence also the first to propose a solution to it.

In Section 15.2 of this chapter our problem is stated in mathematical terms. In Section 15.3 some consequences of the Lagrangean corresponding to our objective function and constraints are examined. Studying the derivative of this Lagrangean yields us some interesting insights. Section 15.4 and Section 15.5 are rather technical. From these two sections an important result follows, namely that there are (at most) two possible solutions. In Section 15.6 bounds are derived in order to be able to compare the results for the two possible solutions without explicitly determining them. Numerical results are presented in Section 15.7. A short summary of the solution obtained is given in Section 15.8.

This chapter has appeared in Kwantitatieve Methoden (De Waal, 1994a).

15.2. The problem

Suppose we have m urns and n balls. Each ball is assigned to an urn independently. The probability to assign a ball to the j-th urn is p_j. The expected number of urns with exactly

one ball can be expressed as a function of the probabilities p_j. We are interested in the following problem: "How should the probabilities p_j be chosen in order to maximise the expected number of urns with exactly one ball?".

In Section 15.1 we already noted that this problem is equivalent to a problem in the theory of statistical disclosure control. If we let the possible identifying features correspond to the urns and the individuals to the balls, then we see that a solution to our problem provides an upper bound on the number of unique individuals in the population. Here we assume that the identifying features are distributed independently. This assumption is not always justified in practice.

It is easy to calculate the expected number of urns with exactly one ball. This number is given by

$$E = \sum_{j=1}^{m} np_j(1-p_j)^{n-1} . \tag{15.1}$$

In the rest of this chapter we will also use the function *N* defined by E/n. We will refer to each of these functions as the target function. We hope that this will not confuse the reader too much. The p_j must be larger than, or equal to, zero. They must also sum to unity. These constraints are expressed by

$$p_j \geq 0 , \text{ for all } j = 1,\ldots,m, \tag{15.2}$$

and

$$\sum_{j=1}^{m} p_j = 1 . \tag{15.3}$$

15.3. The Lagrangean

In order to find the maximum of (15.1) subject to (15.2) and (15.3) we begin by determining the Lagrangean $L(p_1,...,p_m,\lambda)$,

$$L(p_1,...,p_m,\lambda) = \sum_{j=1}^{m} np_j(1-p_j)^{n-1} - \lambda\left(\sum_{j=1}^{m} p_j - 1\right). \tag{15.4}$$

By differentiating *L* with respect to λ we obtain that the sum of the p_i is equal to one. By differentiating with respect to p_i we obtain

$$n(1-p_i)^{n-1} - n(n-1)p_i(1-p_i)^{n-2} = \lambda , \tag{15.5}$$

or alternatively

$$(1-p_i)^{n-2}(1-np_i) = \lambda/n . \tag{15.6}$$

This must hold for all *i*. Therefore we can conclude that the optimal p_i obey

$$(1-np_i)(1-p_i)^{n-2} = (1-np_j)(1-p_j)^{n-2}. \tag{15.7}$$

This relation must hold for all *i* and *j*.

Relation (15.7) suggests that it is useful to study the behaviour of the function defined by

$$f_n(x) = (1-nx)(1-x)^{n-2},\ 0 \le x \le 1. \tag{15.8}$$

We can make the following observations about this function:

1. $f_n(0) = 1$;
2. $f_n(1) = 0$;
3. $f_n(1/n) = 0$;
4. $f_n'(x) = (n-1)(nx-2)(1-x)^{n-3}$;
5. $f_n'(x) = 0$ if $x = 2/n$ or $x = 1$;
6. $f_n'(x) < 0$ if $x < 2/n$;
7. $f_n'(x) > 0$ if $2/n < x < 1$.

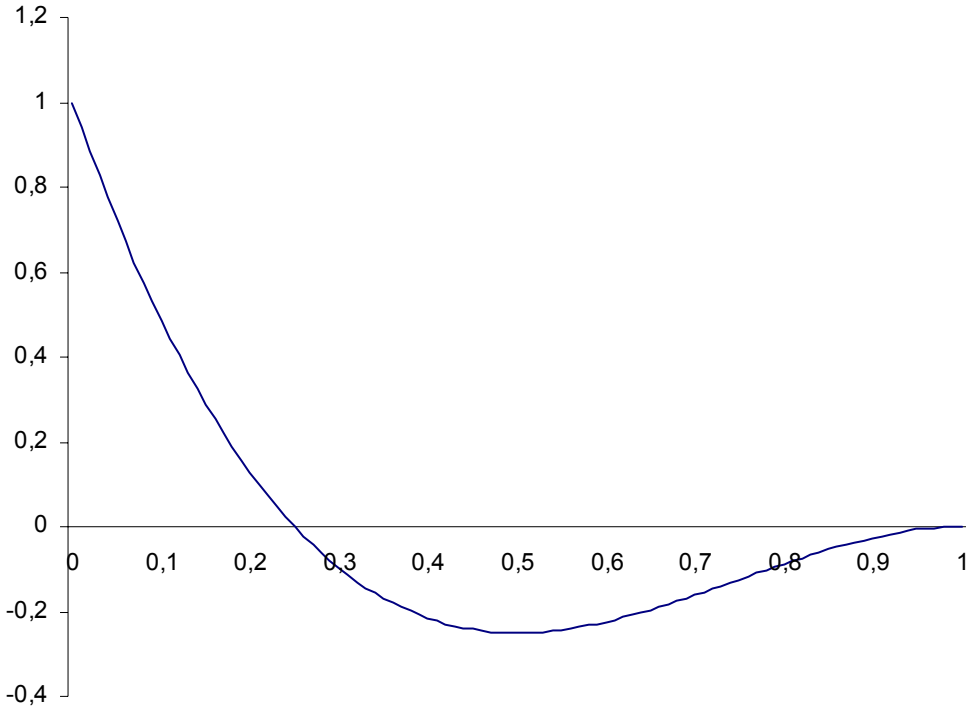
This implies that the equation $f_n(x) = C$, where C is a constant has the following solutions for $0 \le x \le 1$:

8. If $0 < C \le 1$, there is only one solution. For this solution x_0 we have: $0 \le x_0 < 1/n$. In order words, for $0 \le x < 1/n$ the function f_n is injective.
9. If $f_n(2/n) < C \le 0$, there are two solutions x_1 and x_2 between 0 and 1. For these solutions we have: $1/n \le x_1 < 2/n$ and $2/n < x_2 \le 1$. In other words, for $1/n < x < 1$ the function f_n is not injective.
10. If $C = f_n(2/n)$, there is only one solution: $x = 2/n$.

This reveals that the optimal p_i can have at most two different values. Moreover, we know that when the optimal solution has two different p_i-values, then one value lies between 1/*n* and 2/*n* and the other is larger than 2/*n*. We also know that if one p_i is smaller than 1/*n*, then all the p_i have the same value. This implies that if $m \ge n$, the optimal solution is given by $p_i = 1/m$ for all *i*. From now on we therefore assume that $n > m$.

In order to have some visual understanding for observations 8, 9 and 10, we have plotted function f_n. In Figure 15.1 the function f_n is drawn for the case *n* = 4. From Figure 15.1 one can clearly see that observations 8, 9 and 10 hold in this case.

Figure 15.1. The function $f_4(x) = (1-4x)(1-x)^2$.



15.4. Parameterisation of conjugated values

We know that the solutions of the set of equations given by (15.7) have at most two different values for $0 \le p_i \le 1$. From now on we will call these two values conjugated values. In this section a parameterisation of conjugated values is derived. To simplify the notation somewhat we use $1-p_i$ instead of p_i in this section.

Suppose we have two conjugated values $p_1 = 1-Q$ and $p_2 = 1-R$. We suppose that $R = \mu Q$ ($0 \le \mu \le 1$). From the set of equations (15.7) we obtain relation (15.9) between R and Q:

$$nQ^{n-1} + (1-n)Q^{n-2} = nR^{n-1} + (1-n)R^{n-2} . \qquad (15.9)$$

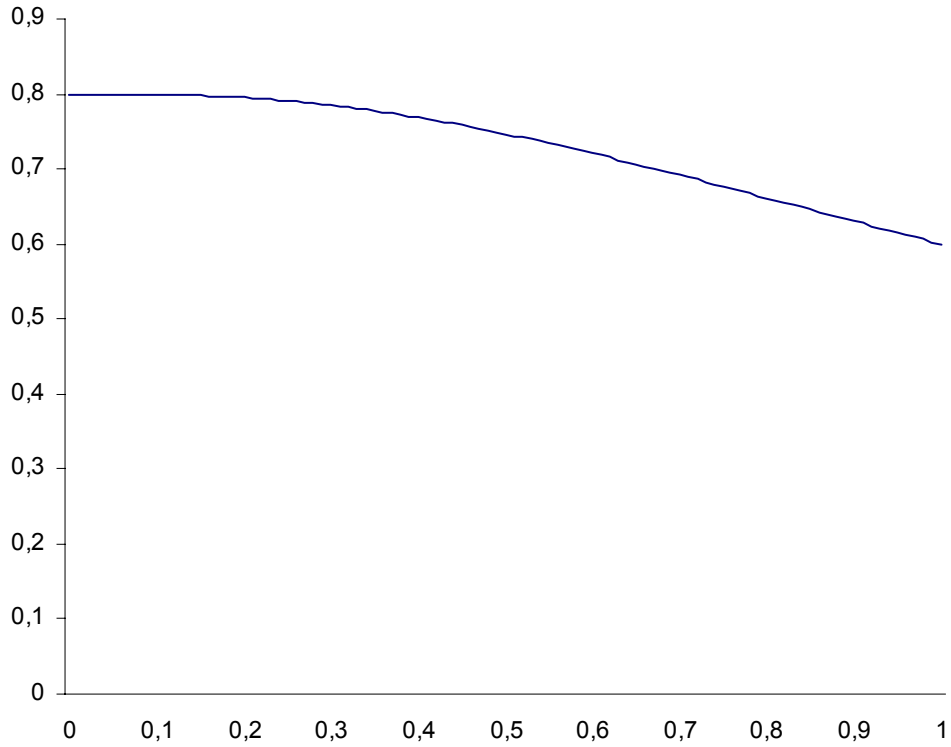
If we substitute $R = \mu Q$ in (15.9), then we find a parameterisation of Q in terms of μ.

$$Q = \frac{(n-1)(1-\mu^{n-2})}{n(1-\mu^{n-1})}, \qquad 0 \le \mu \le 1. \tag{15.10}$$

If $\mu = 0$, then Q is equal to (n-1)/n. The associated probability p_1 is therefore equal to 1/n. R is equal to 0 if $\mu = 0$. The associated probability p_2 is equal to 1. When μ approaches 1 then Q tends to (n-2)/n. The associated probability p_1 tends to 2/n. R tends to (n-2)/n when μ approaches 1. The associated probability p_2 tends to 2/n.

The behaviour of function Q is maybe a bit difficult to understand without some visual aid. In Figure 15.2 the function Q is drawn for the case n=5.

Figure 15.2. The function $Q = \frac{4(1-\mu^3)}{5(1-\mu^4)}$



The derivative of Q with respect to μ is given by

$$\frac{dQ}{d\mu} = \frac{(n-1)}{n} \frac{(n-1)(1-\mu^{n-2})\mu^{n-2} - (n-2)(1-\mu^{n-1})\mu^{n-3}}{(1-\mu^{n-1})^2}. \qquad (15.11)$$

This derivative is less than 0 if $0 \le \mu \le 1$. The parameterisation of Q and R by means of μ is therefore 1-1. The derivative of R with respect to μ is larger than 0 if $0 \le \mu \le 1$.

15.5. The number of urns with the same probability

15.5.1. An important set of equations involving conjugated values

Now we make use of the fact that there are at most two (conjugated) values p and q for the optimal probabilities. We suppose that there are z urns with probability p and m-z urns with probability q. Implicitly we hereby assume that $n > m$ and therefore (cf. points 8 and 9 in Section 15.3) that both probabilities are larger than $1/n$. Note that $z = m$, or $z = 0$, corresponds to the situation that all probabilities are equal. As the reader will remember we have already established that the optimal probabilities are all equal to $1/m$ if $n \le m$.

For p and q relation (15.12) and relation (15.13) below hold.

$$(1-p)^{n-2}(1-np) = (1-q)^{n-2}(1-nq) \qquad (15.12)$$

and

$$zp + (m-z)q = 1. \qquad (15.13)$$

For the moment we do not demand z to be an *integer* between 0 and m. Instead z may be any *real* value between 0 and m. We will first solve the problem for z assumed to be a real value, and later we will modify this solution to obtain the solution for z being integer.

15.5.2. The solutions of the equations

In this section we show that the set of equations (15.12) and (15.13) has at most three different pairs of solutions $(p_i(z), q_i(z))$. These pairs are differentiable with respect to z. The proof of this latter statement is elementary, but rather long and tedious. Therefore, we do not go into all the details of the proof.

The first solution is, of course, given by $p(z) = q(z) = 1/m$. From now on we concentrate on the case that $p(z)$ is unequal to $q(z)$. Without loss of generality we assume that $p(z) < q(z)$. Instead of relation (15.13) we use the following relation

$$z(1-p) + (m-z)(1-q) = m-1. \qquad (15.14)$$

For (1-p) we can substitute the expression given in (15.10), and for (1-q) we can substitute μ times that expression. So, the problem of finding solutions to the set of equations (15.12) and (15.13) translates into the problem of finding the roots of the function $h(\mu)$ defined by

$$h(\mu) = z\frac{n-1}{n}(1-\mu^{n-2}) + (1-z)\frac{n-1}{n}\mu(1-\mu^{n-2}) - (m-1)(1-\mu^{n-1}) \qquad (15.15)$$

for $0 \le \mu < 1$. Now we will list some properties of the function *h*.

It is easy to see that for $\mu = 0$ and $\mu = 1$ we have

1. $h(0) = z\dfrac{n-1}{n} - (m-1)$,

2. $h(1) = 0$.

The function $h(\mu)$ can be studied by examining its first and second derivative. The first derivative is still a complicated expression, but for $\mu = 0$ and $\mu = 1$ we obtain two simple terms:

3. $h'(0) = (m-z)\dfrac{n-1}{n}$,

4. $h'(1) = (2m-n)\dfrac{n-1}{n}$.

The second derivative is given by:

5. $h''(\mu) = (n-2)(n-1)\left(\dfrac{m-1}{m} - (1-z)\dfrac{n-1}{n}\right)\mu^3 - \left(\dfrac{(n-3)(n-2)(n-1)}{n}z\right)\mu^4$.

So, it is very easy to determine when $h''(\mu)$ is positive and when it is negative.

Combining these, and other, facts about the function $h(\mu)$ we are able to draw the following conclusions.

- If $n \ge 2m$, then $h(\mu)$ has one root between 0 and 1
- If $n < 2m$, then $h(\mu)$ has at most two roots between 0 and 1

The first case is not very interesting. We only note that the pair $(p(z),q(z))$ associated to the root $\mu(z)$ is the optimal solution for given value of *z*. To see this we consider the target function $N(p,q)$, which is of course defined by

$$N(p,q) = (m-1)p(1-p)^{n-1} + q(1-q)^{n-1}, \qquad (15.16)$$

where $q = 1-(m-1)p$. We are seeking the maximum of this function. One possible solution is the pair $(p(z),q(z))$ associated to $\mu(z)$. The second derivative of $N(p,q)$ with respect to *p* is given by

$$\frac{\partial^2 N}{\partial p^2} = (m-1)(n-1)((np-2)(1-p)^{n-3} + (m-1)(nq-2)(1-q)^{n-3}). \qquad (15.17)$$

This function is positive for the only other possible solution, $p = q = 1/m$, if $n > 2m$. Therefore, the target function has a local minimum for $p = q = 1/m$. So, the only remaining possible solution for the maximum is the pair $(p(z),q(z))$ associated to $\mu(z)$.

We will describe the second case, $n \le 2m$, in more detail. If there is at least one root, then there is one root $\mu(z)$ for which the associated $p(z)$ converges to $1/m$ when z tends to m. If there is a value z_0 for which there are two roots, then there is one root $\mu_1(z)$ which exists for all $z_0 \le z \le m$, while the other root $\mu_2(z)$ exists for $z_0 \le z \le n(m-1)/(n-1)$, but not for $z > n(m-1)/(n-1)$. This root $\mu_2(z)$ is equal to 0 for z equal to $n(m\text{-}1)/(n\text{-}1)$. For larger values of z $\mu_2(z)$ would become smaller than 0, which is not allowed.

By applying the "implicit function theorem" we can show that for each value of z between 0 and m for which they exist the functions $p(z)$ and $q(z)$ are differentiable with respect to z. We only have to derive the determinant of the Jacobian of the set of equations (15.12) and (15.13). This determinant has to be unequal to zero in order to be able to apply the implicit function theorem. After some, not too difficult, calculations it becomes clear that the determinant is indeed unequal to zero.

15.5.3. Implications of the solutions of the important set of equations

Now we will use the (differentiable) functions $p(z)$ and $q(z)$ to determine the possible optimal values of z. We substitute the pair of functions $p(z)$ and $q(z)$ into the target function N. This gives us another function $M(z)$. The function M is given by

$$M(z) = z(1-p)^{n-1} + (m-z)q(1-q)^{n-1}. \tag{15.18}$$

Because relation (15.13) is valid for all z, we can differentiate this relation with respect to z. We arrive at the following result.

$$(m-z)\frac{dq}{dz} = q - p - z\frac{dp}{dz}. \tag{15.19}$$

Now we are able to determine the derivative of M with respect to z. This will enable us to deduce the possible optimal values for z for the target function N. After that, we have to compare the possible optimal values of N to find the true optimal value. By applying the chain rule, relation (15.12) and relation (15.19) we can show that the derivative of M with respect to z is given by

$$\frac{dM}{dz} = (n-1)(p^2(1-p)^{n-2} - q^2(1-q)^{n-2}), \tag{15.20}$$

where p and q are functions of z.

Using (15.12), or equivalently relation (15.7), again we find

$$(1-q)^{n-2} = (1-p)^{n-2}\frac{1-np}{1-nq}. \tag{15.21}$$

When we substitute this equation into (15.20) we finally arrive at

$$\frac{dM}{dz} = \frac{(n-1)(1-p)^{n-2}}{1-nq}(p-q)(p+q-npq). \tag{15.22}$$

Now we have succeeded in finding an expression for the derivative of M with respect to z. We can therefore determine the optimal z. This turns out to be very easy, because dM/dz has a fixed sign.

Without loss of generality we assume that $p < q$. This is of course equivalent to: $1\text{-}p > 1\text{-}q$. We know now that $p\text{-}q < 0$ and $1\text{-}nq < 0$ (see the conclusions at the end of Section 15.3). So, to establish the sign of dM/dz we only have to determine the sign of $p + q - npq$. This may seem a hard problem, because p and q both depend on the value of z. However, by using the parameterisation of p and q we can demonstrate that the sign of dM/dz does not depend on the actual value of z.

We can rewrite $p + q - npq$ to obtain

$$p+q-npq = -n(1-p)(1-q)+(n-1)(1-p)+(n-1)(1-q)+(2-n)\,. \quad (15.23)$$

From (15.23) and the parameterisation of $1\text{-}p$ and $1\text{-}q$ we can derive

$$p+q-npq = -\frac{1}{(1-\mu^{n-1})}\frac{n-1}{n}(F(\mu)-G(\mu))\,. \quad (15.24)$$

Here $F(\mu)$ and $G(\mu)$ are given by

$$F(\mu) = (1-\mu^{n-1})\left((n-1)\mu(1-\mu^{n-2})-\frac{n(n-2)}{(n-1)}(1-\mu^{n-1})\right) \quad (15.25)$$

and

$$G(\mu) = (1-\mu^{n-2})\left((n-1)\mu(1-\mu^{n-2})-(n-1)(1-\mu^{n-1})\right). \quad (15.26)$$

Because $1-\mu^{n-1} > 1-\mu^{n-2}$ and $n(n\text{-}2)/(n\text{-}1) < n\text{-}1$, we find that $F(\mu)$ is larger than $G(\mu)$ when $0 \le \mu < 1$. In other words, $p + q - npq$ is larger than 0 when $0 \le \mu < 1$. This result does not depend on z. Therefore we can conclude that dM/dz is larger than 0. This implies that in order to optimise the function M we have to make z as large as possible. This, in turn, implies that in order to optimise the target function N we have to make z as large as possible. The actual optimal (real) value of z is determined by the constraints, but we have established that the largest value of z that satisfies all the constraints is the optimal value.

So far we have allowed z to be any real number between 0 and m. Now we remind ourselves that z must be an integer between 0 and m. We know that if $p < q$, then z must be as large as possible.

We have the following cases:

a) If $n \le m$, then the optimal solution is the uniform distribution.

b) If $n > 2m$, the uniform distribution is a local minimum. The optimal real value for z is $n(m\text{-}1)/(n\text{-}1)$, which is larger than $(m\text{-}1)$, but smaller than m. Therefore, the optimal

solution is given by a non-uniform distribution with (m-1) small probabilities and one large probability.

c) If $m < n \leq 2m$, there are two possibilities: either the optimal solution is the uniform distribution or the optimal solution is a non-uniform distribution with (m-1) small probabilities and one large probability. However, it is not clear for which combinations of m and n the uniform distribution is the optimal solution and for which combinations of m and n the non-uniform distribution is the optimal solution. In Section 15.6 this case, $m < n \leq 2m$, is further investigated.

15.6. General remarks about the solution

The solution for $n \leq m$ is given by p_i is equal to $1/m$ for all i. For $n > 2m$ the solution has m-1 probabilities smaller than $2/n$ and one probability larger than $2/n$. If $m < n \leq 2m$ the solution is not clear. In this section we make some remarks about this case.

We start by making the observation that if the non-uniform distribution, i.e. (m-1) probabilities equal to p ($1/n < p < 2/n$) and one probability equal to q ($2/n < q < 1$; $q = 1 - (m\text{-}1)p$), is better than the uniform distribution for a certain number of balls n_0, then this non-uniform distribution is better than the uniform distribution for all $n \geq n_0$. The proof of this assertion is quite simple. To make the dependency on n more explicit we use the following notation:

$$N(p,q;n_0) = (m-1)p(1-p)^{n_0-1} + q(1-q)^{n_0-1}. \tag{15.27}$$

Now, let us suppose that for a certain n_0 the non-uniform distribution is better than the uniform distribution. In other words, we have the following relation

$$N(p,q;n_0) \geq \left(1-\frac{1}{m}\right)^{n_0-1}. \tag{15.28}$$

We have to prove that a similar inequality for n_0+1 instead of n_0 holds. We can do this by making use of the inequality for n_0, and by rewriting the expression for $N(p,q;n_0+1)$. So, we write $N(p,q;n_0+1)$ in the following way

$$\begin{aligned} N(p,q;n_0+1) \;=\; & N(p,q;n_0)\frac{m-1}{m} \;+ \\ & (m-1)p(1/m-p)(1-p)^{n_0-1} + q(1/m-q)(1-q)^{n_0-1} \end{aligned} \tag{15.29}$$

We can combine the last two terms of this expression by making use of another inequality, namely

$$p(1-p)^{n_0-1} \geq q(1-q)^{n_0-1}. \tag{15.30}$$

This inequality holds because $1/(n_0+1) < 1/n_0 < p < q$. Using this inequality to combine the last two terms of the expression for $N(p,q;n_0+1)$ we find yet another inequality, namely

$$N(p,q;n_0+1) \geq (1-1/m)^{n_0} + (1-(m-1)p-q)q(1-q)^{n_0} . \tag{15.31}$$

Finally, by using 1 - (*m*-1)*p* – *q* = 0, we see that we have succeeded in deriving the desired inequality. So, we can draw the conclusion that if, for a certain n_0, there is a non-uniform distribution which is better than the uniform distribution, then for all $n \geq n_0$ there is a non-uniform distribution which is better than the uniform distribution. The problem remains to determine the critical number n_0, given a certain number of urns *m*.

We can evaluate the target function *N* by assuming that the solution is given by uniformly distributed p_i, i.e. $p_i = 1/m$. The value of *N* for uniformly distributed p_i is denoted by N_{uni}. For N_{uni} we have the following expression.

$$N_{\text{uni}} = \left(\frac{m-1}{m}\right)^{n-1} . \tag{15.32}$$

By assuming that the non-uniform solution of the equations for the conjugated values (see (15.12) and (15.13)) exists we can estimate the value of the target function for this solution. This value will be denoted by N_{non}, the maximal expected number of urns with exactly one ball in the case of a non-uniform distribution. In Section 15.5 we have derived that the target function is maximal if *z* is as large as possible. The largest possible real value for *z* is *n*(*m*-1)/(*n*-1). Therefore, an upper bound on N_{non} is given by

$$N_{\text{non,max}} = \frac{m-1}{n-1}\left(\frac{n-1}{n}\right)^{n-1} . \tag{15.33}$$

On the other hand we can find a lower bound $N_{\text{non,low}}$ for N_{non}. This can be obtained by substituting any probability distribution p_i into the target function *N*. A suitable choice is:

$$p_i = 1/n , \qquad \text{for } i=1, 2,\ldots,m-1$$

$$p_m = 1-(m-1)/n$$

Substituting these expressions in *N* yields

$$N_{\text{non,low}} = \frac{m-1}{n}\left(\frac{n-1}{n}\right)^{n-1} + \left(1-\frac{m-1}{n}\right)\left(\frac{m-1}{n}\right)^{n-1} . \tag{15.34}$$

Relations (15.32), (15.33) and (15.34) give criteria to decide which distribution is better:

- if $N_{\text{uni}} \geq N_{\text{non,max}}$, then the uniform distribution is better
- if $N_{\text{uni}} \leq N_{\text{non,low}}$, then the non-uniform distribution is better.

In these two cases we can decide which distribution will be the optimal one without explicitly determining the non-uniform distribution. We are able to decide which distribution is the optimal distribution immediately by looking at the numbers *m* and *n*. By the way, if the latter situation occurs, i.e. if $N_{\text{uni}} \le N_{\text{non,low}}$, then we can be sure that the non-uniform distribution exists. If N_{uni} lies between $N_{\text{non,low}}$ and $N_{\text{non,max}}$, then we have to determine the non-uniform distribution and substitute this solution into the target function. In this case we cannot decide which distribution will be best without explicitly determining the non-uniform distribution.

We investigate the behaviour of N_{uni} and N_{non} if *n* tends to infinity. We do this for two different cases. First, we assume that $m = \alpha n$, where $0 < \alpha < 1$ is a constant. Second, we assume that $m = n\text{-}\beta$, where $\beta > 0$ is a constant.

If $m = \alpha n$, then we have

$$\lim_{n\to\infty} N_{\text{uni}} = e^{-1/\alpha} \tag{15.35}$$

and

$$\lim_{n\to\infty} N_{\text{non,max}} = \lim_{n\to\infty} N_{\text{non,low}} = \alpha/e \tag{15.36}$$

It is an elementary exercise to check that

$$e^{-1/\alpha} < \alpha/e\,. \tag{15.37}$$

for $0 < \alpha < 1$. So, we can conclude that if we keep the ratio $\alpha = n/m$ fixed, then the non-uniform distribution is always better than the uniform distribution for *n* large enough.

If $m = n\text{-}\beta$, we have

$$\lim_{n\to\infty} N_{\text{uni}} = \lim_{n\to\infty} N_{\text{non,max}} = \lim_{n\to\infty} N_{\text{non,low}} = 1/e\,. \tag{15.38}$$

In fact, we have the following

$$(n-1)(N_{\text{uni}} - N_{\text{non,max}}) = \frac{\beta}{e} + g(n) \text{ with } \lim_{n\to\infty} g(n) = 0\,. \tag{15.39}$$

So, we can conclude that if we keep the difference $\beta = n\text{-}m$ fixed, then the uniform distribution is better than any non-uniform distribution for *n* large enough.

15.7. Numerical results

Given the number of urns *m* it is interesting to know the smallest number of balls for which the optimal solution is not uniformly distributed. As a first step to find this number we can apply the criteria given in Section 15.6 to determine which distribution is better. We define E_{uni}, E_{non}, $E_{\text{non,low}}$ and $E_{\text{non,max}}$ by *n* times the corresponding *N*-value. In Table 15.1 the highest number of balls for which E_{uni} is still larger than $E_{\text{non,max}}$, n_u, is listed. For this number of balls and for any smaller number of balls the uniform

distribution is the optimal one. In Table 15.1 the lowest number of balls for which E_{uni} is smaller than $E_{\text{non,low}}$, n_n, is also listed. For this number of balls and for any larger number of balls the non-uniform distribution is the optimal one. For numbers of balls between n_u and n_n the criteria given in Section 15.6 cannot determine which distribution is the optimal one.

Table 15.1. Critical values n_u and n_n as given by the criteria from Section 15.6.

number of urns m	n_u	n_n
5	6	9
10	11	16
15	16	22
20	21	27
30	31	39
40	41	50
50	51	61
60	61	72
70	71	83
80	81	94
90	91	105
100	101	115
150	151	168
200	201	221
300	301	326
400	401	429
500	501	533
1,000	1,001	1,046

For numbers of balls between n_u and n_n it is not clear yet which distribution is the optimal one. Therefore a numerical routine has been implemented. This routine determines the optimal solution to the problem given m urns and n balls, and computes the expected number E of urns with exactly one ball. The results of our numerical experiments are shown in Table 15.2. In this table the "last optimal uniform distribution" and the "first optimal non-uniform distribution" are listed. By this we mean that for any number of balls

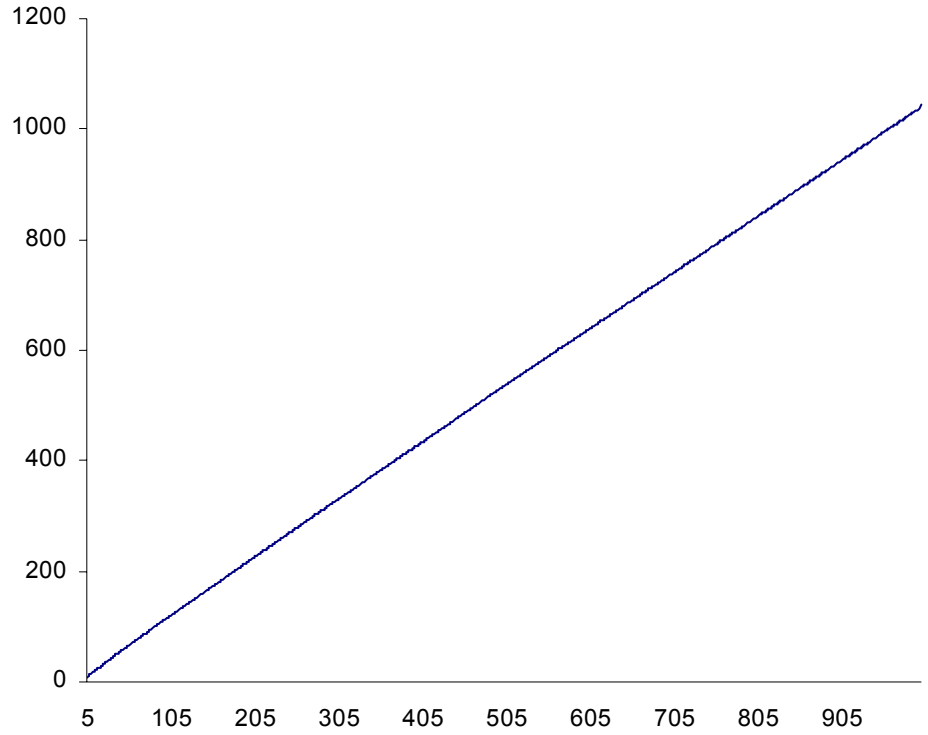
smaller than, or equal to, the number of balls of the last optimal uniform distribution the uniform distribution is the optimal one. For any number of balls larger than, or equal to, the number of balls of the first non-uniform distribution the non-uniform distribution is the optimal one. In other words, the number of balls of the first optimal non-uniform distribution is the critical value $n_0(m)$. In the case of the uniform distribution the probabilities are, of course, given by $1/m$. In the case of the non-uniform distribution there are m-1 small probabilities p and one large probability given by 1-(m-1)p. In Table 15.2 this small probability p is listed. The value of p and the value of the target function E can be found by maximising (15.16) subject to $q = 1-(m-1)p$ and $0 \le p \le 1/m$ and multiplying the result by n. This is a simple optimisation problem involving only one unknown and can be solved by several well-known standard techniques. As we know that $m < n_0(m) \le 2m$, we can find $n_0(m)$ by means of a binary search procedure by solving $\log(m)$ optimisation problems.

Table 15.2. The solution of the problem for given *m* and *n*

number of urns (m)	last optimal uniform distribution		first optimal non-uniform distribution		
	#balls (n)	value target function E	#balls (n)	value target function E	smallest p
5	8	1.678	9	1.568	1.16×0.1
10	15	3.432	16	3.420	6.28×0.01
15	21	5.284	22	5.271	4.56×0.01
20	26	7.212	27	7.123	3.71×0.01
30	38	10.840	39	10.807	2.57×0.01
40	49	14.535	50	14.493	2.00×0.01
50	60	18.218	61	18.175	1.64×0.01
60	71	21.893	72	21.816	1.39×0.01
70	82	25.565	83	25.538	1.20×0.01
80	93	29.235	94	29.218	1.06×0.01
90	104	32.902	105	32.898	9.52×0.001
100	114	36.617	115	36.579	8.70×0.001
150	167	55.016	168	54.978	5.95×0.001
200	220	73.397	221	73.374	4.52×0.001
300	325	110.170	326	110.165	3.07×0.001
400	428	146.979	429	146.955	2.55×0.001
500	532	183.751	533	183.744	1.88×0.001
1,000	1,045	367.694	1,046	367.687	9.56×0.0001

In Figure 15.3 it is shown for which combinations of m and n the uniform distribution is the optimal one, and for which combinations of m and n a non-uniform distribution is the optimal one.

Figure 15.3. The number of balls for the first optimal non-uniform distribution as a function of the number of urns.



The graph of the number of balls for the first optimal non-uniform distribution as a function of the number of urns is almost a straight line. This is, of course, not very surprising if we consider Table 15.2. However, the result is rather surprising if we look at the complexity of the equations that describe the relation between the number of urns, the number of balls and the optimal distribution.

In Table 15.3 the expected number of urns with exactly one ball for the two possible optimal distributions are compared. If there are 5 urns and 8 balls the possible optimal non-uniform distribution does not exist, because the function $h(\mu)$ does not have a root between 0 and 1.

Notice that the value of E_{non} is extremely well approximated by $E_{\text{non,low}}$. So, in practice we may use $E_{\text{non,low}}$ instead of E_{non}. If $E_{\text{non,low}}$ is larger than E_{uni}, then the non-uniform distribution is the optimal one. In this case the optimal value of E is almost equal to $E_{\text{non,low}}$. If E_{uni} is larger than $E_{\text{non,low}}$, then generally the uniform distribution is the optimal one. In this case the optimal value of E is equal to E_{uni}.

Table 15.3. Comparison between the two possible optimal distributions

# urns	# balls	E_{uni}	E_{non}	$E_{non,low}$	$E_{non,max}$
5	8	1.678	-	1.602	1.795
5	9	1.510	1.568	1.567	1.754
10	15	3.432	3.431	3.430	3.670
10	16	3.294	3.420	3.420	3.646
15	21	5.284	5.279	5.279	5.540
15	22	5.166	5.271	5.271	5.522
20	26	7.212	7.130	7.130	7.412
20	27	7.115	7.123	7.123	7.396
30	38	10.840	10.811	10.811	11.103
30	39	10.754	10.807	10.807	11.092
40	49	14.535	14.496	14.496	14.797
40	50	14.461	14.493	14.493	14.788
50	60	18.218	18.178	18.178	18.486
50	61	18.151	18.175	18.175	18.478
60	71	21.893	21.859	21.859	22.171
60	72	21.832	21.857	21.857	22.165
70	82	25.565	25.540	25.540	25.855
70	83	25.507	25.538	25.538	25.849
80	93	29.235	29.220	29.220	29.537
80	94	29.180	29.218	29.218	29.532
90	104	32.902	32.900	32.900	33.219
90	105	32.850	32.898	32.898	33.214
100	114	36.617	36.581	36.581	36.904
100	115	36.850	36.579	36.579	36.900
150	167	55.016	54.979	54.979	55.310
150	168	54.976	54.978	54.978	55.307
200	220	73.397	73.375	73.375	73.710
200	221	73.362	73.374	73.374	73.708
300	325	110.170	110.165	110.165	110.505
300	326	110.140	110.165	110.165	110.504
400	428	146.979	146.956	146.956	147.300
400	429	146.954	146.955	146.955	147.299
500	532	183.751	183.745	183.745	184.091
500	533	183.728	183.744	183.744	184.090
1,000	1,045	367.693	367.688	367.688	368.040
1,000	1,046	367.677	367.687	367.687	368.039

15.8. Summary

The first result we derived was that the probabilities of the optimal solution can have at most two different values. This was a rather easy result, obtained in Section 15.3.

We still did not know how many probabilities have one of the possible values and how many probabilities have the other possible value, though. This question was examined in Section 15.4 and Section 15.5. After much ado, we found that there are two possibilities: either all the probabilities are equal, or there are m-1 small probabilities, which are all equal, and one large probability.

At that moment we were faced with the question of determining for what combinations of m and n all the optimal probabilities have the same value, and for what combinations of m and n there are m-1 small probabilities and one large probability. Part of the answer to this question was already obtained in Section 15.5.

If $n \leq m$, the optimal probabilities are all equal. If $n \geq 2m$, there is one large probability and m-1 small, equal probabilities. In Section 15.6 we examined this question in more detail. We were able to describe the behaviour of the optimal solution if m and n tend to infinity if we assume that either the ratio, or the difference, of m and n is fixed. We also obtained an upper bound, and a lower bound, on the expected number of urns with exactly one ball for the possibly optimal non-uniform distribution. This gives us a criterion to decide whether the uniform distribution or the non-uniform distribution is better, without explicitly determining the possibly optimal non-uniform distribution.

- if $N_{\text{uni}} \geq N_{\text{non,max}}$, the uniform distribution is better;
- if $N_{\text{uni}} \leq N_{\text{non,low}}$, the non-uniform distribution is better.

Unfortunately, there are still combinations of m and n for which we are unable to determine which distribution is the optimal one without explicitly determining the possibly optimal non-uniform distribution. For these cases we have $N_{\text{non,low}} \leq N_{\text{uni}} \leq N_{\text{non,max}}$. In Section 15.7 numerical results are presented for a number of cases. From these numerical results we can conclude that in order to determine the optimal distribution it is in general sufficient to compare the numbers N_{uni} and $N_{\text{non,low}}$. Generally, if $N_{\text{uni}} > N_{\text{non,low}}$ then the uniform distribution is the optimal one, and if $N_{\text{uni}} < N_{\text{non,low}}$ then the non-uniform distribution is the optimal one.

The theory developed in this chapter could be applied in practice as a first, quick check to determine whether identifying information can be released on a certain level of detail. The number of possible identifying features is then compared with the number of respondents. This comparison provides an upper bound on the expected number of unique individuals in the population. If the maximum number of expected unique individuals in the population is low, it is worthwhile considering releasing this amount of identifying information. If the maximum expected number of unique individuals is high, less detailed identifying information should probably be provided. Such a check can be performed very quickly, without actually having to consult the data set itself. This quick check enables one to limit the possibilities for releasing identifying information beforehand. For the limited number

of remaining possibilities, one can then compute the actual expected number of unique individuals in the population by means of a more time-consuming procedure.

16. Synthetic and Combined Estimators in Statistical Disclosure Control

16.1. Introduction

The disclosure avoidance policy of Statistics Netherlands prescribes that the keys that have to be examined for a microdata set for research (for social surveys) consists of three identifying variables, one of which is always a geographical indicator (see Chapter 14 for definitions of the terms "key" and "identifying variable", and a discussion of statistical disclosure control in general). The (estimated) population frequency of the trivariate combinations should be at least d, where d is a certain well-chosen threshold parameter. When a certain combination does not occur frequently enough in the population, disclosure limitation techniques (see e.g. Greenberg, 1990; Marsh et al., 1994) are applied. Examples of such techniques are recoding and suppression, as we mentioned in Chapter 14.

This rule, including an appropriate value for the threshold parameter d, has been found after a time-consuming trial-and-error process. Many different kinds of combinations have been checked, using many values for the threshold parameter. The final result, the above-mentioned rule, seems to be satisfactory: on the one hand the microdata sets resulting from application of this rule are considered sufficiently protected against disclosure, and on other the hand their information content is still rich enough to suit many statistical analyses. For more information on the kinds of microdata sets released by Statistics Netherlands and their rules we refer to Keller and Willenborg (1993).

Application of the above procedure is trivial if the number of population elements (the population frequency) is known for each category for which a minimum population frequency is required. Often this will not be the case, however, and in such situations one can consider estimating these population frequencies from the sample data. If the sampling fraction is sufficiently large, the usual direct estimator for the population frequency (the sample frequency divided by the sampling fraction) can be applied to estimate the population frequencies accurately. If the sampling fraction is not large enough, however, the direct estimator will be too imprecise to be useful. For instance, suppose that the minimum population frequency of a certain category were set at, say 100, then with a sampling fraction a little less than 1:100, the direct estimator would be zero for zero sample frequencies and more than the minimum of 100 for sample frequencies of 1 or larger. This would imply that no disclosure protection measures were necessary for small samples, a highly implausible result. Of course, no one would consider estimating a population frequency on the basis of only a single sample observation.

As an alternative, we describe in this chapter the application of small area estimators (see Chaudhuri, 1994, and Pfeffermann, 2002, for overviews of small area estimation), such as synthetic and combined (or compromise) estimators for the required population frequencies. Small area estimators are based upon the sample data as well as on a model for the population proportions rather than, as is the case with direct estimators, upon the sample only. The model is of vital importance for the quality of a synthetic estimator. If an

appropriate model is used then the resulting synthetic estimator will usually be quite good, but when an inappropriate model is used the estimator can be severely biased. Unfortunately, it is difficult, and in practice often impossible, to establish whether a model is appropriate or not as such a model generally contains assumptions on unobservable random effects. A combined estimator is a combination of a direct estimator and a synthetic one. Generally, a combined estimator is less hampered by the use of an inappropriate model than a synthetic estimator because the bias of the synthetic component of the combination is, to some extent, compensated for by the unbiased direct component.

For more detailed discussions on the disclosure problem in general we refer to Duncan and Lambert (1989), Bethlehem, Keller and Pannekoek (1990), Mokken et al. (1992), and Skinner et al. (1994). For a discussion on the disclosure problem in general and a discussion on the approach based on protecting the individuals with value combinations that occur rarely in the population we refer to De Waal and Willenborg (1996), Willenborg and De Waal (1996 and 2001), and Chapter 14 of this book.

The remainder of this chapter is organised as follows. In Section 16.2 the synthetic and combined estimators we use are described and estimators for the expected mean square errors are derived. In Section 16.3 the proposed estimators are compared by means of an example based on data from the Dutch Labour Force Survey (LFS). A summary of our conclusions is given in Section 16.4.

We have developed and tested the estimators of this chapter in conjunction with Pannekoek. Part of this chapter has appeared in Journal of Official Statistics (Pannekoek and De Waal, 1998). This article has been supplemented by material from Pannekoek and De Waal (1995).

16.2. Synthetic and combined estimators for proportions in small areas

16.2.1. The synthetic estimator

The proportion μ_{ij} of population elements in an area *i* that belong to category *j* is equal to Y_{ij}/N_i, where N_i is the number of inhabitants of area *i* and Y_{ij} is the number of inhabitants of area *i* belonging to category *j*.

Assuming simple random sampling with replacement the sample proportion is an unbiased estimator Z_{ij} for μ_{ij}. So, we define Z_{ij} by

$$Z_{ij} = \frac{y_{ij}}{n_i}, \tag{16.1}$$

where n_i is the sample size in area *i* and y_{ij} is the corresponding number of units in the sample in area *i* that belong to category *j*. An unbiased estimator for the number of units Y_{ij} in the population in area *i* that belong to category *j* is $N_i Z_{ij}$. As is well-known the variance and the mean square error (MSE) of Z_{ij} with respect to the sample distribution is given by

$$\mathrm{Var}_s(Z_{ij}) = \mathrm{MSE}_s(Z_{ij}) = \frac{1}{n_i}\mu_{ij}(1-\mu_{ij}). \tag{16.2}$$

We can use the overall sample proportion, S_{ij}, as a synthetic estimator for μ_{ij}. So, we define S_{ij} by

$$S_{ij} = \frac{y_{+j}}{n}, \tag{16.3}$$

where $y_{+j} = \sum_i y_{ij}$ and $n = \sum_i n_i$. The synthetic estimator S_{ij} will, in general, be a biased estimator for μ_{ij}. Only if the μ_{ij} are equal for all areas *i* will S_{ij} be an unbiased estimator for μ_{ij}. A corresponding synthetic estimator for the number of units Y_{ij} in the population in area *i* that belong to category *j* is NS_{ij}.

The variance of S_{ij} with respect to the sample distribution is given by

$$\mathrm{Var}_s(S_{ij}) = \frac{1}{n}\mu_{+j}(1-\mu_{+j}), \tag{16.4}$$

where $\mu_{+j} = \sum_i N_i\mu_{ij} \big/ N = \sum_i Y_{ij} \big/ N$.

The variance of Z_{ij} is at least equal to the variance of S_{ij} because $n_i \le n$. On the other hand, the synthetic estimator S_{ij} is biased whereas the direct estimator Z_{ij} is not. The bias of S_{ij} is given by

$$b_{ij} = \mathrm{E}_s S_{ij} - \mu_{ij} = \mu_{+j} - \mu_{ij}, \tag{16.5}$$

where E_s denotes the expectation with respect to the sample distribution. The mean square error of S_{ij} is given by

$$\mathrm{MSE}_s(S_{ij}) = \mathrm{Var}_s(S_{ij}) + b_{ij}^2 . \tag{16.6}$$

16.2.2. Estimators for the EMSE of Z_{ij} and S_{ij}

The MSE (variance) of Z_{ij} depends on μ_{ij} (see (16.2)) and the MSE of S_{ij} depends on b_{ij} which in turn depends also on μ_{ij} (see (16.5) and (16.6)). The dependence on μ_{ij} causes difficulties for estimating these MSE's since there is no satisfactory unbiased estimator for μ_{ij} available (this was the reason for using synthetic estimation in the first place). The usual approach to resolve this problem is to assume that b_{ij} is a random variable with expectation $\mathrm{E}_b b_{ij}$ equal to zero and variance $\mathrm{Var}_b(b_{ij})$ equal to, say, σ_j^2. Here E_b and Var_b denote the expectation and the variance with respect to the distribution

of b_{ij} respectively. With these assumptions we can use, instead of the MSE, the expected value with respect to the distribution of b_{ij} of the MSE (EMSE) as a measure of the precision of both Z_{ij} and S_{ij}. These EMSE's do not depend on the area specific μ_{ij} but on both μ_{+j} and σ_j^2 that do not depend on the area *i* but only on the category *j* which makes it possible to estimate the EMSE's.

The expected mean squared error (EMSE) of Z_{ij} is given by

$$\begin{aligned} \mathrm{EMSE}(Z_{ij}) &= \mathrm{E}_b\mathrm{E}_s(Z_{ij}-\mu_{ij})^2 = \mathrm{E}_b\mu_{ij}(1-\mu_{ij})/n_i \\ &= \mu_{+j}(1-\mu_{+j})/n_i - \sigma_j^2/n_i . \end{aligned} \tag{16.7}$$

The expected mean squared error of S_{ij} is given by

$$\mathrm{EMSE}(S_{ij}) = \mathrm{E}_b\mathrm{E}_s(S_{ij}-\mu_{ij})^2 = \mathrm{Var}_s(S_{ij}) + \mathrm{E}_b b_{ij}^2 = \mu_{+j}(1-\mu_{+j})/n + \sigma_j^2 . \tag{16.8}$$

In order to evaluate EMSE(Z_{ij}) and EMSE(S_{ij}) it is necessary to estimate μ_{+j} and σ_j^2. An estimator for μ_{+j} is S_{ij}, i.e. y_{+j}/n. An estimator for σ_j^2 can be obtained by means of the sum of the squared differences between the estimated numbers $n_i S_{ij}$ and $n_i Z_{ij}$. The expectation of this squared difference is equal to

$$\mathrm{E}_b\mathrm{E}_s n_i^2(S_{ij}-Z_{ij})^2 = \mu_{+j}(1-\mu_{+j})n_i(1+n_i/n) + \sigma_j^2 n_i(n_i-1) \tag{16.9}$$

if $\mathrm{E}_b\mathrm{E}_s S_{ij}Z_{ij} = \mu_{+j}^2$. This latter assumption is justified if the number of different areas is sufficiently large. By setting the sum of all squared differences equal to the expectation of this sum, we obtain the following moment estimate for σ_j^2:

$$\hat{\sigma}_j^2 = \frac{\sum_i n_i^2(S_{ij}-Z_{ij})^2 - S_{ij}(1-S_{ij})\sum_i n_i\left(\frac{n_i}{n}+1\right)}{\sum_i n_i(n_i-1)} . \tag{16.10}$$

Spjøtvoll and Thomsen (1987) apply a simpler estimator instead of (16.10). Their estimator is equal to

$$\hat{\sigma}_j^2 = \frac{\sum_i (S_{ij}-Z_{ij})^2 - mS_{ij}(1-S_{ij})}{I-m} , \tag{16.11}$$

where $m = \sum_i 1/n_i$ and $I = \sum_i 1$, i.e. I is equal to the number of different areas in the sample. If the variance of the synthetic estimator S_{ij} is negligible and all n_i are equal the estimator given by (16.10) is the same as the estimator given by (16.11).

16.2.3. The combined estimator

It is well known that it is possible to construct an estimator with a smaller EMSE than both the direct estimator and the synthetic estimator by using a convex combination of these two estimators. This combined estimator, C_{ij}, is given by

$$C_{ij} = W_{ij} Z_{ij} + (1 - W_{ij}) S_{ij} \tag{16.12}$$

where

$$W_{ij} = \frac{\text{EMSE}(S_{ij})}{\text{EMSE}(Z_{ij}) + \text{EMSE}(S_{ij})}. \tag{16.13}$$

The weight W_{ij} is chosen such that the expected mean square error of C_{ij} is minimal. Formula (16.13) shows that the weight approaches 1 if the EMSE of S_{ij} is large compared to the EMSE of Z_{ij}. Since the variance of S_{ij} is small this will happen when the 'working assumption' of homogeneous proportions, i.e. the μ_{ij} are equal for all *i*, does not hold at all and, consequently, the bias of S_{ij} will be large. In this case, the combined estimator C_{ij} will be close to the unbiased estimator Z_{ij}. In the other extreme, if the bias of S_{ij} is small (resulting in a small EMSE for S_{ij}) or if the variance of Z_{ij} is large (resulting in a large EMSE for Z_{ij}) the weight will approach 0 and the combined estimator will be close to the synthetic estimator S_{ij}.

The expected mean square error of C_{ij} is approximately given by

$$\text{EMSE}(C_{ij}) = W_{ij}^2 \text{EMSE}(Z_{ij}) + (1 - W_{ij}^2) \text{EMSE}(S_{ij}). \tag{16.14}$$

The expected mean square error of C_{ij} is at most equal to the minimum of the expected mean square errors of Z_{ij} and S_{ij}.

An estimator of the form (16.12) can also be obtained by an empirical Bayes argument (see Bishop, Fienberg and Holland, 1975, Ch. 12; Albert and Gupta, 1983; Gelman et al., 1995, Ch. 2). In this approach, for each *j*, the parameters μ_{ij} are viewed as realisations of a random variable, M_j say, with expectation μ_{+j} and variance σ_j^2. If we take the distribution of M_j (the prior distribution) to be the Beta(α_j, β_j) distribution, we have $\mu_{+j} = \alpha_j / (\alpha_j + \beta_j)$ and $\sigma_j^2 = \mu_{+j}(1 - \mu_{+j}) / (\alpha_j + \beta_j + 1)$. Furthermore, if it is assumed that the conditional distribution of y_{ij} given $M_j = \mu_{ij}$ is binomial with parameters n_i and μ_{ij}, then the posterior distribution (the conditional distribution of M_j given y_{ij}) is a beta $(y_{ij} + \alpha_j, n_i - y_{ij} + \beta_j)$ distribution. This beta-binomial model is a special case of the more general Dirichlet-multinomial model discussed in Bishop Fienberg and Holland (1975).

The posterior expectation is a Bayesian estimator for μ_{ij}, given by

$$C_{ij}^{B} = \frac{y_{ij} + \alpha_j}{n_i + \alpha_i + \beta_j} = W_{ij}^{B} Z_{ij} + (1 - W_{ij}^{B}) S_{ij} \qquad (16.15)$$

with

$$W_{ij}^{B} = \frac{n_i}{n_i + \alpha_i + \beta_j} = \frac{\sigma_j^2}{\sigma_j^2 + \mu_{+j}(1-\mu_{+j})/n_i - \sigma_j^2/n_i} = \frac{\sigma_j^2}{\sigma_j^2 + \mathrm{EMSE}(Z_{ij})}. \qquad (16.16)$$

The Bayesian estimator can only be calculated if the parameters of the prior distribution are known. Alternatively, an *empirical* Bayesian estimator can be used in which the parameters σ_j^2 and μ_{+j} are replaced by estimates obtained from data values y_{ij} (see e.g. Carlin and Louis, 1996). If μ_{+j} is estimated by S_{ij}, the empirical Bayes estimator will be a linear combination of Z_{ij} and S_{ij}, just like the combined estimator (16.12). It will use different weights, however, because the sampling variance of S_{ij} is not taken into account. If in the combined estimator, the sampling variance of S_{ij} is ignored, then $\mathrm{EMSE}(S_{ij}) = \sigma_j^2$ (see (16.8)) and the combined estimator used in this chapter will be an empirical Bayes estimator. If the variance of S_{ij} is ignored and the n_i are all equal or σ_j^2 is estimated by (16.11) instead of (16.10), then our estimator is equal to the estimator used by Spjøtvoll and Thomsen (1987).

16.2.4. Stratified estimators

The synthetic estimator S_{ij} is based on the 'working assumption' of homogeneity of the population proportions μ_{ij}, i.e. the μ_{ij} are equal for all areas *i*. Although this assumption does not have to be satisfied exactly for the synthetic estimator to perform well, since the bias that is introduced by deviations from this assumption may be compensated for by the small variance of the synthetic estimator, it is worthwhile investigating if this homogeneity assumption can be relaxed. A straightforward way to proceed is to stratify, i.e. to divide, the areas into a small (compared to the number of areas) number of groups of areas and to assume that the μ_{ij} are equal within groups of areas but allow them to vary between groups. This requires that an auxiliary variable is available that indicates to which group each area belongs. For instance, in the application of this chapter, the areas are municipalities and the auxiliary variable is 'degree of urbanisation' in five categories. Using this auxiliary information allows for a synthetic estimator T_{ij} based on five different values for μ_{ij} (one for each category) instead of only one value for the synthetic estimator S_{ij}.

Let $k(r)$ denote the category of the auxiliary variable corresponding to area *r*. The synthetic estimator with auxiliary information, T_{ij}, can then be written as

$$T_{ij} = \frac{\sum_{r \in k(i)} y_{rj}}{\sum_{r \in k(i)} n_r}. \tag{16.17}$$

Like S_{ij} the estimator T_{ij} will, in general, be biased (although to a lesser extent). The bias of T_{ij} is given by

$$b_{ij}^T = \mathrm{E}_s T_{ij} - \mu_{ij}. \tag{16.18}$$

Again we assume that the bias b_{ij}^T is a random variable with expectation $\mathrm{E}_b b_{ij}^T$ equal to zero and variance $\mathrm{Var}_b(b_{ij}^T)$ equal to σ_j^2.

The expected mean square error of T_{ij} is given by

$$\mathrm{EMSE}(T_{ij}) = \frac{\nu_{ij}(1-\nu_{ij})}{\sum_{r \in k(i)} n_r} + \sigma_j^2, \tag{16.19}$$

where ν_{ij} denotes the proportion of population elements that score on category $k(i)$ of the auxiliary variable and on category j of the variable under consideration, i.e. ν_{ij} is equal to

$$\nu_{ij} = \frac{\sum_{r \in k(i)} Y_{rj}}{\sum_{r \in k(i)} N_r}. \tag{16.20}$$

The number ν_{ij} can be estimated by T_{ij}, i.e. by (16.17).

An analysis similar to the analysis of Section 16.2.2. shows that an estimator for σ_j^2 is given by (16.10) where S_{ij} has been replaced by T_{ij}.

Using T_{ij} we can construct another estimator D_{ij} that is similar to the combined estimator C_{ij} of Section 16.2.3. The estimator D_{ij}, which we will call the combined estimator with auxiliary information, is given by (16.12) and the weight W_{ij} by (16.13) where S_{ij} has been replaced by T_{ij} in both formulas. The expected mean square error of D_{ij} is given by (16.14) where T_{ij} has been substituted for S_{ij} again. In general, stratification reduces bias but increases variance. Therefore, it is not clear whether the expected mean square error of T_{ij} is smaller than that of D_{ij} or not.

16.3. Example

16.3.1. Introduction

The data we have used for this example have been obtained from the Dutch LFS 1991. The microdata set from this survey consists of 84,796 records. Of these records 42,248 have been obtained from male respondents and 42,548 from female respondents. These respondents were ranging in age from 15 to 75. From this data set we have used as identifying variables, sex, an area indicator consisting of 646 municipalities and the variable 'occupation' with 90 different categories. For this example it is supposed that there is a disclosure avoidance rule requiring the population frequency for the combination of occupation and sex within each municipality to be above a certain value. So, the problem is to estimate these frequencies. For convenience we will use in this example the records for males only.

For each of the 646 municipalities and for each of the 90 categories of 'occupation', the population proportions and frequencies have been estimated using the estimators described in the previous sections of this chapter. To describe the performance of these estimators in a concise manner, averages of the EMSE's of the estimated proportions were calculated: an average over all 90 categories, an average over categories that did not conform to the homogeneity assumption underlying the synthetic estimator and an average over categories that did conform to this assumption (in the next subsection it is explained how "conforming to the model" is defined). As is apparent from the discussion in Section 16.2.3 the synthetic and combined estimates are similar for categories for which the homogeneity model is a good approximation to reality. In Subsections 16.3.3 and 16.3.5 we illustrate how these estimates can diverge for categories that are considered outliers with respect to the homogeneity model. The use of auxiliary information to improve the synthetic and combined estimators is studied in Subsection 16.3.4. As auxiliary information we have made use of 'degree of urbanisation' a categorical variable with five categories that is available for each municipality. It is shown that auxiliary information can indeed improve the estimators but it is not always best to use all the available auxiliary information.

16.3.2. Definition of outliers

The combined estimator C_{ij} will be approximately equal to the synthetic estimator S_{ij} for areas that are in accordance with the homogeneity assumption, i.e. areas for which the proportions for a certain category are close to the mean proportion $\mu_{+j} = \sum_i \mu_{ij} / n$ for that category (see Section 16.2.3). It will be of interest to see how these estimators compare for areas that deviate from the homogeneity assumption (outliers). For this, we need to determine whether or not a number y_{ij} of units in the sample in an area *i* has to be considered to be an outlier with respect to the homogeneity assumption for category *j*.

A simple test for outliers is based on the distribution under homogeneity of the number of units in the sample per area. These numbers y_{ij} are independently Poisson distributed with parameter $n_i \mu_{+j}$. If the probability that a Poisson($n_i \mu_{+j}$) distributed random variable is at

least y_{ij} is less than a certain threshold value *t*, then area *i* is considered to be an outlier for category *j*. Likewise, if the probability that a Poisson($n_i\mu_{+j}$) distributed variable is at most y_{ij} is less than *t*, then area *i* is also considered to be an outlier for category *j*. The probabilities can be used to order the outliers. In this way the twenty worst outliers can be listed. This list will be used in Subsection 16.3.4 to illustrate the results for the estimators considered.

16.3.3. Comparison of expected mean square errors

The average expected mean square errors over three groups of categories of variable 'occupation' of the direct estimator Z_{ij}, the synthetic estimator S_{ij} and the combined estimator C_{ij} are given in Table 16.1. The first group of categories considered consists of all 90 categories of 'occupation', the second group of the categories with many outliers and the third group of the categories with a few outliers. A category is considered to have many outliers when the number of outlying municipalities is at least 7, otherwise the category is considered to have few outliers.

Table 16.1. Comparison between the average expected mean square errors ($\times 10^{-4}$) of Z_{ij}, S_{ij} and C_{ij} for all categories and for two groups of categories.

	Z_{ij}	S_{ij}	C_{ij}
All categories	1.06	0.17	0.11
Many outliers	2.93	0.69	0.42
Few outliers	0.52	0.02	0.02

For all three groups of categories listed in Table 16.1 we see that the average expected mean square error of the synthetic estimator S_{ij} is clearly less than the average expected mean square error of the direct estimator Z_{ij}.

Table 16.1 moreover clearly demonstrates that the differences between the results for the synthetic estimator S_{ij} and the combined estimator C_{ij} are to be found for categories with many outliers. In Subsection 16.3.5 we will therefore examine the results of Z_{ij}, S_{ij} and C_{ij} for the twenty worst outliers.

16.3.4. Using auxiliary information

As auxiliary information we have made use of 'degree of urbanisation'. This 'degree of urbanisation' consists of five categories, ranging from very urbanised municipalities (category 1), to rural municipalities (category 5). That is, in the remainder of this chapter

D_{ij} and T_{ij} are defined by using all five categories of 'degree of urbanisation'. In Table 16.2 we compare the average expected mean square errors of Z_{ij}, S_{ij}, C_{ij}, D_{ij}, and T_{ij}.

Table 16.2. Comparison between the average expected mean square errors ($\times 10^{-4}$) of Z_{ij}, S_{ij}, T_{ij}, C_{ij} and D_{ij}.

	Z_{ij}	S_{ij}	T_{ij}	C_{ij}	D_{ij}
Average expected mean squared error	1.06	0.17	0.08	0.11	0.06

Table 16.2 shows that using auxiliary information results in a substantial improvement in terms of the average expected mean square for both the synthetic estimator and the combined estimator.

The use of more auxiliary information will generally lead to a decrease of the bias of the resulting estimator. However, the variance of this estimator usually increases as a result of using more auxiliary information. This increase of the variance may be so large that the expected mean square error also increases. In the extreme case an auxiliary variable could be so detailed that each category describes only one municipality. The resulting estimator would then be the direct estimator, without bias but with a large variance. It is clear that this estimator will generally not have the lowest expected mean square error. Therefore, we have considered some other models where two or more categories of the auxiliary variable have been merged.

As the auxiliary variable we have used, i.e. 'degree of urbanisation', is an ordinal variable we have only taken those models into account where adjacent categories of this variables have been merged. For instance, we have considered a model where categories 2, 3, 4 and 5 have been merged, but not a model where categories 2, 3 and 5 have been merged, because categories 3 and 5 are not adjacent. The class of models where the collapsed auxiliary variable 'degree of urbanisation' has two categories left is denoted by class II. Likewise the classes of models where the collapsed auxiliary variable has three respectively four categories left are denoted by class III and class IV, respectively. In each class, the model with the lowest average expected mean square error will be considered the optimal model for this class and will be denoted by II_0, III_0 and IV_0, respectively. Note that it is not necessary to introduce the classes of models where the (collapsed) auxiliary variable 'degree of urbanisation' has one or five categories because both of these classes consist of one model only. The corresponding estimators are given by C_{ij} and D_{ij}, respectively.

Model II_0 turns out to be the model where 'degrees of urbanisation' two to five have been merged into one category, model III_0 the model where 'degrees of urbanisation' two and three and also four and five have been merged, and model IV_0 the model where 'degrees of urbanisation' four and five have been merged. In all three cases it is necessary to merge 'degrees of urbanisation' four and five.

In Table 16.3 the average expected mean square errors of these estimators C_{ij}, II_0, III_0, IV_0 and D_{ij} are listed.

Table 16.3. The average EMSE for a number of models.

Model	Average EMSE ($\times 10^{-5}$)
C_{ij}	1.115
II_0	0.721
III_0	0.622
IV_0	0.596
D_{ij}	0.599

Table 16.3 illustrates that it is not always best to use all the information from the auxiliary variable. Model IV_0 has a smaller average expected mean square error than D_{ij}. The differences in average EMSE between the models that use auxiliary information are small in this case. However, if auxiliary variables with many more categories than just five were available, it would become important to consider the collapsing of categories because this could very well lead to a better estimator than when all available auxiliary information is used.

Note that it would be inappropriate to select the 'best' model on the basis of Table 6.3 as this table is based on the outcome of one 'experiment' only. We only use Table 6.3 to illustrate that the expected mean square error of the estimators may increase if more auxiliary information is used.

16.3.5. Results for categories with many outliers

In this subsection we give the results of Z_{ij}, S_{ij}, C_{ij}, T_{ij} and D_{ij} for the twenty worst outliers (see Subsection 16.3.2 for an explanation of how these outliers have been determined). For non-outliers the results of these estimators are more or less the same. So, for non-outliers it is not very important which estimator is actually used. For outliers, the results of the estimators do differ substantially, and it is important which estimator is used. To decide which of the estimators seems most appropriate for the data set under consideration, we use our knowledge of Dutch society with respect to the twenty worst outliers. The results of Z_{ij}, S_{ij}, C_{ij}, T_{ij} and D_{ij} for the twenty worst outliers are given in Table 16.4.

Table 16.4. The results of Z_{ij}, S_{ij}, C_{ij}, T_{ij} and D_{ij} for the twenty worst outliers.

Category	Municipality	n_i	n_iZ_{ij}	n_iS_{ij}	n_iC_{ij}	n_iT_{ij}	n_iD_{ij}
1. Members of the armed forces	Den Helder	326	28	1.0	20.9	1.5	24.1
2. Farmers	Amsterdam	2,694	0	33.0	1.0	1.7	0.6
3. Fishermen, hunters, etc.	Urk	58	6	< 0.1	1.4	<0.1	4.0
4. Farmers	Rotterdam	2,202	0	27.0	1.0	1.4	0.5
5. Fishermen, hunters, etc.	Wieringen	45	5	< 0.1	1.0	<0.1	2.3
6. Biologists, biochemists, etc.	Wageningen	203	8	0.3	2.5	0.4	5.2
7. Plumbers, welders, etc.	Amsterdam	2,694	4	28.1	6.5	17.5	5.4
8. Bricklayers, carpenters, etc.	Edam-Volendam	187	18	3.4	10.3	2.9	9.4
9. Professional workers n.e.c.[1)]	Amsterdam	2,694	41	15.7	38.6	27.4	37.6
10. Not reporting any occupation	Enschede	749	170	110.5	161.1	115.4	160.8
11. Chemical processors	Terneuzen	243	8	0.6	3.1	0.6	5.8
12. Farmers	Naaldwijk	166	13	2.0	9.3	2.8	9.6
13. Farmers	Nistelrode	70	9	0.9	4.5	2.1	4.1
14. Bricklayers, carpenters, etc.	Amsterdam	2,694	18	48.7	20.2	24.5	21.6
15. Legal professionals	Amsterdam	2,694	19	4.7	17.6	9.8	16.3
16. Farmers	The Hague	1,966	4	24.1	4.8	1.2	2.8
17. Not reporting any occupation	Haarlemmermeer	655	53	96.6	60.3	94.9	60.3
18. Musicians, actors, etc.	Amsterdam	2,694	15	3.1	13.8	7.0	13.1
19. Sculptors, painters, etc.	Amsterdam	2,694	20	5.4	18.4	9.4	18.1
20. Wood preparation workers, etc.	Pekela	110	4	0.1	1.0	0.1	1.9

[1] n.e.c. = not elsewhere classified

In Table 16.4 we see that if there are many respondents the combined estimators C_{ij} and D_{ij} are almost the same as the direct estimator Z_{ij}. This is a positive feature of C_{ij} and D_{ij}, because when the number of respondents is large the variance of Z_{ij} is relatively small, i.e. Z_{ij} is rather reliable. When the number of respondents is small the differences between the combined estimates and the direct estimates can be rather large because the combined estimator is then drawn strongly towards the synthetic estimator. This is desirable, because when the number of respondents is small Z_{ij} will be quite unreliable and one would be willing to accept a 'more synthetic' estimate. The difference between Z_{ij} and the synthetic estimators S_{ij} and T_{ij} is of course rather large in all cases of Table 16.4 because only outliers are considered here.

Stratification leads to a more appropriate model for several variables, such as "Farmers". The number of farmers in a municipality is correlated with the 'degree of urbanisation'. Using 'degree of urbanisation' as auxiliary information hence reduces bias for the number of farmers in a municipality.

16.3.6. Consequences of the estimators for statistical disclosure control

In this subsection we illustrate the results of the various estimators for our statistical disclosure control problem. As we have indicated in Section 16.1, we apply a disclosure control rule of the following kind:

> A combination of values of identifying variables is considered safe, i.e. may be published without further protection, if this combination occurs at least *d* times in the population, where *d* is a suitably chosen threshold.

Because the population frequency of a combination of values of identifying variables is generally unknown, the above rule is in practice replaced by the rule that the *estimated* population frequency should be at least equal to threshold *d* in order for a combination of values to be safe.

Table 16.5 gives the percentage of unsafe combinations of municipality and occupation among the combinations that occur at least once in the survey for various values of the threshold *d*. The total number of combinations equals 646×90=58,140, and the number of combinations that occur at least once in the survey equals 14,432. So, the percentage of unsafe combinations of municipality and occupation among all possible combinations can be obtained from Table 16.5 by multiplying the numbers in Table 16.5 by 14,432/58,140.

Table 16.5. Percentage of unsafe combinations among the combinations that occur at least once in the survey for various values of the threshold *d*.

Estimator	Threshold				
	$d = 10$	$d = 20$	$d = 50$	$d = 100$	$d = 200$
Z_{ij} (Direct estimator)	0	0	0	1.5	43.0
S_{ij} (Synthetic estimator)	3.3	7.7	21.8	39.3	62.0
C_{ij} (Combined estimator)	1.8	4.9	17.2	35.0	58.8
T_{ij} (Stratified synthetic estimator)	3.2	7.9	22.0	38.3	59.5
D_{ij} (Stratified combined estimator)	1.9	5.5	18.5	35.9	57.9

The direct estimator reveals no unsafe combinations for threshold values less than 100. This is not surprising, because in this example the sampling fraction is not the same for each municipality but varies around an average of 1/130, with a maximum of 1/50. This leads to direct estimates that are 50 or larger for combinations with one sample observation. These are implausible estimates but we cannot expect a reasonable estimate for a population frequency on the basis of only one sample observation.

The synthetic and combined estimators produce more plausible estimates, in the sense that it is possible to conclude that combinations are unsafe for threshold values less than 100. The large percentage of unsafe cells for the larger threshold values are also plausible, because the population cell frequencies are 94 on average and will vary considerably around this average since there are small and large municipalities and the distribution over the occupations is also far from uniform. The combined estimators of course give results that are between those of the direct estimator and the synthetic estimators.

A more formal comparison of the estimators could be made by investigating the probability of publishing a cell whose population count is below the threshold. This would involve the evaluation of the probability that the estimated population cell count is equal to or larger than the threshold given that the true population count is below the threshold. Such probability statements, however, require a number of assumptions and some further investigation of these assumptions seems necessary. For instance, if we want to calculate the probability that the estimated population count is below the threshold value given that the true population count Y_{ij} has a value Y_0, say, which is smaller than the threshold, we need the distribution of the estimator under the hypothesis $H_0 : Y_{ij} = Y_0$. For the direct estimator one could simply assume a binomial distribution with the probability Y_0 / N_i (see Pannekoek, 1999), but appropriate assumptions for the synthetic and combined estimators are less obvious. For the combined estimator C_{ij}, for example, we could assume, in line with the assumptions used in the Bayesian approach outlined in Section 16.2.3, that the distribution under H_0 is a beta distribution with expectation Y_0 / N_i. To fully specify this

distribution we also need an estimate for the variance under H_0 and further assumptions are needed to obtain such an estimate.

16.4. Conclusions

An often applied procedure for disclosure protection of microdata sets (for social surveys) is to prescribe a minimum number of population elements for each category of a (combination of) identifying variable(s) and to take measures to ensure that there are no categories with a population frequency less than the prescribed minimum. In many cases the population frequencies will be unknown and the disclosure protection procedure can only be applied if a reasonable estimator for these frequencies is available. The usual unbiased direct estimator cannot be applied because it is based on too few sample observations. For instance, in the example in this chapter we considered the rule that the population frequency of each combination of municipality, sex and occupation should exceed a specified minimum. If this minimum were set at, say, 100 then, with a sample fraction a little less than 1:100, the direct estimator would be zero for zero sample frequencies and more than the minimum of 100 for sample frequencies of 1 or larger. This would imply that disclosure protection measures would not be required for such a data set; not even for combinations of values of identifying variables that occur extremely rarely in the population.

Small area estimators are proposed in this chapter as an alternative: a synthetic estimator and a combined estimator. Both kinds of estimators were applied with and without using an auxiliary variable with respect to the municipalities (degree of urbanisation). For occupations that are (highly) correlated with degree of urbanisation, such as “Farmers”, the use of auxiliary information clearly leads to improved estimates as can be concluded from Table 16.4 and our knowledge of Dutch society.

Based on Tables 16.2 to 16.4 and our knowledge of Dutch society, we would personally select small area estimator D_{ij} to estimate the number of persons with a certain occupation in a municipality for the data set we used in our example. For other variables or data sets, other small area estimators might be more suitable. In any case, the example shows that small area estimators, such as D_{ij}, can be successfully applied to estimate certain population frequencies.

Other small area estimators proposed in literature (see e.g. Chaudhuri, 1994, and Pfeffermann, 2002) also appear to be suitable for estimating the number of persons with a certain occupation in a municipality as well as for similar population frequencies. In comparison to many small area estimators discussed in literature, the synthetic and combined estimators we propose in this chapter are relatively simple to implement and apply. For a statistical office like Statistics Netherlands this simplicity is definitely an advantage, as the estimators need to be applied on a routine basis.

With respect to the disclosure problem, the results for the small area estimators S_{ij}, C_{ij}, D_{ij} and T_{ij} were similar. Of the 58,140 estimated population frequencies (90 occupations times 646 municipalities) slightly less than 40% were below a minimum of 100, so according to these estimators substantial disclosure protection measures will be necessary

if this threshold is used. This is in line with practices at statistical offices, which will almost always use much less detailed area indicators than "municipality" and also use classifications of "occupation" with much less than 90 categories (a one-digit classification is common).

In μ-ARGUS (see Hundepool et al., 2002b) the approach of this chapter has not (yet?) been implemented. Instead a much simpler approach is used. The population frequencies of the combinations to be checked are not estimated by μ-ARGUS from the data set that is to be protected. Instead the user of μ-ARGUS has to provide for each combination to be checked a threshold value. If the frequency in the data set of this combination is above the threshold value, the combination is considered safe. Otherwise, the combination is considered unsafe.

17. Optimal Local Suppression in Microdata

17.1. Introduction

In former days statistical offices used to publish only macrodata, i.e., tables. This was sufficient to satisfy the demands of the users of statistical data. Nowadays, however, the users of statistical data want to have data that are as detailed as possible. Not only do they want more detailed tables, but they also want to have microdata, i.e., data for individual respondents. This is mainly due to the increased power of modern computers, which enables users of statistical data to analyse these microdata by themselves. As a consequence of the demand for microdata statistical offices are put in a difficult position. On the one hand it is their duty to satisfy this demand, on the other hand they should protect the privacy of their respondents.

To protect the privacy of respondents a statistical office should prevent the disclosure of sensitive information on individual respondents by an intruder. To this end, a statistical office usually tries to prevent the re-identification of individual respondents. Re-identification of individual respondents can occur when values of several so-called identifying variables are taken into consideration. Identifying variables, intuitively, concern characteristics of individuals that can be known by other people, and that could be used to track somebody down. An example of a variable that might qualify as non-identifying is "The party for which you voted when you voted for the first time in your life". Examples of identifying variables are "Age", "Sex", "Domicile", and "Occupation". The values of identifying variables can be assumed known to friends and acquaintances of a respondent. Although each identifying variable is generally not sufficient to identify an individual when considered separately, a combination of identifying variables might be sufficient. When a combination of values of identifying variables is unique, i.e., occurs only once in the population, then an intruder might identify the corresponding individual. For example, the combination "Age = 20", "Sex = Female", "Domicile = Amsterdam", and "Occupation = Miner" is very likely a unique combination (if it is not an error!). So, an intruder may identify this individual, i.e., he or she may be able to determine the name of this individual. Subsequently, the intruder may be able to use the released microdata set to disclose sensitive information about this respondent.

In practice, however, it is a bad idea to prevent only the occurrence of respondents in the microdata set who are unique in the population (with respect to a certain combination of values). First, because unicity in the population, in contrast to unicity in the microdata set is hard to establish. Second, because an intruder may look at other combinations of values than the statistical office does. For these reasons it is better to avoid the occurrence of combinations of values in the microdata set that are rare in the population, instead of trying to avoid only the population-uniques in the microdata set. Whether or not rare combinations occur in a microdata set is a well-know criterion for deciding whether this data set may be released. It is, for example, used, along with other criteria, to decide

whether the so-called Samples of Anonymised Records (SARs) from the British population census may be released (see e.g. Willenborg and De Waal, 1996).

To prevent the occurrence of rare combinations of values in the microdata set, statistical disclosure control (SDC) rules at Statistics Netherlands prescribe which combinations of values of identifying variables have to be checked before a microdata set may be disseminated. Moreover, the rules also prescribe how many times these combinations have to occur in order to be considered safe for release, i.e., the rules prescribe certain threshold values. For instance, the SDC rules may describe that the bivariate combinations "Occupation = Statistician" × "Sex = Male" and "Occupation = Statistician" × "Nationality = non-Dutch" each should occur at least 20 times in the microdata set to be considered safe. If the frequency of a particular combination is at least the prescribed threshold value then this combination is considered safe, otherwise the combination is considered unsafe and disclosure limitation measures should be applied. The threshold values may depend on the combination that has to be checked. For example, the threshold value of a bivariate combination may be different from the threshold value of a trivariate combination. For a discussion of this approach see Pannekoek and De Waal (1998), Zaslavsky and Horton (1998), and Chapters 14 and 16 of this book.

To safeguard a microdata set against statistical disclosure two techniques are often applied at Statistics Netherlands, namely *global recoding* and *local suppression.* When global recoding is applied several categories of a variable are combined into a single one. When local suppression is applied the value of a variable in a record is replaced by "missing". These two measures reduce the information content in the microdata set, and hence make re-identification of individual respondents less likely. In Section 17.2 we describe both protection measures in some more detail. The protection measures are, of course, applied only when the privacy of some respondents is endangered. Information that is deemed safe is not protected in order to release as much information as possible. In this chapter we explain how to determine the minimum number of local suppressions such that the resulting microdata set is considered safe. In this way we try to meet the aim of SDC: to preserve as much information as possible while at the same time safeguarding the data.

Local suppression and especially global recoding are not only applied by Statistics Netherlands but also by other statistical offices. In fact, global recoding is by far the most widely used method to protect microdata. Local suppression is less common than global recoding, but is, for example, used to protect the SARs from the British population census (see e.g. Willenborg and De Waal, 1996).

The focus of this chapter is on providing mathematical formulations for various local suppression problems. Such mathematical formulations were unknown at the time this material was written. Jointly with Willenborg we have developed the optimisation models that are presented in this chapter. Solution methods for these mathematical problems are briefly described.

The remainder of this chapter is organised as follows. The problem of minimising the number of local suppressions while protecting a microdata set is described in Section 17.3. This problem is a special case of a general mathematical problem. The general mathematical problem is stated in Section 17.4. A similar problem, namely the problem of minimising the number of different categories that are affected by local suppressions, i.e.,

that are suppressed in at least one record, is discussed in Section 17.5. Special solutions to the problems in Sections 17.4 and 17.5 are examined in Section 17.6. In Section 17.7 solution methods for the various local suppression problems considered are briefly examined. Finally, Section 17.8 concludes this chapter with a short discussion of its main findings and some suggestions for future research.

More information on the view of Statistics Netherlands with respect to the protection of microdata sets can be found in De Waal and Willenborg (1996) and Chapter 14 of this book. For more information on SDC in general we refer to Duncan and Lambert (1986, 1989), Fienberg (1994), Lambert (1993), Marsh, Dale and Skinner (1994), Paass (1988), Skinner et al. (1994) and Willenborg and De Waal (1996 and 2001).

Part of this chapter has appeared in Journal of Official Statistics (De Waal and Willenborg, 1998).

17.2. Global recoding and local suppression

In the present section we discuss the meaning and application of global recoding and local suppression to produce safe microdata. In the case of the procedures applied at Statistics Netherlands this amounts to "removal" of unsafe combinations. In the present section we explain how this removal is to be interpreted in the case of global recoding and in the case of local suppression. Of course, the elimination of unsafe combinations should cause as little information loss (suitably quantified in one way or another) as possible. In Subsection 17.2.1 global recoding is considered and in Subsection 17.2.2 local suppression. Subsection 17.2.3 discusses both techniques, in particular if they are used simultaneously to eliminate unsafe combinations from a microdata file.

17.2.1. Global recoding

In the case of global recoding several categories of a variable V are collapsed into a single one. In the corresponding microdata file the values of V appearing in the respective records are replaced by the new codes. A global recode is applied to the entire data set, not only to the unsafe part of the data set. This is done to obtain a uniform categorisation of each variable. As an example consider the variable "Age" that can take the values 0, 1,2, ..., 99, each of which stands for the age of a person in years. Suppose "Age" is recoded by collapsing the original values into 10-year classes, i.e., by collapsing the original values 0, 1, ..., 9 into a single category 0-9, the original values 10, 11, ..., 19 into a single category 10-19, etc. Then each original value of the variable "Age" in the microdata set is replaced by the corresponding 10-year class. For instance, for each record in which the original value of "Age" equals 26 the new, recoded, value equals 20-29 and similarly for all other ages. The effect of globally recoding the variable "Age" is that less detailed information about the age of a person is released.

In the SDC package μ-ARGUS (see Hundepool et al., 2002b), which has been developed by Statistics Netherlands, global recoding can be applied both interactively and automatically. In the former mode the user should specify which variables to recode and which categories to combine. In the latter mode, μ-ARGUS picks a coding from a set of possible codings for the variable it wants to recode. For instance, for the variable "Region"

the possible codings could be municipalities (in the original file), regions within provinces, provinces, and clusters of provinces. These alternative codings have been prepared prior to the SDC process, for each identifying variable present in the microdata set. Advantages of this way of global recoding, compared to the general case, are that it is easier to automate, and that it allows control of the codings that will be generated. This prevents the possibility that a coding of a variable is generated that is not used in practice.

17.2.2. Local suppression

In the case of local suppression, the value of a variable V in a record R is replaced by a missing value. In the corresponding microdata file the original value of V in record R is replaced by the "missing" value. Whereas a global recode is applied to the entire data set, a local suppression is only applied to a particular value in a particular record. This means that local suppression does not require that definitions of variables need to be changed, because it does not effect the coding of any variable.

To explain the mechanics of this elimination process consider for example, a file of the Dutch Labour Force Survey, containing "Region" (municipalities) and "Occupation". Suppose that, among other things, one has to check the bivariate combination "Region" × "Occupation". Assume that, for instance, the combination "Urk" × "mathematician" occurs once in the file and the threshold is set to three, which means that this combination is considered unsafe. (Urk is a small village in The Netherlands). If we locally suppress "Urk" in the record in which the combination appears, it has to be checked that the remaining 'combination' "mathematician" appears frequently enough in the file, i.e., at least three times. If so, the remaining combination is safe (which is likely). It would also be possible to suppress the value "mathematician" instead of the value "Urk". Then it has to be checked that the combination "Urk" occurs frequently enough in the file. Whatever value is locally suppressed depends on the purposes one has in mind for the analyses of the protected file, and could formally be quantified in terms of information loss. The intuitive goal that one wants to pursue is to produce a safe file with a minimum loss of information.

It should be noted that there is an important difference between local suppression in microdata sets and cell suppression in tables. If a cell value in a table is suppressed it is often necessary to suppress some additional cell values, because usually the marginal totals of the tables are published. These marginal totals allow one in many cases to compute the value of a suppressed internal cell value if no additional cell values have been suppressed (for more information on cell suppression in tables see Willenborg and De Waal, 1996 and 2001). If a value in a record in a microdata set is locally suppressed, then this value cannot be computed, because there are no marginal totals.

17.2.3. Discussion

Both techniques, global recoding and local suppression, can be used to eliminate unsafe combinations, each in its own way. These techniques can be used separately or in combination. Both techniques cause a certain loss of information in the data file to which they are applied. Moreover, local suppression may induce biased estimates when it is being dealt with in a straightforward (but naive) way, such as ignoring the missing values. For that reason it should be applied only on a relatively small scale. Global recoding is the

preferable technique when a large number of unsafe combinations have to be eliminated (and hence the number of required local suppressions would be large). On the other hand, local suppressions may be preferable to global recodings when much information would be lost due to these global recodings. When the number of local suppressions is small the bias introduced in estimates is often negligible, whereas the information loss that would have occurred when the microdata set should have been protected by global recodings only would have been substantial. In practice the right balance has to be found between applying global recodings on the one hand and local suppressions on the other.

At Statistics Netherlands first some variables are globally recoded and subsequently the remaining unsafe records are protected by locally suppressing some values. The package μ-ARGUS is designed to carry out these tasks smoothly. In this chapter we assume that the approach outlined above is used, i.e., that the global recodings have already been carried out. Only the local suppressions remain to be determined. In the remainder of this chapter we explain how the number of local suppressions can be minimised.

This approach can in turn be used as a stepping stone to a more comprehensive – and in practice more useful – model in which the optimum mix of global recodings and local suppressions to eliminate a given set of unsafe combinations has to be calculated (cf. Hurkens and Tiourine, 1998a and 1998b). Such a model can, in turn, be seen as a precursor to a model in which the unsafety definition of microdata on the basis of a frequency criterion for certain combinations of values is replaced by a probabilistic model that yields the disclosure risk for each record in a file (cf. Chapter 14, and De Waal and Willenborg, 1996). The aim is then again to modify an unsafe microdata set by global recoding and local suppression, in such a way that a safe one is obtained, with minimum information loss. A microdata file is then considered safe if the disclosure risk for each record is below a given threshold value.

17.3. Optimal local suppression

The easiest way to determine which variable values should be locally suppressed would be to do this for each combination that has to be checked and for each record separately. Basically there are two ways of doing this. First, a value may be set to "missing" immediately after a certain unsafe combination is identified. The resulting microdata set, with missings due to local suppression, is then used to determine whether or not other combinations, possibly in other records, are safe. Second, the original microdata set may be used to determine whether or not a certain combination is safe. However, both approaches lead to some practical problems.

Suppose that a value is set to "missing" immediately after a certain unsafe combination is identified and that the resulting microdata set, with missings due to local suppression, is used to determine whether or not other combinations, possibly in other records, are safe. The problem with this sequential approach is that some combinations may incorrectly seem to appear not frequently enough. For example, if we suppress the value "Statistician" in the combination "Occupation = Statistician" × "Nationality = non-Dutch", then this may have the consequence that later on the combination "Occupation = Statistician" × "Sex = Male" might not seem to occur frequently enough. This combination would therefore be

considered unsafe. However, this combination could occur frequently enough in the original microdata set. In that case it should in fact be considered safe.

Alternatively, the original microdata set can be used to determine whether or not a combination is safe. This approach may also lead to problems. Suppose, for instance, that the combination "Occupation = Statistician" × "Nationality = non-Dutch" does not occur frequently enough in the (original) file and that we decide to suppress the value of "Nationality", i.e., "non-Dutch", in each record in which this combination occurs. Suppose, furthermore, that the combination "Occupation = Statistician" × "Sex = Female" also does not occur frequently enough and in this case we decide to suppress the value "Female" in each of the corresponding records. Then it is likely that we suppress too much. If there were records of non-Dutch female statisticians in the microdata set, then it would have been better if we had suppressed the single value "Statistician" for these persons instead of the two values "non-Dutch" and "Female". Here we make the assumption that the combinations obtained after suppressing the value "Statistician", i.e., "Nationality = non-Dutch" and "Sex = Female", occur frequently enough in the population. Whether this assumption is correct has to be checked, of course.

We conclude that we cannot decide for each unsafe combination and record *separately* which values should be suppressed if we want to minimise the number of local suppressions. We have to decide which values have to be suppressed for all the unsafe combinations and records *simultaneously*. In Section 17.4 we examine how the resulting problem can be formally stated as a 0-1 integer programming problem.

17.4. Models for optimal local suppression

In this section we formulate the local suppression problem as a general 0-1 integer programming problem. We show that the problem discussed in Section 17.3 is a special case of this general problem. We begin by defining the term 'minimum unsafe combination', or MINUC. These MINUCs will play a crucial role in the remainder of this chapter.

To fix our minds we suppose that the applicable SDC rules require that it is necessary to check whether certain trivariate combinations of categories of identifying variables occur frequently enough. The fact that we are considering combinations of three categories of identifying variables is not really restrictive. The number three could be replaced by any other number without affecting the essence of the method. In the sequel we only consider the identifying variables of the microdata set because our measures to protect a microdata set involve only such variables. In other words, whenever we refer to (a category/value of) a variable we will mean (a category/value of) an identifying variable.

We start by checking all the univariates. In the case that a category of a variable is considered safe, we check the bivariate combinations in which this variable occurs. In the case that a category of a variable does not occur frequently enough, e.g. "Occupation = Mayor", we do not have to check the bivariate combinations involving "Occupation = Mayor", e.g., "Occupation = Mayor" × "Sex = Female", because they will be unsafe as well. Next, we check the trivariate combinations in which only safe bivariate combinations

occur. After checking the required trivariate combinations we are able to list for all the records the minimum unsafe univariate, bivariate and trivariate combinations.

A consequence of this way of constructing the MINUCs is that whenever we suppress a value in a minimum unsafe *n*-variate combination the resulting (*n*-1)-variate combination will be safe. Moreover, when in each MINUC of a record at least one value is suppressed then this record is considered safe. This property of the MINUCs makes it easy to find the minimum number of local suppressions.

Suppose we need to suppress some values in some records. For each value *j* in a MINUC in record *i* we introduce a dummy variable y_{ij}. This dummy variable is equal to 0 if value *j* in record *i* is not suppressed or if value *j* does not occur in record *i;* otherwise it is equal to 1. For each MINUC and for each record we have the constraint stating that at least one value of a MINUC in a record must be suppressed. In other words, the sum of the y_{ij} of the corresponding values is at least 1. As we have remarked before this constraint is necessary and sufficient in order to make this combination safe. As a target function we use a weighted sum of the y_{ij}.

In mathematical terms we consider the following 0-1 integer programming problem. Let the total number of unsafe records be denoted by *I* and the total number of different values involved in the MINUCs by *J.* After renumbering the records and the variables the dummy variables y_{ij} *(i* = 1, ...,*I; j* = 1, ...,*J)* must satisfy

$$y_{ij} = \begin{cases} 1 & \text{if value } j \text{ in record } i \text{ is suppressed;} \\ 0 & \text{if value } j \text{ in record } i \text{ is not suppressed,} \\ & \text{or if value } j \text{ does not occur in record } i. \end{cases} \qquad (17.1)$$

Suppose there are *K* MINUCs in the microdata set. Let c_{jk} (*j*=1,…,*J*; *k*=1,…,*K*) and d_{ik} (*i*=1,…,*I*; *k*=1…,*K*) be defined by

$$c_{jk} = \begin{cases} 1 & \text{if value } j \text{ occurs in MINUC } k \\ 0 & \text{otherwise} \end{cases} \qquad (17.2)$$

and

$$d_{ik} = \begin{cases} 1 & \text{if MINUC } k \text{ occurs in record } i \\ 0 & \text{otherwise.} \end{cases} \qquad (17.3)$$

The constraints of the problem are given by

$$\sum_{j=1}^{J} c_{jk} y_{ij} \geq d_{ik} \ , \quad \text{for all } i=1,\ldots,I;\ k=1,\ldots,K \qquad (17.4)$$

since we have to suppress at least one category in each unsafe combination.

We consider the following target function

$$\sum_{i=1}^{I}\sum_{j=1}^{J} w_{ij} y_{ij} \qquad (17.5)$$

where w_{ij} denotes the non-negative weight of value j in record i which needs to be specified by the user. Our problem is to minimise the target function (17.5) under the constraints given by (17.1) and (17.4).

Note that the problem discussed in Section 17.3 is obtained if we choose all the weights w_{ij} equal to one. Because the weights in the target function (17.5) may be arbitrary non-negative numbers the problem stated above is more general than the problem of Section 17.3. The weights allow one to indicate how important one considers a specific value in a specific record to be as far as local suppression is concerned.

Also note that the problem above can be decomposed into subproblems for each record separately. For each record i the target function (17.5) then has to be replaced by the target function

$$\sum_{j=1}^{J} w_{ij} y_{ij} \qquad \text{for all } i=1,\ldots,I. \qquad (17.6)$$

The constraints to be considered for this problem consist of all those given in (17.4) as far as they pertain to record i.

This subproblem for each record can sometimes be partitioned into a number of smaller subproblems. Consider the MINUCs of a particular record to be the vertices of a graph. Two MINUCs are joined by an edge if and only if they have a value in common. This graph may be disconnected. In that case it consists of several connected subgraphs that are mutually disconnected. Each subgraph corresponds to a subproblem, namely the problem of minimising (17.6) under the constraints that the MINUCs corresponding to the vertices are made safe. Thus sometimes we will be able to reduce the original problem to a number of smaller subproblems.

Even these subproblems may sometimes be reduced to still smaller problems, some of which are trivial. This reduction follows from the observation that only the dummy variables corresponding to values that occur in more than one MINUC have to be considered. Combinations that are still unsafe after some of these values that occur in more than one MINUC have been suppressed can be made safe by suppressing any of the values involved with the lowest weight.

Thus in practice we can expect that the general optimisation problem will be reduced to a number of small subproblems. For a realistic SDC rules, a MINUC contains only about five variables at most. Moreover, for realistic data sets there are at most only a few dozen MINUCS, involving only a few dozen variables. This implies that in many practical cases it is even possible to minimise the target function by complete enumeration.

Example 17.1: Throughout the remainder of this chapter we illustrate the optimal solutions to the problems considered by means of an example. In this example there are eleven

unsafe records, seven variables (V_1 to V_7) and 21 different values (A to U). These eleven records contain the following MINUCs:

record 1: "V_1=A" × "V_2=B" and "V_2=B" × "V_3=C"

record 2: "V_1=A" × "V_4=D" and "V_1=A" × "V_5=E"

record 3: "V_2=F" × "V_3=C"

record 4: "V_1=G" × "V_2=H" and "V_2=H" × "V_3=I"

record 5: "V_5=J" × "V_6=K"

record 6: "V_5=J" × "V_6=L"

record 7: "V_1=M"× "V_2=N" and "V_2=N" × "V_3=O"

record 8: "V_1=M" × "V_3=O"

record 9: "V_5=P" × "V_6=Q" and "V_6=Q" × "V_7=R"

record 10: "V_5=S" × "V_6=T"

record 11: "V_5=S" × "V_7=U" ■

Thus records 1, 2, 4, 7 and 9 each contains two unsafe combinations and the remaining records one each. Note that the records containing two unsafe combinations each have one overlapping variable/value, that is a variable-value combination that appears in both MINUCs. For record 1 this is "V_2=B", for record 2 "V_1=A", for record 4 "V_2=H", for record 7 "V_2=N", and for record 9 "V_6=Q".

Example 17.2: If target function (17.5) has weights w_{ij} all equal to one, then an optimal solution of the problem considered in this section is given by:

suppress in record 1: "V_2=B"

suppress in record 2: "V_1=A"

suppress in record 3: "V_2=F"

suppress in record 4: "V_2=H"

suppress in record 5: "V_5=J"

suppress in record 6: "V_5=J"

suppress in record 7: "V_2=N"

suppress in record 8: "V_3=O"

suppress in record 9: "V_6=Q"

suppress in record 10: "V_6=T"

suppress in record 11: "V_7=U"

So, 11 values are locally suppressed and 10 different categories are involved. ■

A heuristic optimisation routine for solving the above mathematical problem has been implemented in μ-ARGUS (see Hundepool et al., 2002b). For the mathematical problems that are discussed later in this chapter no solution methods have been in implemented in μ-ARGUS yet.

17.5. Minimising the number of different affected categories

Instead of minimising the total number of local suppressions the user of the data might want to minimise the number of different categories that are affected by the local suppressions, i.e., he or she might want to minimise the number of categories that is suppressed in at least one record. A rationale for this could be that he or she considers a category that is suppressed in some records to be unsuited, or hardly suited, for statistical analysis. In other words, affected categories are of no, or only limited, value to him or her.

We can formulate this second problem as follows. First we introduce some new dummy variables. For each category j that occurs in an unsafe combination we introduce a dummy variable z_j, defined as

$$z_j = \begin{cases} 1 & \text{if category } j \text{ is affected, i.e., if category } j \text{ is} \\ & \text{suppressed in some record;} \\ 0 & \text{if category } j \text{ is not affected.} \end{cases} \qquad (17.7)$$

Note that the z_j are independent of the records. The following constraints have to be satisfied.

$$\sum_{j=1}^{J} c_{jk} z_j \geq 1 \qquad \text{for all } k=1,\ldots,K. \qquad (17.8)$$

We consider the following target function.

$$\sum_{j=1}^{J} z_j \,. \qquad (17.9)$$

Target function (17.9) must be minimised under the constraints given by (17.8). The optimisation problem that then arises is a set-covering problem. For set-covering problems special solution methods and computer implementations thereof are available.

This problem can be extended by replacing (17.9) with a weighted sum of the z_j. This would enable the user to indicate how important he or she considers each category to be. Very important categories should be given a large weight; unimportant categories should be given a small weight.

For this problem records cannot be considered independently, as was the case for the problem of Section 17.4. For this reason, in practice, minimisation of the number of different affected categories is a much harder problem than minimisation of the total number of suppressed values. In a realistic data set, there may be many, say several thousands, unsafe records, each involving a number of MINUCs. The total number of variables involved in the MINUCs may be several dozens.

In some cases the problem described in this section can be reduced to smaller sub-problems, because the remarks made in Section 17.4 apply to this case as well. In particular, the problem can sometimes be further reduced to subproblems corresponding to connected subgraphs. In this case the MINUCs correspond to the vertices of a graph. Two vertices are joined by an edge if and only if the corresponding MINUCs have a value in common and both MINUCs occur simultaneously in at least one record.

Example 17.3: We consider our example again. An optimal solution to the problem considered in this section is given by:

suppress in record 1: "V_1=*A*" and "V_3=*C*"

suppress in record 2: "V_1=*A*"

suppress in record 3: "V_3=*C*"

suppress in record 4: "V_2=*H*"

suppress in record 5: "V_5=*J*"

suppress in record 6: "V_5=*J*"

suppress in record 7: "V_1=*M*" and "V_3=*O*"

suppress in record 8: "V_3=*O*"

suppress in record 9: "V_6=*Q*"

suppress in record 10: "V_5=*S*"

suppress in record 11: "V_5=*S*"

So, 13 values are locally suppressed and eight different categories are involved. ■

17.6. Special solutions

After the problems of Sections 17.4 and 17.5 have been solved there are usually a number of possible optimal solutions. Among these possible solutions a solution that is optimal with respect to some additional criterion can be chosen. In this section we consider some of these extended problems.

Suppose that the number of local suppressions has been minimised by solving the 0-1 integer programming problem of Section 17.4. Suppose, furthermore, that among these solutions we want to find the solution that affects a maximum number of different

categories. As a result the local suppressions will probably spread more or less evenly over the categories.

This problem can be formalised as follows. Let the minimum number of local suppressions be denoted by $N_{\min}$. This number is known because we assume that the problem of Section 17.4 has been solved. We want to use both the variables y_{ij} and the variables z_j in one problem. The variable z_j should be equal to one if and only if there is a y_{ij} equal to one for some *i*. This can be achieved by using a large number *M* and introducing the following constraints:

$$Mz_j \geq \sum_{i=1}^{I} y_{ij} \qquad \text{for all } j=1,\ldots,J \tag{17.10}$$

and

$$y_{ij} \geq z_j \text{ for all } j=1,\ldots,J. \tag{17.11}$$

As we want the number of local suppressions to be minimal we have to add the following constraint.

$$\sum_{i=1}^{I}\sum_{j=1}^{J} y_{ij} = N_{\min} \ . \tag{17.12}$$

The target function we consider is given by (17.9). This target function must be maximised under the constraints given by (17.4), (17.10), (17.11) and (17.12).

Example 17.4: We consider our example again. A solution to the problem stated above is given by:

suppress in record 1: "V_2=*B*"

suppress in record 2: "V_1=*A*"

suppress in record 3: "V_2=*F*"

suppress in record 4: "V_2=*H*"

suppress in record 5: "V_6=*K*"

suppress in record 6: "V_6=*L*"

suppress in record 7: "V_2=*N*"

suppress in record 8: "V_1=*M*"

suppress in record 9: "V_6=*Q*"

suppress in record 10: "V_6=*T*"

suppress in record 11: "V_7=*U*"

Thus 11 values are locally suppressed and 11 different categories are affected. ■

Similar to the problem above the user of the data might want to suppress as few different categories as possible while at the same time suppressing as few values as possible.

In this case the target function (17.9) must be *minimised* under the constraints given by (17.4), (17.10), (17.11) and (17.12).

Example 17.5: We consider our example again. A solution to the problem above is given by:

suppress in record 1: "V_2=*B*"

suppress in record 2: "V_1=*A*"

suppress in record 3: "V_2=*F*"

suppress in record 4: "V_2=*H*"

suppress in record 5: "V_5=*J*"

suppress in record 6: "V_5=*J*"

suppress in record 7: "V_2=*N*"

suppress in record 8: "V_1=*M*"

suppress in record 9: "V_6=*Q*"

suppress in record 10: "V_5=*S*"

suppress in record 11: "V_5=*S*"

Thus 11 values are locally suppressed and 9 different categories are affected. ■

The final problem we consider is the following. Suppose that the number of different categories affected has been minimised by solving the 0-1 integer programming problem of Section 17.5. Suppose, furthermore, that among these solutions we want to find a solution that suppresses a minimum number of values.

Let the minimum number of different categories affected be denoted by $M_{\min}$ This number is known because we assume that the problem of Section 17.5 has been solved. We introduce the following constraint:

$$\sum_{j=1}^{J} z_j = M_{\min} \ . \qquad (17.13)$$

In this case we have to minimise (17.5) with all w_{ij} equal to one under the constraints given by (17.4), (17.10), (17.11) and (17.13).

Note that the problem above is easy to solve once the problem of minimising (17.9) under the constraints (17.8) has been solved. On the one hand at least one value per MINUC

should be suppressed. Thus the optimal value of the target function (17.5) (with $w_{ij} = 1$ for all i and j) is at least equal to the number of MINUCs. On the other hand it is sufficient to suppress only one value in a MINUC to make this combination safe. This implies that for a MINUC in which n values have been suppressed we can "re-open" any (n-1) values, i.e., replace the missings by the original values. Thus the optimal value of the target function (17.5) is at most equal to the number of MINUCs. Combining both results we conclude that the optimal value of the target function equals the number of MINUCs, and that a solution to the above problem can be found by re-opening any (n-1) values in each MlNUC in which n values have been suppressed.

Example 17.6: For the last time we consider our example. A solution to the problem above is given by:

suppress in record 1: "V_1=*A*" and "V_3=*C*"

suppress in record 2: "V_1=*A*"

suppress in record 3: "V_3=*C*"

suppress in record 4: "V_2=*H*"

suppress in record 5: "V_5=*J*"

suppress in record 6: "V_5=*J*"

suppress in record 7: "V_2=*N*"

suppress in record 8: "V_1=*M*"

suppress in record 9: "V_6=*Q*"

suppress in record 10: "V_5=*S*"

suppress in record 11: "V_5=*S*"

Thus 12 values are locally suppressed and 9 different categories are affected. ■

17.7. Solution methods

In this section we sketch some solution methods for the problems described in Sections 17.4 to 17.6. First of all, these problems can be solved by using a standard algorithm for solving 0-1 integer problems, such as a branch-and-bound algorithm (cf. Nemhauser and Wolsey, 1988), or by specialised algorithms for the set-covering problem (cf. Wolsey, 1998). For some local suppression problems, such as minimisation of the total number of suppressed values, these algorithms are sufficiently fast to be applied in practice. For other local suppressions problem, such as minimisation of the number of different affected categories, these exact algorithms may be too slow to be successfully applied on practical instances. It is therefore interesting to explore the possibility of applying fast, but suboptimal, heuristics to such computationally complex local suppression problems. In 1995 interesting work in this respect was carried out at Statistics Netherlands by Van Gelderen.

Van Gelderen was a student at the University of Amsterdam who did an internship at Statistics Netherlands, where he was jointly supervised by Dr. Willenborg and the author of this book. In his report Van Gelderen (1995) describes several heuristics for solving the local suppression problems of Sections 17.4 (minimisation of the total number of suppressed values) and 17.5 (minimisation of the total number of affected categories). Hereby, he concentrates on the latter problem as this is the most complex and interesting one from a practical computational point of view. Van Gelderen considers two basic greedy algorithms. In the first greedy algorithm the value that is chosen to be suppressed at any stage is the value that makes the most MINUCs safe at that stage. In the second one a value is chosen that occurs in MlNUCs involving as few values as possible, for example, when only bivariate and trivariate MINUCs occur in the problem then the bivariate MINUCs are protected first. Van Gelderen (1995) also describes an exchange procedure. After a partial solution has been generated, it is likely that values that entered the solution early in the selection process no longer contribute as much to the solution as they did when selected. That is, it may be advantageous to replace a value in the partial solution by a value not occurring in this partial solution. These three ingredients, the two basic greedy algorithms and the exchange procedure, are combined to construct six hybrid algorithms that have already been proposed by Vasko and Wilson (1986) for general set-covering problems.

Apart from greedy algorithms Van Gelderen (1995) examines a heuristic based on the sum of the frequencies of the values in each of the (remaining) MINUCs. The combination with the lowest sum of frequencies, i.e., a combination that is made up of relatively rare values, is chosen. From this combination the value that has the highest frequency, i.e., the value that simultaneously protects as many MINUCs as possible, is selected for suppression. This heuristic is also combined with the first basic greedy algorithm, yielding a kind of modified Vasko and Wilson algorithm.

Finally, Van Gelderen (1995) describes a probabilistic heuristic. At each stage of this heuristic the frequencies of the values in the remaining MINUCs are calculated. Based on these frequencies the probabilities for selecting a specific value for suppression are determined. The probability of suppressing a value is its frequency divided by the sum of the frequencies of all values. The exchange procedure described above is incorporated into this probabilistic heuristic in order not to miss the really important values.

These heuristics have been applied to the problem of minimising the number of affected categories (see Section 17.5). The instances were generated randomly. Each value was considered equally important, i.e., target function (17.9) was used. The largest problem considered consisted of 1,000 unsafe combinations involving 64 different categories. For practical purposes the dimensions of this problem instance can be considered small to (perhaps) moderate. With GAMS\OLS, which uses a branch-and-bound technique for solving mixed integer programming problems, Van Gelderen found a solution involving 45 categories after 13 hours computing time on a Dell 486DX\33MHz. The solution process, which was interrupted after one million iterations, indicated that the optimal value involved at least 41 categories.

To find the optimal solution to the problem, Van Gelderen placed a request on the newsgroup ‘sci.op-research’ on the Internet. Peter Barth from the Max Planck Institut für Informatik in Saarbrücken (Germany) succeeded in solving the problem and proving

optimality of this solution. To find this solution Peter Barth used a Davis-Putnam based algorithm (see for example Chandru and Hooker, 1999, for a description of the Davis-Putnam algorithm). It took him 1,158 seconds of user CPU-time to solve this problem on a SPARC 10/31, a then fast machine, and 3,479 seconds of user CPU-time to prove optimality. Knowing that the optimal solution involved 43 categories, Van Gelderen added a cut saying that at least 43 categories had to be suppressed to the problem formulation, and re-run GAMS\OLS. After for more than 13 hours a solution involving 43 categories was found.

The heuristics each took less than half a second on a Dell 486DX\33MHz PC integrated in a Novell network with diskless workstations. The worst heuristic for this case, the first basic greedy algorithm, obtained a suboptimal solution of 47 different affected categories. The best heuristic for this case, the algorithm based on the sum of frequencies, obtained a suboptimal solution of 44 different affected categories.

For other, smaller, problems the solutions of the heuristics were only compared to each other, not to the optimal solution. For these smaller problems the heuristics were also combined, i.e., each heuristic of such a combination was used to solve the problem to sub-optimality and the best solution found was selected. It turned out that these combinations of different heuristics yield very good results. A combination of three heuristics of the Vasko and Wilson kind, three heuristics of the modified Vasko and Wilson kind, and the probabilistic heuristic accounted for over 95% of the best solutions.

17.8. Discussion

Optimal local suppression procedures can be automated and used in many practical situations, although theoretically the problems are computationally hard (NP-complete). In particular, optimal local suppression procedures can be automated if a local suppression problem splits into several smaller subproblems, each of which can be solved efficiently. This is, for instance, the case if the total number of suppressed values is to be minimised, because then the problem can be solved record-wise. If the number of different affected categories is to be minimised, the problem – in practice – tends to be somewhat more difficult. However, in many cases the problem can be decomposed into a number of smaller, and therefore easier to solve, problems. Moreover, fast heuristics that give good results, i.e., with solutions close to the optimal one, have been constructed. Thus it is possible to solve this problem in many practical instances as well within a relatively short period of time.

For the problem of minimising the total number of suppressed values a heuristic, inspired by the heuristics developed by Van Gelderen (1995), has been implemented in μ-ARGUS, the computer program for SDC developed by Statistics Netherlands (see Hundepool et al., 2002b). For the other problems, which are computationally much harder to solve in practice, no heuristics have yet been implemented in this computer package.

Automating the global recodings is the next logical step to automating the local suppressions. Optimisation models for a special form of global recoding have been developed by Hurkens and Tiourine (1998b), generalising the kind of models presented. The models these authors consider are able to deal with local suppression and global recoding simultaneously. For these models it is assumed that a data protector has, for each

variable that appears in the combinations of variables checked, provided a set of alternative codings. It is also requested from this person to specify how much information is lost when a particular global recoding is applied. The idea is that a global recoding for a variable in this case is nothing but a selection of one of the predefined codings for that variable. The optimisation problem that has to be solved is to select the global recodings in such a way that the resulting microdata set is safe and the associated information loss is minimised.

18. Information Loss for Microdata Sets

18.1. Introduction

Microdata sets have to be protected against disclosure before they can safely be disseminated by a statistical office. More precisely, the risk of disclosure of confidential information should be sufficiently low before a microdata set may be released. Statistical disclosure control (SDC) aims to limit the disclosure risk. This can be done, as is the case at Statistics Netherlands, by checking whether certain combinations of scores occur frequently enough in the population (see for example De Waal and Willenborg, 1996, Willenborg and De Waal, 1996 and 2001, and Chapter 14 of the present book). If such a combination does not occur frequently enough in the population, the disclosure risk of the microdata set under consideration is considered too high and appropriate SDC measures should be taken.

A number of SDC measures can be taken to protect a microdata set: recoding, suppression and perturbation, for example. Recoding is collapsing categories of a variable, suppression is the replacement of a value in a record by a missing value, and perturbation is the replacement of one value by another one. Protecting a microdata set by one of these measures leads to a loss of information. Our aim is to retain as much information in the microdata set while making this data set safe. When microdata are interactively protected using the computer program μ-ARGUS (see Hundepool et al., 2002b) the data protector employs an intuitive meaning of information loss. If the automatic mode of μ-ARGUS is used – in which the best combination of local suppressions and global recodings has to be found by the package itself – a quantification of information loss should be used that allows the package to make the necessary decisions and trade-offs. To achieve this it is necessary to quantify the information loss due to an SDC measure.

In this chapter we consider two methods to quantify the information loss in a microdata file due to local recoding, global recoding, local suppression, or perturbation. One is objective and uses the entropy concept. The other is subjective, and uses weights specified by the data protector. These weights express the predilections of the data protector (who should be enlightened by the users' needs!) for preserving the information of certain variables over others and certain predefined codings for a variable over others. We have developed both methods to quantify the information loss jointly with Willenborg.

The basic idea to evaluate the information loss formally is to use the entropy function for some suitably chosen probability distribution. This probability distribution is a stochastic model for the changes attributable to the applied SDC measures. The basis for the probability function is a 'transition probability matrix' for a variable. Such a matrix gives (an estimate of) the probability that an 'old' value in a record is changed to a particular 'new' one as a result of a modification of the microdata. The matrix for a particular variable in a file may be estimated by assuming a model for the changes due to the SDC measures and by subsequently estimating the corresponding probabilities. To estimate the probabilities it is necessary to make an assumption about the available information on

which these estimates are based. For instance, these probabilities may be estimated by comparing the old and the new files and counting the number of changes that have occurred in the old file with respect to the variable(s) under consideration.

We start our discussion on information loss by considering a technique that can be seen as more elementary than both global recoding and local suppression, namely local recoding. Sections 18.2 to 18.5 discuss the information loss due to local recoding, global recoding, local suppression and data perturbation, respectively. To apply the proposed method it is necessary to estimate so-called transition probabilities. This is the subject of Section 18.6. A subjective method for measuring information loss is described briefly in Section 18.7. Both these information loss measures, the one based on the entropy as well as the subjective one, are used in μ-ARGUS (cf. Hundepool et al., 2002b) in the 'automatic mode' when using a mixture of global recoding and local suppression. The chapter concludes with a short discussion in Section 18.8.

Part of this chapter has appeared in Netherlands Official Statistics (De Waal and Willenborg, 1999b).

18.2. Information loss due to local recoding

In order to be able to define the information loss due to global recoding and local suppression we first consider a related but simpler action, which we shall refer to as *local recoding*. By local recoding we mean that a variable is recoded for *one* record only, whereas by global recoding we mean that a variable is recoded for *all* records in which one of the recoded categories occurs.

Suppose that a certain combination of scores in the file, e.g. 'Age = 17' and 'Marital Status = Widowed' does not occur frequently enough in the population. The records in which this combination occurs have to be protected. This can be achieved, as far as this particular combination is concerned, by recoding the variable 'Marital Status' in these records. For instance, the value 'Marital Status = Widowed' may be replaced by 'Marital Status = Widowed or divorced', assuming that the combination 'Age = 17' and 'Marital Status = Widowed or divorced' occurs frequently enough. In this case there is some uncertainty about the original value of 'Marital Status' for a user of the microdata set.

Now suppose that we can assign a probability p'_W to the event that the original value of 'Marital Status' equals 'Widowed' given that the new, recoded, value equals 'Widowed or divorced', and a probability p'_D that the original value equals 'Divorced'. That is,

$$p'_W = \frac{p_W}{p_W + p_D} \tag{18.1}$$

and

$$p'_D = \frac{p_D}{p_W + p_D}, \tag{18.2}$$

where p_W is the probability that the original value of 'Marital Status' in the record under consideration equals 'Widowed', and p_D is the probability that the original value of 'Marital Status' equals 'Divorced'.

In this case the entropy H_{MS}, i.e. the information loss due to local recoding of 'Marital Status', is given by

$$H_{MS} = -p'_W \log(p'_W) - p'_D \log(p'_D). \tag{18.3}$$

The entropy can be interpreted in several different ways. In one interpretation the quantity H_{MS} given by (18.3) represents a measure for the uncertainty with respect to the original value of 'Marital Status' given that the new value is 'Widowed or divorced'. The entropy is the highest if both values, 'Widowed' and 'Divorced', are equally likely. The entropy is the lowest if one of these values occurs with probability one and the other with probability zero. In this latter case there is in fact no uncertainty about the original value.

In general, when categories C_1, C_2,..., C_n of a variable V in a particular record are combined into a single one, denoted as C_1+C_2+...+C_n, then the information loss H_V due to this local recoding is given by

$$H_V = -\sum_{i=1}^{n} p'_i \log(p'_i), \tag{18.4}$$

where p'_i is the conditional probability that the original value of V in the record under consideration is equal to C_i given that the recoded value equals C_1+C_2+...+C_n. That is,

$$p'_i = \frac{p_i}{\sum_{i=1}^{n} p_i}, \tag{18.5}$$

where p_i is the probability that the original value of V in the record under consideration equals C_i.

So far we have only considered the situation that one variable in one record is recoded. Now we consider the case that several variables are recoded in one record. When variables V_1, V_2,..., V_m are recoded in a particular record k, then the information loss H_k^r in record k due to these local recodings can be measured by

$$H_k^r = -\sum_{i_1,i_2,...,i_m} P(C_{1i_1}, C_{2i_2},..., C_{mi_m}) \log(P(C_{1i_1}, C_{2i_2},..., C_{mi_m})), \tag{18.6}$$

where $P(C_{1i_1}, C_{2i_2},..., C_{mi_m})$ is the joint probability distribution of V_1, V_2,..., V_m. The superscript r indicates that H_k^r is the information loss due to (local)recoding. When we make the simplifying assumption that the variables V_1, V_2,..., V_m are independent then the information loss H_k^r in record k due to the local recodings is measured by the sum of the information losses due to the individual local recodings, i.e.

$$H_k^r = \sum_{j=1}^{m} H_{V_j}^r, \tag{18.7}$$

where $H_{V_j}^r$ denotes the information loss due to the local recoding of variable V_j, which is given by (18.4).

Finally, we consider the case that several variables are recoded in several records. When variables are recoded in records 1, 2,..., K then we measure the total information loss H_{tot}^r due to local recodings in these records by the sum of the information losses in the individual records, i.e.

$$H_{tot}^r = \sum_{k=1}^{K} H_k^r, \tag{18.8}$$

where H_k^r denotes the information loss due to the local recodings applied to record k, which is given by (18.7).

Example 18.1:

We will illustrate our information measures for various statistical disclosure control techniques by means of a simple sample data set. This data set involves only one variable and 20 records. The only variable is 'Marital Status' with possible values 'Never been married', 'Married', 'Divorced', and 'Widowed'. In our sample data set 8 records have the value 'Never been married', 5 records the value 'Married', 5 records the value 'Divorced', and the remaining 2 records the value 'Widowed'. As the probability that the value of 'Marital Status' equals a category C we take the fraction of records with value C in the sample data set. That is, $p_N = 8/20$, $p_M = 5/20$, $p_D = 5/20$, and $p_W = 2/20$, where p_N denotes the probability that the value of 'Marital Status' equals 'Never been married', p_M the probability that the value of 'Marital Status' equals 'Married', p_D the probability that the value of 'Marital Status' equals 'Divorced', and p_W the probability that the value of 'Marital Status' equals 'Widowed'. Now, if we recode the value 'Widowed' into 'Never been married or widowed' in a record, the information loss for that record is given by

$$H = -(1/5)\log(1/5) - (4/5)\log(4/5) = 0.722,$$

if we take 2 as the base of the logarithm. Similarly, if we recode the value 'Married' into 'Married or divorced' in a record, the information loss for that record is given by

$$H = -(1/2)\log(1/2) - (1/2)\log(1/2) = 1.$$

The information loss in the second case is hence larger than for the first case. This is not surprising. If we assume that in each record with value 'Never been married or widowed' the original value is 'Never been married', we are correct in 4 out of 5 cases (assuming that the fraction of records with original value 'Never been married' that is recoded into 'Never been married or widowed' is the same as the fraction of records with original value 'Widowed' that is recoded into 'Never been married or widowed'). In the second case, we

can only correctly guess the original value of 'Married or divorced' in 1 out of 2 cases on the average. ■

18.3. Information loss due to global recoding

In practice, local recodings are hardly ever applied, because they lead to a rather odd kind of microdata set in which the categorisation of each variable can differ per record. In this chapter we use local recoding only as a stepping stone to global recoding. Instead of local recodings one usually applies global recodings. This means that when a variable V is recoded in a particular record, then this recoding is applied to variable V in all records in the microdata set. In this way a uniform categorisation is obtained, i.e. the categorisation of a variable is the same for each record. For instance, when the categories 'Widowed' and 'Divorced' of the variable 'Marital Status' are collapsed into the single category 'Widowed or divorced', then this is done for all records in which the value of 'Marital Status' equals 'Widowed' or 'Divorced'.

Measuring the information loss due to a global recoding is rather easy once the information loss due to local recodings has been defined, because a global recoding can be seen as a number of local recodings that have been applied to all records in the microdata set. Suppose that a variable V is recoded by combining some of the old categories $C_1, C_2,..., C_n$ such that the new categories are given by $D_1, D_2,..., D_m$ ($m \leq n$), where each D_i may be a combination of several C_j. In that case the information loss H_V^r due to the recoding of V can be measured, in each record in which the value of V equals D_j, by

$$H_V^r = -\sum_{i=1}^{n} p_{ij} \log(p_{ij}), \tag{18.9}$$

where p_{ij} denotes the probability that the original category of V in the record under consideration is equal to C_i given that the new category equals D_j.

The total information loss in all records due to global recodings can be measured again by (18.8). An important difference between measuring the information loss due to local recodings and global ones is that in the case of local recodings the recoding of a variable V may differ per record, whereas in the case of global recodings the same recoding of V is applied in each record.

Example 18.2:

We use the same sample data set as in Example 18.1. If we globally recode the values 'Never been married' and 'Widowed' into a single value 'Never been married or widowed', the information loss is given by $10 \times 0.722 = 7.22$. ■

18.4. Information loss due to local suppression

A combination in a microdata set that does not occur frequently enough can also be protected by local suppression, i.e. one or more values in this combination can be deleted.

For example, when the combination 'Age=17' and 'Marital Status = Widowed' does not occur frequently enough, then this combination can be protected by replacing the value of 'Marital Status' by a missing value. Local suppression of the value of a variable V is not applied to all records in a microdata set, but to some of the records only.

The information loss due to local suppressions can be measured in different ways. When local suppressions are not applied in combination with global recoding then the situation can be treated relatively easily. The information loss may be expressed as a weighted sum of the numbers of locally suppressed categories (see also Chapter 17). When local suppressions are applied in combination with global recodings then the situation is more difficult. In this case the entropy could again be used to measure the information loss. Both situations, local suppressions not in combination with recodings and local suppressions in combination with recodings, are examined below. It should be noted that the entropy can also be used to measure the information loss when only local suppression is applied. Using the entropy is better from a theoretical point of view, but is more difficult to apply in practice.

In De Waal and Willenborg (1994a) and Chapter 17 of this book the problem of finding the minimum number of local suppressions to eliminate a set of unsafe combinations, i.e. combinations of scores that are considered rare in the population, in a microdata file is considered. The number of local suppressions is considered as a measure for the information loss due to the suppressions. This problem is also extended to eliminating a set of unsafe combinations while minimising a more general linear target function. For instance, to each variable V_i a weight w_i can be assigned. The information loss in the microdata set due to the local suppressions is then measured by $\sum_i w_i s_i$, where the sum is taken over all variables and s_i equals the number of times that a value of variable V_i is suppressed. Using this linear target function instead of the, non-linear, entropy has the advantage that the problem of determining the optimal local suppressions reduces to solving a 0-1 mixed integer programming problem (see Section 17.5). For such problems several algorithms are available.

The situation is different in De Waal and Willenborg (1995c), where the problem is to find the optimum mix of local suppressions and global recodings to eliminate a given set of unsafe combinations. In that case it is necessary to find a trade-off in information loss due to either type of action. To be able to do this the entropy is introduced.

The way to quantify information loss due to local suppression by means of the entropy is very simple once one realises that local suppression is an extreme form of local recoding, namely all categories of a variable are collapsed into a single one. Information loss for the local suppression of a value of variable V, having n categories, in a particular record is

$$H_V^s = -\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}\log(p_{ij}) \qquad (18.10)$$

where p_i denotes the probability that the original value of variable V in the record under consideration equals C_i.

Example 18.3:

Again we use the sample data set of Example 18.1. If we locally suppress the value of 'Marital Status' for those records for which this value equals 'Widowed', the information loss per record is given by

$$H = -(2/20)\log(2/20) - (5/20)\log(2/20) - (5/20)\log(5/20) - (8/20)\log(8/2) = 1.86.$$

The total information loss due to local suppression of the value 'Widowed' in the entire data set is hence given by $2 \times 1.86 = 3.72$. From Example 18.2 and the present example we can conclude that as long as only relatively few values are suppressed local suppression generally leads to less information loss than global recoding.

Note that our example is far too simplistic for practical purposes. In particular, the probability distribution for the original value of the suppressed data is completely inappropriate here. A clever user with a sufficient understanding of the data set might make the educated guess that all suppressed values equal 'Widowed'. The information loss would then be in fact zero, and the data set would not be protected at all! ■

The total information loss due to local suppressions in record k, H_k^s, is given by formula (18.7) with $H_{V_j}^r$ replaced by $H_{V_j}^s$. The total information loss due to local suppressions in all records, H_{tot}^s, is given by (18.8) with H_k^r replaced by H_k^s. We define the total information loss due to global recodings and local suppressions by

$$H_{tot}^{r+s} = H_{tot}^r + H_{tot}^s. \qquad (18.11)$$

The important aspect of definition (18.11) is that information loss due to (global) recoding can be directly compared to information loss due to local suppression.

18.5. Information loss due to perturbation

The final technique we consider in this chapter to protect a microdata set is perturbation. When perturbation is applied a value of a variable in a record is replaced by another, non-missing, value. As an illustration of perturbation, suppose that the value of the variable 'Marital Status' in a particular record equals 'Widowed'. When perturbation is applied to 'Marital Status' in this record, then the old, original, value 'Widowed' may be replaced by the new, perturbed, value 'Married', for instance.

Suppose we can assign a probability p_{ij}^k to the event that the old, original, value of a variable V in a record k equals C_i given that the new, perturbed, value equals C_j, i.e.

$$p_{ij}^k = \Pr(\text{old code} = C_i \text{ in record } k \mid \text{new code} = C_j \text{ in record } k). \qquad (18.12)$$

Now information loss for variable V due to perturbation, H_V^p, can be calculated, in each record in which the value of V equals C_j, by formula (18.9), with the p_{ij} replaced by the above p_{ij}^k.

Example 18.4:

We once again use the sample data set of Example 18.1. We assume that this data set is protected by perturbing the value 'Widowed' ('W') in one record into 'Never been married' ('N'), and conversely the value 'Never been married' in one record into 'Widowed'. The values 'Married' ('M') and 'Divorced' ('D') are not perturbed. So,

$p_{M,M} = 1$, $p_{*,M} = 0$, where * denotes 'N', 'D', or 'W',

$p_{D,D} = 1$, $p_{*,D} = 0$, where * denotes 'N', 'M', or 'W',

$p_{N,N} = 7/8$, $p_{W,N} = 1/8$, $p_{*,N} = 0$, where * denotes 'M', or 'D',

$p_{W,W} = 1/2$, $p_{N,W} = 1/2$, $p_{*,W} = 0$, where * denotes 'M', or 'D'.

The information loss due to the above perturbation is hence given by

$$H = 2(-(1/2)\log(1/2)-(1/2)\log(1/2))-7(7/8)\log(7/8)-(1/8)\log(1/8) = 5.81.$$

We conclude that in our examples local suppression leads to less information loss than perturbation. An explanation is that in Example 18.3 uncertainty is introduced into only two records, whereas in the present examples uncertainty is introduced into 10 records.

■

The total information loss due to perturbation in record k, H_k^p, is defined by (18.7) with $H_{V_j}^r$ replaced by $H_{V_j}^p$. The total information loss due to perturbation in all records, H_{tot}^p, is defined by (18.8) with H_k^r replaced by H_k^p. Finally, we define the total information loss due to global recodings, local suppressions and perturbations by

$$H_{tot}^{r+s+p} = H_{tot}^r + H_{tot}^s + H_{tot}^p. \qquad (18.13)$$

Definition (18.13) shows the strength of our approach: it offers a general framework to compare the information losses due to various SDC techniques. In particular, (18.13) allows us to compare the information losses due to (global) recoding, local suppression, and perturbation to each other.

18.6. Estimation of transition probabilities

In the previous sections we have implicitly assumed that the 'transition probabilities' can be estimated. The estimation of these probabilities in practice is the subject of the present section. Once the probabilities have been estimated an estimate for the information loss

can be obtained by substituting these estimates into the appropriate formulas. To estimate the probabilities a model, or scenario, has to be assumed. Based on a model the statistical office can 'estimate' the transition probabilities, which in turn can be used to evaluate the information loss due to SDC measures.

In this chapter we use two simple models: a model based on the information available to the statistical office itself, model I, and a model based on the information available to a user of the data, model II. In model I we assume that we know which changes, and how many, have been made to the microdata set, but that we do not know which changes have been made in a particular record. That is, we place ourselves in the situation of an employee of the statistical office who has much information on the applied modifications available, but has not carried out the protection procedure himself. In Examples 18.1 to 18.4 we have in fact used model I to estimate the required probabilities. In model II we place ourselves in the situation of a user of the microdata set. Such a user does not have detailed information about the changes that have been made to the microdata set. However, he does have population knowledge, which can be used to estimate which changes have been made to the microdata set. By selecting one of the models, I or II, the statistical office can estimate the transition probabilities.

To be honest the estimation of the transition probabilities is a hard problem, which is impossible to do in general. The complexity of the estimation problem is partly caused by dependencies between different variables in the same record. These dependencies can be either of a logical or a statistical nature. An example of logical dependency is: when the value of 'Marital Status' equals 'Married' then the value of 'Age' cannot equal 12 years. An example of a statistical dependency is: when the value of 'Marital Status' is 'Widowed' then the probability that 'Age' equals 17 years is low. To estimate the original value of a variable that has been recoded, suppressed or perturbed these dependencies should be taken into consideration. In fact, to estimate the transition probabilities accurately elaborate multivariate techniques should be used. These multivariate techniques would be very difficult and time-consuming to apply.

To complicate estimation of the transition probabilities even more, knowledge about the SDC process itself should be taken into account. For instance, in Example 18.3 a user may know, or suspect, that only categories that do not occur frequently enough in the data set have been suppressed. As 'Never been married', 'Married', and 'Divorced' do occur in the released data set, the user might conclude that both suppressed values equal 'Widowed'. Using his knowledge about the statistical disclosure control process hence leads the user to the very simple – and very effective – probabilities:

$$\Pr(\text{original value equals 'Widowed'} \mid \text{value has been suppressed}) = 1$$

and

$$\Pr(\text{original value equals } C \mid \text{value has been suppressed}) = 0$$

for C different from 'Widowed'. As it is unclear how to model the user's knowledge of the SDC process, we will not examine this aspect any further. Instead we will only examine approaches where no knowledge about the SDC process is assumed.

We propose two simple approaches, which are described below. These approaches allow us to obtain rather crude estimates of the transition probabilities, and hence estimates of the information loss in a microdata set due to the SDC measures examined in this chapter. At first sight the fact that only rather inaccurate estimates of the information loss are obtained in this way may seem to be a problem. However, it should be borne in mind that measuring the 'true' information loss in a microdata set is essentially impossible, because the 'true' information loss is for a substantial part determined by subjective considerations by the users of this data set. The method to evaluate the information loss due to SDC measures that is described in this chapter should merely be seen as a practical and straightforward way to find the appropriate SDC measures. It should not be seen as the ultimate way to evaluate the information loss due to SDC measures.

Now we describe our simple approaches to obtain estimates for the transition probabilities. The basic idea of both approaches is to ignore the relationships that might exist between the variables in a record. To estimate a transition probability for a variable V we only consider its univariate probability distribution.

18.6.1. Recoding

We start by considering recoding. Suppose that some of the old categories C_1, C_2,..., C_n of a variable V are combined in such a way that the new categories are given by D_1, D_2,..., D_m ($m \leq n$), where each D_j is a combination of one or more C_i. To evaluate the information loss due to this recoding we need to estimate the probability p_{ij}^k that the old, original, category equals C_i given that the new, recoded, category equals D_j in a particular record k. The index k indicates that p_{ij}^k, in principle, depends on (the values in) the record under consideration. We shall assume that D_j is obtained by collapsing C_1 to C_s. The probability p_{ij}^k can then be estimated by

$$\hat{p}_{ij}^k = \frac{n_i}{\sum_{t=1}^{s} n_t}, \tag{18.14}$$

where n_i denotes the number of times that C_i (i=1,...,s) occurs in the original, unprotected, microdata set. Note that this estimate does not depend on (the values in) record k.

In model I (18.14) is used as a simple estimate for the probability p_{ij}^k. A user of the data set cannot use (18.14), however, because the numbers n_i (i=1,...,s) are unknown to him. Therefore, in model II another estimate should be used, for example

$$\tilde{p}_{ij}^k = \frac{\hat{N}_i}{\sum_{t=1}^{s} \hat{N}_t}, \tag{18.15}$$

where $\hat{N}_i$ denotes an estimate of the number of times that C_i (i=1,...,s) occurs in the population.

Better estimates for p_{ij}^k may be obtained by subdividing the population into several strata relevant for the variable under consideration. The estimate (18.14) is then constant for all records from the same stratum, instead of constant for all records. The same applies to the estimate (18.15).

Example 18.5:

In Example 18.1 we have already demonstrated model I. Here we illustrate model II. Suppose categories 'Never been married' and 'Widowed' have been recoded into a single category 'Never been married or widowed'. Furthermore, suppose that an estimate for the number of times that 'Never been married' occurs in the target population is 980, and an estimate for the number of times that 'Widowed' occurs in the target population is 195. In that case model II yields the following estimated transition probabilities:

Pr(original = 'Never been married' | recoded = 'Never been married or widowed')
= 980/(980+195)

and

Pr(original = 'Widowed' | recoded = 'Never been married or widowed') = 195/(980+195).

■

18.6.2. Local suppression

The case of local suppression is similar to the case of (local) recoding. Suppose that a value C_1, C_2,..., C_{n-1}, or C_n of variable V is suppressed in a record k. To evaluate the information loss due to this local suppression we need to estimate the probability p_i^k that the original category equals C_i. Again the index k indicates that, in principle, this probability p_i^k depends on the record under consideration. The probability p_i^k is then estimated by

$$\hat{p}_i^k = \frac{n_i}{n_V}, \tag{18.16}$$

where n_i equals the number of times that C_i occurs in the microdata set and n_V equals the number of times that variable V has a non-missing value. Note again that this estimate does not depend on the particular record k.

In model I (18.16) is used as an estimate for the probability p_i^k, but a user of the data set cannot use (18.16) because the numbers n_i (i=1,...,s) are unknown to him. Therefore, in model II another estimate should be used, for example

$$\tilde{p}_i^k = \frac{\hat{N}_i}{\hat{N}_V}, \tag{18.17}$$

where $\hat{N}_i$ denotes an estimate of the number of times that C_i (i=1,...,s) occurs in the population, and $\hat{N}_V$ an estimate of the size of the target population of variable V.

Like in the case of recoding, better estimates may be obtained by subdividing the population into strata relevant for the variable under consideration.

Example 18.6:

We illustrate model II. We use the same data as in Example 18.5. Furthermore, we assume that the estimated frequency of 'Married' in the target population is 540, and the estimated frequency of 'Divorced' in the target population 485. The total target population hence consists of 980 + 540 + 485 + 195, i.e. 2,000 persons. Model II yields the following estimated transition probabilities:

Pr(original value = 'Never been married' | value is suppressed) = 980/2,000,

Pr(original value = 'Married' | value is suppressed) = 540/2,000,

Pr(original value = 'Divorced' | value is suppressed) = 485/2,000,

and

Pr(original value = 'Widowed' | value is suppressed) = 195/2,000. ■

18.6.3. Perturbation

Now we consider the final and most difficult case, namely perturbation. We need to estimate the probability p_{ij}^k that the old, original, value of a variable V in record k equals C_i given that the new, perturbed, value equals C_j. That is, we need to estimate p_{ij}^k defined by (18.12). We assume again that p_{ij}^k does not depend on k, i.e. $p_{ij}^k \equiv p_{ij}$.

We also assume the transition probability t_{ij} , indicating the probability that for an arbitrarily chosen record the category C_i is replaced by category C_j, i.e.

$$t_{ij} = \Pr(\text{new code} = C_j \mid \text{old code} = C_i) , \tag{18.18}$$

is used in model I as well as in model II. In other words, we assume that these transition probabilities t_{ij} are known to the general public, for example because the statistical office itself publishes these transition probabilities.

We also need to estimate the probability q_i that the old, original value equals C_i.

$$q_i = \Pr(\text{old code} = C_i) . \tag{18.19}$$

In model I we can use as an estimate for q_i

$$\hat{q}_i = \frac{n_i}{n_V}, \tag{18.20}$$

where n_i again equals the number of times that the old, original, value equals C_i, and n_V the number of times that variable V has a non-missing value. A user of the microdata set should use another estimate because the numbers n_i and n_V are unknown to him. So, in model II q_i should be estimated differently, for example by

$$\tilde{q}_i = \frac{\hat{N}_i}{\hat{N}_V}, \tag{18.21}$$

where $\hat{N}_i$ denotes an estimate of the number of times that C_i (i=1,...,s) occurs in the population, and $\hat{N}_V$ an estimate of the size of the target population of variable V.

Finally, by applying Bayes' rule to p_{ij}, q_i and t_{ij} we obtain

$$p_{ij} = \frac{t_{ij} q_i}{\sum_i t_{ij} q_i}. \tag{18.22}$$

By substituting the estimate (18.20) for q_i in (18.22) we obtain

$$\hat{p}_{ij} = \frac{t_{ij} \hat{q}_i}{\sum_i t_{ij} \hat{q}_i} \tag{18.23}$$

as an estimate for $p_{ij} = p_{ij}^k$ in model I. This estimate can also be obtained in the following way

$$p_{ij} = \frac{m_{ij}}{m_j}, \tag{18.24}$$

where m_{ij} equals the number of times that a code C_i is changed to C_j, and m_j equals the number of times that the code C_j occurs in the perturbed microdata set. Note that (18.23) and (18.24) are defined only when m_j differs from zero.

In model II, on the other hand, we may substitute the estimate (18.21) for q_i in (18.22) to obtain

$$\tilde{p}_{ij} = \frac{t_{ij} \tilde{q}_i}{\sum_i t_{ij} \tilde{q}_i} \tag{18.25}$$

as an estimate for $p_{ij} = p_{ij}^k$.

Again better estimates may be obtained by subdividing the population into strata.

Example 18.7:

We illustrate model II by means of the same data as in Example 18.6. Moreover, we assume that the transition probability $t_{W,N}$ to perturb the value 'Widowed' into 'Never been married' equals 1/2, and the transition probabilities to perturb the value 'Widowed' into 'Married' or 'Divorced' equal 0. The probability that the value 'Widowed' is not perturbed is therefore given by 1/2. We also assume that the transition probability $t_{N,W}$ to perturb the value 'Never been married' into 'Widowed' equals 1/8, and the transition probabilities to perturb the value 'Never been married' into 'Married' or 'Divorced' equal 0. The probability that the value 'Never been married' is not perturbed is therefore given by 7/8.

Given these data we can use model II to estimate the various probabilities. We find, for instance,

$$\Pr(\text{old} = \text{'Widowed'} \mid \text{new} = \text{'Widowed'}) = \frac{(195/2000)(1/2)}{(195/2000)(1/2)+(980/2000)(1/8)}$$

$$= 0.443.$$

and

Pr(old = 'Never been married' | new = 'Never been married') = 0.898. ■

18.7. Subjective information loss measure: weights

In μ-ARGUS (see Hundepool et al., 2002b) it is possible to use an automatic mode for protecting a microdata set. In this case the program searches for an optimal mix of global recodes and local suppressions to protect the microdata set. If the program is to perform this task, the data protector should make the necessary preparations, including specifying for each identifying variable a set of possible predefined codings (for example for a regional variable codings at the municipality, county, province and area level are possible). Then for each variable he should indicate how important each variable is for him by specifying weights for each variable. Furthermore, for each variable he should indicate how important each of the alternative codings is for him, again by specifying weights. The weight that the system then uses for a particular coding of a particular variable is (proportional to) the product of the variable weight and the coding weight. Furthermore, the user should specify a weight for each identifying variable indicating the cost of suppressing a value of this variable.

The data protector is offered no guidance in specifying these weights in the current version of μ-ARGUS (version 3.1), but he has the full power of this approach based on weights available.

18.8. Discussion

In order to estimate the information loss due to recoding, local suppression and perturbation we propose to use formula (18.13). To apply this formula one has to estimate the transition probabilities. To obtain simple estimates of the transition probabilities corresponding to recoding, local suppression and perturbation we suggest using (18.14), (18.16) and (18.23), respectively, in model I, or (18.15), (18.17) and (18.25), respectively, in model II.

The measure for the information loss obtained in this way is a rather crude one. One reason for this crudeness is that dependencies between variables in the same record are not taken into account. As a consequence the computed information loss will in many cases be higher than the actual information loss. A better information measure may be obtained by constructing more realistic models for the transition probabilities. For example, in Section 18.6 the same transition probability is assigned to each record. More realistic models for the transition probabilities can be constructed by subdividing the records into several strata, where each stratum has its own transition probability. These models, and a number of other models, remain to be examined.

Another reason for the crudeness of our information loss measures is that potential knowledge by a user about the applied SDC techniques to protect the data is not taken into account. The proposed estimation methods for the transition probabilities required to calculate our entropy-based information measures simply neglect the potential presence of such knowledge, which may lead to overestimation of the actual information loss. A possible approach to obtain better estimates for the required transition probabilities is to assume different scenarios for various potential users. In each scenario different knowledge about the applied SDC measures may be assumed. Each scenario corresponds to a certain class of users, each group having their own loss of information.

The attractive aspect about the entropy-based information loss approach is that it is general and versatile, deriving the information loss for various modification techniques such as global recoding and local suppression from a common principle, thereby making a direct comparison between information losses due to different data modification techniques possible. The entropy-based measure of information loss is objective, as it is only based on objective information such as variables, domains, and objective probabilities over these domains. There is no option for a data protector to express his preferences for certain variables or certain recodings. Therefore we have discussed a second type of information loss model in the present chapter, which can actually be called entirely subjective. The model allows the user to express his preferences (over variables, values or codings) in terms of weights.

The disadvantage of this latter approach – which is not shared by the entropy-based approach – is the difficult intercomparability for different data modification techniques. In practice this is likely to be only achievable by a certain degree of experimenting and fine-tuning, through judgement and valuation of the resulting safe microdata. It must be admitted, though, that up to now there has been relatively little experience in setting these subjective weights satisfactorily. A lot of experimenting and testing is needed with real data to develop some intuition in applying this method.

19. Statistical Disclosure Control and Sampling Weights

19.1. Introduction

Before a microdata set can be disseminated by a statistical office it has to be checked whether sensitive information about individual respondents can be disclosed. This should not be possible, of course, because this would endanger the privacy of these respondents. The procedure to check whether the dissemination of a microdata set can lead to disclosure of sensitive information usually amounts to examining how much so-called (indirectly) identifying information is contained in the microdata set. Examples of such identifying information is the age of a respondent and his domicile. If too much identifying information is contained in the microdata set, it is considered unsafe for release. In that case suitable statistical disclosure control (SDC) measures must be taken.

When a statistical office releases a microdata set sampling weights are sometimes included as a service to the public. A description of the used auxiliary variables, their categories and the sampling method is in that case also provided. Unfortunately, the sampling weights, innocent as they may seem, can provide additional identifying information to an intruder. In this chapter we demonstrate that in almost any practical case, such an intruder will indeed be able to determine which combination of categories of the auxiliary variables, i.e. which stratum, corresponds to a specific weight when his knowledge about the population is sufficiently large. After the intruder has matched the weights to the strata, he might be able to (mis-)use the additional identifying information in combination with the other identifying information in the data set to disclose sensitive information about an individual respondent.

In this chapter we describe methods to match weights to strata. These methods are generalisations of a method by Van Kouwen (1993). That latter method can only be used in the case that the numbers of categories of the auxiliary variables obey certain restrictions. In particular, Van Kouwen (1993) does not describe how to apply his method for general numbers of categories of the auxiliary variables, but only provides some examples for specific cases. The present chapter deals with the general case where the auxiliary variables may have arbitrary numbers of categories. Besides giving descriptions of our general methods, we also sketch how weights may be protected by subsampling or by adding noise to them. The methods to match weights to strata described in this chapter as well as the methods to protect weights have been developed in collaboration with Willenborg.

The remainder of this chapter is organised as follows. In Section 19.2 two methods to match the sampling weights and the strata are described. In Section 19.3 several examples are given to illustrate these methods. A number of SDC measures to prevent the possibility to derive additional identifying information from the sampling weights is discussed in Section 19.4. A summary of the results in Section 19.5 concludes this chapter.

Part of this chapter has appeared in Journal of Official Statistics (De Waal and Willenborg, 1997). This article has been supplemented by material from De Waal and Willenborg (1995a).

19.2. Disclosure of identifying information from sampling weights

Sampling weights can be determined by means of several procedures. In this chapter we consider three kinds of such procedures, namely post-stratification, linear weighting and multiplicative weighting (the latter is also called raking, raking ratio estimation or iterative proportional fitting). Details on linear weighting can be found in Bethlehem and Keller (1987) and on multiplicative weighting in Deville and Särndal (1992). As the methods to derive additional identifying information are different for post-stratification on the one hand and linear and multiplicative weighting on the other hand we distinguish between these two situations.

We assume that an intruder knows the population frequencies that have been used to evaluate the sampling weights (almost) perfectly. This is quite a plausible assumption as information provided by the auxiliary variables is generally published by the statistical agency itself. Moreover, we assume that sampling weights corresponding to different strata are different. Again this is a plausible assumption as we are dealing with real life data.

19.2.1. Post-stratification

The case of post-stratification is very easy. By counting the frequency of a certain weight in the sample and by multiplying this frequency by the weight an intruder can determine the number of persons in the population that score on the stratum that corresponds to this weight. Because the intruder has a (nearly) perfect description of the population with respect to the auxiliary variables he can subsequently match the weights to the strata defined by the categories of the auxiliary variables. An example is given in Section 19.3.

19.2.2. Linear/multiplicative weighting

For both linear weighting and multiplicative weighting the product of a weight and its frequency in the sample is generally unequal to the frequency of the corresponding stratum in the population. Such a product is usually only an approximation of the frequency of the corresponding stratum. This complicates the situation for an intruder considerably. However, because the products of the weights and their frequencies do sum up to the marginal totals in the population an intruder is in many cases still able to derive identifying information from the sampling weights. In the sequel we show how an intruder might proceed. We concentrate on two methods the intruder may apply. This does not imply, however, that other methods are impossible; other methods may also be possible. We start by considering the case of multiplicative weighting. After we have examined this case we show how the results obtained can be translated to the case of linear weighting.

To demonstrate how an intruder may proceed we will assume that m auxiliary variables have been used to determine the sampling weights. These auxiliary variables will be denoted by V_i (i=1,…,m). The number of categories of these variables will be denoted by

n_i (i=1,…,m). The categories themselves will be denoted by C_{ij} (i=1,…,m; j=1,…, n_i). We will assume that the variables are ordered in such a way that $n_1 \le n_2 \le ... \le n_m$.

A weight in the multiplicative case is given by

$$W_{i_1 i_2 ... i_m} = F_{i_1}^1 \times F_{i_2}^2 \times ... \times F_{i_m}^m$$

where F_i^k depends only on the category C_{ki}. When two weights have the same category index, we say that these weights have a category in common. Now we discuss the two methods to derive additional identifying information from the sampling weights.

Method 1:

In the first method the disclosure problem is split into two parts. In the first step, the intruder determines which weights have exactly (m-1) categories in common. In the second step, the intruder determines the actual auxiliary variables and categories.

Step 1:

The intruder can begin his mischievous disclosure attempts by listing all the different weights that occur in the microdata set. To determine which weights correspond to the same variable an intruder can then evaluate all the ratios of pairs of weights. These ratios are listed. Whenever we refer to *the ratios list* in the sequel we will mean this list of ratios of weights. Such a ratio has the following form

$$\frac{W_{k_1 k_2 ... k_m}}{W_{l_1 l_2 ... l_m}} = \frac{F_{k_1}^1 \times F_{k_2}^2 \times ... \times F_{k_m}^m}{F_{l_1}^1 \times F_{l_2}^2 \times ... \times F_{l_m}^m}. \qquad (19.1)$$

The ratios F_k^i / F_l^i will also be denoted by $R(i,k,l)$. We will assume that a ratio $R(i,k,l)$ is different from $R(i',k',l')$ whenever $(i',k',l') \neq (i,k,l)$. Moreover, we will assume that $R(i,k,l) \times R(i',k',l') = 1$ if and only if $i' = i$, $k' = l$ and $l' = k$. Note that a ratio $R(i,k,l) = 1$ if and only if k=1.

The value of a ratio of two weights as given by (19.1) occurs

$$\prod_{i \in \{i | k_i = l_i\}} n_{k_i} \qquad (19.2)$$

times in the list, where $\prod_{i \in \varnothing} n_{k_i} = 1$ by definition. For instance, when $k_i \neq l_i$ for all i=1,…,m, then the value of this ratio occurs only once.

The frequency of the ratios of which the weights have all categories except one, say category C_{st}, in common is given by

$$\prod_{i\in\{i|i\neq s\}} n_i \equiv G_s \ . \tag{19.3}$$

If all the G_s are different from the frequencies of ratios of which the weights have less than (m-1) categories in common, then an intruder can determine which weights have (m-1) categories in common. In this case he would be through with the first step. When, moreover, all the values G_s (s=1,…,m) were distinct, then the intruder would even know the associated variables to these common categories. He would only need to find out the actual categories.

When a G_s were equal to the frequency of a ratio of which the weights have less than (m-1) categories in common, then the intruder would not know yet which weights have (m-1) categories in common. This situation occurs when G_s can be written as

$$G_s = n_{i1} \times n_{i2} \times ... \times n_{im} \tag{19.4}$$

for some combination $(i_1, i_2, ..., i_t)$, where t is less than (m-1) and all i_j are distinct. In that case s must be one of the indices i_1, i_2, …, i_t, say $s = i_t$.

Because the above situation can occur, we might hope that an intruder is not always able to determine which weights have (m-1) categories in common. Unfortunately, we have the following theorem.

Theorem 19.1 Assuming that a ratio $R(i,k,l)$ differs from $R(i',k',l')$ whenever $(i,k,l) \neq (i',k',l')$ and that $R(i,k,l) \times R(i',k',l') = 1$ if and only if $i' = i$, $k' = l$ and $l' = k$, an intruder can always determine which weights have (m-1) categories in common.

Proof. We start by observing that weights that have a ratio that occurs $n_2 \times n_3 \times ... \times n_m$ times in the list have (m-1) categories in common, because relation (19.4) cannot be satisfied because $n_1 \leq n_2 \leq ... \leq n_m$. The corresponding ratios can be determined. If possible we determine other weights that have (m-1) categories in common. We can do this for those G_t for which (19.4) cannot be satisfied.

Now suppose there are weights that have a ratio that occurs G_p times in the list (for some p), but for which the intruder cannot determine yet whether or not they have (m-1) categories in common. There may be several numbers p for which this situation occurs. The smallest number will be denoted by s. So, the ratios $R(i,k,l)$ can be determined for i=1,...,s-1. Suppose there are q numbers s, s+1, …, s+q-1 such that $G_s = G_{s+1} = ... = G_{s+q-1}$. Let the ratios of weights that occur G_s times be denoted by R_α^* where α is an index.

These ratios R_α^* are either equal to a $R(s+j,k,l)$ (for j=0, …, q-1) or to a product

$$\prod_{i \neq i_1, i_2, \ldots, i_{t-1}, s, s+1, \ldots, s+q-1} R(i,k,l) \,. \tag{19.5}$$

In the latter case we have the following relation

$$G_s = n_{i_1} \times n_{i_2} \times \ldots \times n_{i_{t-1}} \times n_s \times n_{s+1} \times \ldots \times n_{s+q-1} \,, \tag{19.6}$$

where $t < m$-q-1 and all i_j are distinct. The intruder can multiply the ratios R^*_α by the $R(i,k_i,l_i)$ ' s for i=1,…, s-l. We distinguish between three cases:

1. If R^*_α is equal to a $R(s+j,k,l)$ (for j=0, …, q-1), then the product occurs G_s/n_i times in the ratios list. This follows from (19.2) and $i \neq s+j$.

2. If R^*_α is equal to a product given by (19.5) and $1 \notin \{i_1, i_2, \ldots, i_{t-1}\}$, then the products occur either zero, G_s or $n_i G_s$ times in the ratios list. Namely, when $1 \notin \{i_1, i_2, \ldots, i_{t-1}\}$ then a factor $R(i,k'_i,l'_i)$ occurs in (19.5). So, the product of R^*_α and $R(i,k_i,l_i)$ occurs zero times in $k_i \neq l'_i$ and $l_i \neq k'_i$. The product occurs G_s times if $k_i = l'_i$ and $l_i \neq k'_i$, or $k_i \neq l'_i$ and $l_i = k'_i$. Finally, the product occurs $n_i G_s$ times if $k_i = l'_i$ and $l_i = k'_i$.

3. If R^*_α is equal to a product given by (19.5) and $1 \in \{i_1, i_2, \ldots, i_{t-1}\}$, then the products occur G_s/n_i times in the ratios list. This follows from (19.2) and the fact that no factor $R(i,k_i,l_i)$ occurs in (19.5)

So, if the products occur zero, G_s or $n_i G_s$ times in the ratios list, then the intruder knows that the weights of which the ratio is given by R^*_α do not have (m-1) categories in common. When the products occur G_s/n_i times, then the intruder cannot decide yet whether or not the weights of which the ratio is given by R^*_α have (m-1) categories in common.

So, only if 1,2, ..., s-1 all were elements of $\{i_1, i_2, \ldots, i_{t-1}\}$, the intruder would not be able to determine in this way whether or not the weights with such a ratio R^*_α have (m-1) categories in common.

We claim, however, that when 1,2, ..., s-l are elements of $\{i_1, i_2, \ldots, i_{t-1}\}$, then the weights of which the ratio is R^*_α have (m-1) categories in common. Suppose they did not have (m-1) categories in common. In that case relation (19.6) must be obeyed. Because, 1,2, ..., s-l are elements of $\{i_1, i_2, \ldots, i_{t-1}\}$, $t < m - q$ - 1 and $n_1 \leq n_2 \leq \ldots \leq n_m$, we can conclude that the product $n_{s+q} \times \ldots \times n_m$, i.e., the product of the m - (s + q) + 1 largest n_i is equal to a product of less than m - (s + q) + 1 distinct n_i . This clearly is a contradiction. Hence, 1, 2,, …, s-1 cannot all be elements of $\{i_1, i_2, \ldots, i_{t-1}\}$.

We have demonstrated that the intruder can determine which ratios that occur G_s times have (m-1) categories in common. Now the intruder can apply the same procedure for numbers larger than s. In this way the intruder can determine all weights that have (m-1) categories in common. This concludes the proof and Step 1. ■

Step 2:

After the intruder has determined which weights have (m-1) categories in common he has to determine the actual variables and categories involved.

He can begin by making sets of weights that have *the same* (m-1) categories in common. This is very easy. When weights W_1 and W_2 have (m-1) categories in common and so do W_1 and W_3 and also W_2 and W_3, then W_1, W_2 and W_3 have the same (m-1) categories in common. So, they are placed in the same set. Such a set of weights that have the same (m-1) categories in common will be called an *equivalence class.* Note that the equivalence classes are not disjoint: each weight is an element of m equivalence classes. The total number of equivalence classes is

$$\sum_{i=1}^{m}\prod_{j\neq i} n_j \ . \tag{19.7}$$

The number of elements of the equivalence class of which the weights do not have a category of variable V_i in common is n_i, i.e. the number of categories of V_i.

Now suppose weights W_1 and W_2 are elements of an equivalence class and that W_1' and W_2' are elements of another equivalence class. When the ratios W_1/W_2 and W_1'/W_2' are equal, then W_1 and W_1' have the same category of a variable V_i in common. Namely suppose that W_1 and W_2 have (m-1) categories in common, then their ratio is given by a $R(i,k,l)$. The ratio of W_1' and W_2' is also given by a $R(i',k',l')$. We have assumed that these ratios are equal if and only if $i'=i$, $k'=l$ and $l'=k$. This implies that W_1 and W_1' have category k of variable i in common.

By examining all the ratios of weights in this way the intruder can determine all the weights that have the same category of V_i in common. Note that the intruder may not know the variable V_i. However, if the value n_i occurred only once among $n_1,n_2,...,n_m$,then the intruder would know the variable V_i. The sets of weights that have the same category of a variable in common are not disjoint,

The intruder can evaluate the number of times that a category occurs in the population by multiplying all the weights that have the same category in common by their frequencies and taking the sum of these products, All that remains to be done is to find the categories and variables that correspond to the evaluated population frequencies by comparing these evaluated frequencies to the known frequencies in the population. This concludes the second step. ■

Method 2:

The second method to derive additional identifying information from the sampling weights consists of one step only. Again the ratios of pairs of weights are evaluated. Like in the first method ratios are listed. Now, however, only those ratios that occur G_s times for some *s* are listed, where G_s is given by (19.3). In other words, ratios of weights are listed only when these weights may have (*m*-1) categories in common. Of course, when a G_s satisfies (19.4), then also ratios of weights are listed of which these weights have less than (*m*-1) categories in common.

Now suppose the ratio W_{11}/W_{12} occurs G_s times, i.e. $W_{11}/W_{12} = W_{21}/W_{22} = \ldots = W_{G_s 1}/W_{G_s 2}$ for certain weights W_{i1} and W_{i2} (i=1,…, G_s). We do not know yet whether the weights W_{i1} and W_{i2} have (*m*-1) categories in common or not. Like in Step 2 of method 1 we multiply the weights W_{i1} by their frequencies f_{i1} in the sample and take the sum of these products, i.e. we compute

$$\sum_{i=1}^{G_s} W_{i1} f_{i1} \,. \tag{19.8}$$

Now we examine the two possible cases.

1. *The weights W_{i1} and W_{i2} have* (*m*-1) *categories in common*

When the weights W_{i1} and W_{i2} (i=1,…, G_s) have (*m*-1) categories in common, then the W_{i1} are all the weights that have the same category C_{kj} of a certain variable V_k in common. Like in step 2 of method 1 the number given by (19.8) is then equal to the frequency of category C_{kj} of variable V_k in the population. So, we can determine a category corresponding to a weight W_{i1} whenever W_{i1}/W_{i2} have (*m*-1) categories in common.

2. *The weights W_{i1} and W_{i2} do not have* (*m*-1) *categories in common*

When the weights W_{i1} and W_{i2} (i=1,…, G_s) do not have (*m*-1) categories in common, then the number given by (19.8) is not equal to one of the known frequencies of the categories in the population.

So, by comparing (19.8) to the known frequencies of the categories in the population we can conclude whether or not the weights W_{i1} and W_{i2} have (*m*-1) categories in common. If they do have (*m*-1) categories in common, we can moreover determine the category corresponding to the W_{i1}. In other words, by comparing (19.8) to the known frequencies of the categories in the population an attacker can derive to which stratum a particular weight corresponds. ■

When linear weighting has been used instead of multiplicative weighting almost the same two methods as described above can be applied. In fact one should only replace ratios by differences. For instance, instead of the ratios list an intruder should make a differences list. Examples of both multiplicative weighting and linear weighting are given in Section 19.3.

19.3. Examples

In this section we give some examples to demonstrate how sampling weights can provide additional identifying information. We start by giving an example in the case that the sampling weights have been determined by post-stratification.

Example 19.1:

Suppose that two auxiliary variables A and B have been used to calculate the sampling weights. The number of categories of A is two and of B three. We suppose that post-stratification has been used. Suppose, furthermore, that the frequencies of the strata in the population are given by Table 19.1.

Table 19.1. Frequencies of the strata in the population.

Stratum	Frequency in the population
$A_1 \times B_1$	1368
$A_1 \times B_2$	725
$A_1 \times B_3$	896
$A_2 \times B_1$	2,633
$A_2 \times B_2$	2,787
$A_2 \times B_3$	1,642

The weights are listed in ascending order in Table 19.2.

Table 19.2. Weights of the strata.

Nr.	Frequency in sample	Weight	Weight × Frequency (rounded)
1	20	82.095	1,642
2	10	89.596	896
3	29	96.102	2,787
4	25	105.320	2,633
5	6	120.833	725
6	10	136.799	1,368

The weight of a stratum multiplied by its corresponding frequency in the sample is by definition equal to the size of this stratum in the population. So, if the intruder knew the frequencies of the strata in the population as given in Table 19.1, then he would be able to determine which weight corresponds to which stratum. For instance, it is easy to see that weight 82.095 corresponds to stratum $A_2 \times B_3$, and weight 89.596 to $A_1 \times B_3$.

If the intruder knew the frequencies of the strata in the population only approximately, then he would have to choose the most likely way to match the weights with the strata. If the knowledge of the intruder about the frequencies of the strata in the population is sufficiently precise, then he will be able to determine which stratum belongs to a specific weight. Suppose, for instance, that the intruder would think that stratum i occurs $X_i = X_{\text{pop},i} + \varepsilon_i$ times in the population, where $X_{\text{pop},i}$ is the actual frequency of stratum i in the population and ε_i is an error term with $-50 \le \varepsilon_i \le 50$. It is easy to see that even in this case the intruder would be able to match the weights to their corresponding strata correctly. ■

Now we turn to the case where the sampling weights have been determined by linear weighting or multiplicative weighting. Of both methods described in Section 19.2 we give an example.

Example 19.2:

Suppose that three auxiliary variables *A*, *B* and *C* have been used. The number of categories of variable *A* is two, of variable *B* three and of variable *C* six. We assume that multiplicative weighting has been used. As two times three is six, it is not easy to determine the weights that have both a category of *A* and a category of *B* in common. The example illustrates how this can be done. In this example we apply method 1 of Section 19.2 to derive additional identifying information from the sampling weights. Of course, we could have applied method 2 as well.

In Table 19.3 the knowledge of the intruder about the frequencies of the categories of the auxiliary variables is shown.

Table 19.3. Frequencies of the categories of the auxiliary variables in the population. (Known to the intruder)

Category	Frequency in the population
A_1	7,480,000
A_2	7,649,000
B_1	6,572,000
B_2	7,037,000
B_3	1,520,000
C_1	2,765,000
C_2	3,570,000
C_3	3,605,000
C_4	2,549,000
C_5	1,811,000
C_6	829,000

The weights that are released are listed in ascending order in Table 19.4.

Table 19.4. Weights of the strata.

Nr.	Frequency in the sample	Weight	Weight × Frequency
1	495	94.6384	46,846.01
2	703	95.2153	66,936.36
3	960	96.0524	92,210.30
4	3,368	96.6379	325,476.45
5	7,004	96.8195	678,123.78
6	15,749	96.8338	1,525,035.52
7	6,174	97.4097	601,407.49
8	12,999	97.4241	1,266,415.88
9	168	98.1456	16,488.46
10	105	98.2806	10,319.46
11	620	98.7439	61,221.22
12	233	98.8797	23,038.97
13	2,868	99.0655	284,119.85
14	940	99.2341	93,280.05
15	805	99.6120	80,187.66
16	4,626	99.6694	461,070.64
17	623	99.8391	62,199.76
18	3,599	100.2193	360,689.26
19	1,848	100.4076	185,553.24
20	13,989	100.7081	1,408,805.61
21	1,338	100.7168	134,759.08
22	4,121	100.8228	415,490.76
23	1,236	101.0197	124,860.35
24	13,385	101.3220	1,356,194.97
25	1,959	101.3308	198,507.04
26	2,495	101.4374	253,086.31
27	10,583	101.5212	1,074,398.86
28	9,652	102.1400	985,855.28
29	0	102.2128	0.00
30	1,228	102.3292	125,660.26
31	0	102.8358	0.00
32	1,643	102.9530	169,151.78
33	0	103.0291	0.00
34	12,688	103.1464	1,308,721.52
35	0	103.6571	0.00
36	12,844	103.7752	1,332,888.67

We apply the method of Section 19.2 to extract identifying information from the sampling weights. We begin by making the ratios list. Because this list is rather large – it contains

36×36 ratios – only a part is presented in Appendix A. Only ratios less than one that occur more than once are listed. Of course, the reciprocals of these ratios occur more than once too in the actual ratios list.

Examining the ratios list we see, for instance, that the ratio of W_1 and W_2, i.e. 0.99394, occurs $18 = n_2 \times n_3$ times in the ratios list. So, W_1 and W_2 have a category of variable *B* and a category of variable *C* in common. The ratio of W_1 and W_3, i.e. 0.98528, occurs $12 = n_1 \times n_3$ times in the list. Because $12 \neq n_2$, we know that W_1 and W_3 have a category of variable *A* and a category of variable *C* in common.

The situation is somewhat more difficult for W_1 and W_6. The ratio of these weights, i.e. 0.97733, occurs 6 times in the list. Because $6 = n_1 \times n_2 = n_3$, we cannot determine yet whether or not W_1 and W_6 have two categories in common. The same situation occurs for W_1 and W_7. The ratio of these weights, i.e. 0.97155, also occurs 6 times.

We multiply the ratio of W_1 and W_6 and the ratio of W_1 and W_7 by the ratios of weights that have categories of variables *A* and *C* in common and by the ratios of weights that have categories of variables *B* and *C* in common. So, we multiply the ratio of W_1 and W_6 by the ratio of W_2 and W_1, for instance. The resulting ratio, W_2 divided by W_6, equals 0.98329. This number occurs $3 = G_3/n_1$ times in the list. Multiplying the ratio of W_1 and W_6 by the ratio of each pair of weights that have categories of variables *B* and *C* in common results in a ratio that occurs $G_3/n_1 = 3$ times in the list.

Similarly, when we multiply the ratio of W_1 and W_6 by the ratio of W_3 and W_1, then the resulting ratio, W_3 divided by W_6, occurs $2 = G_3/n_2$ times in the list. Multiplying the ratio of W_1 and W_6 by the ratio of each pair of weights that have categories of variables *A* and *C* in common results in a ratio that occurs $G_3/n_2 = 6$ times in the list.

According to the proof of Theorem 19.1, we can conclude that W_1 and W_6 have two categories in common. These two categories are categories from *A* and *B*.

When we multiply the ratio of W_1 and W_7 by the ratio of W_2 and W_1, then the resulting ratio, W_1 divided by W_7, equals 0.97747. This number occurs 12 times in the list. Therefore, according to the proof of Theorem 19.1, W_1 and W_7 do not have two categories in common.

In the above way one can determine all the weights that have two categories in common. The next step is to determine which weights form equivalence classes. This is easy. For example, W_1 and W_3 have two categories in common and so have W_1 and W_5, and W_3 and W_5. Therefore, W_1, W_3 and W_5 have the same two categories in common. In particular, W_1, W_3 and W_5 form an equivalence class. Likewise W_1 and W_2 form an equivalence class, and so do W_1, W_6, W_9, W_{14}, W_{20} and W_{22}.

Because the ratio of W_1 and W_6, i.e. 0.97733, is equal to the ratio of W_7 and W_{16}, the numerators of these ratios, W_1 and W_7, have a, still unknown, category of variable *C* in common. In this way, we can determine all the weights that have a, as yet unknown, category C_i of variable *C* in common. This enables us to evaluate the frequencies of C_i in the population by multiplying these weights by their frequencies and subsequently taking the sum of these products. This can be done for all categories of *C*.

Similarly, we can determine all the weights that have a, still unknown, category A_j of variable *A* in common and all the weights that have a, also unknown, category B_k of variable *B* in common. Subsequently, frequencies of A_j and of B_k in the population can be evaluated by multiplying the corresponding weights by their frequencies and taking the sum of these products. The results are listed in Table 19.5.

Table 19.5. Frequencies of the categories of the auxiliary variables in the population. (Evaluation based on microdata set)

Category	Corresponding weight indices	Frequency in the population
A_x	1 3 5 6 9 10 13 14 15 19 20 21 22 27 29 30 33 34	7,480,000
A_y	2 4 7 8 11 12 16 17 18 23 24 25 26 28 31 32 35 36	7,649,000
B_x	1 2 6 8 9 11 14 17 20 22 24 26	6,572,000
B_y	5 7 13 16 19 23 27 28 33 34 35 36	7,037,000
B_z	3 4 10 12 15 18 21 25 29 30 31 32	1,520,000
C_a	20 24 29 31 33 35	2,765,000
C_b	6 8 10 12 13 16	3,570,000
C_c	22 26 30 32 34 36	3,605,000
C_d	14 17 21 25 27 28	2,549,000
C_e	1 2 3 4 5 7	1,811,000
C_f	9 11 15 18 19 23	829,000

All that remains is to find the actual categories corresponding to the weights. Comparing Table 19.5 to Table 19.3 yields that $A_x = A_1$, $A_y = A_2$, $B_x = B_1$, $B_y = B_2$, $B_z = B_3$, $C_a = C_1$, $C_b = C_2$, $C_c = C_3$, $C_d = C_4$, $C_e = C_5$ and $C_f = C_6$. So, we have matched the weights to the strata. For instance, weight 1 corresponds to categories A_1, B_1 and C_5. ■

Example 19.3:

The third example illustrates how variables and categories could be recognised when there are several variables with the same number of categories. In this example we apply method 2 of Section 19.2 to derive additional identifying information from the sampling weights. Of course, we could have applied method 1 as well.

Suppose that three auxiliary variables A, B and C have been used. The number of categories A is two and of both B and C four. So, $n_1 = 2$, $n_2 = 4$ and $n_3 = 4$. We suppose that linear weighting has been used. Suppose, furthermore, that the frequencies of the categories of the auxiliary variables in the population are given in Table 19.6.

Table 19.6. Frequencies of the categories of the auxiliary variables in the population. (Known to the intruder)

Category	Frequency in the population
A_1	1,485,135
A_2	1,514,865
B_1	754,875
B_2	735,023
B_3	775,036
B_4	735,066
C_1	735,443
C_2	784,387
C_3	745,122
C_4	735,048

The weights are listed in ascending order in Table 19.7.

Table 19.7. Weights of the strata.

Nr.	Frequency in sample	Weight	Weight × Frequency
1	96	932.4877	89,518.82
2	89	933.1395	83,049.42
3	97	944.7638	91,642.09
4	85	945.4156	80,360.33
5	104	952.5411	99,064.28
6	105	959.1034	100,705.86
7	97	964.8172	93,587.27
8	97	969.4501	94,036.66
9	93	970.1019	90,219.48
10	100	971.3795	97,137.95
11	109	981.7261	107,008.15
12	102	982.3780	100,202.55
13	102	989.5035	100,929.36
14	88	995.7425	87,625.34
15	93	996.0658	92,634.12
16	97	996.3943	96,650.25
17	87	1,001.7796	87,154.83
18	88	1,008.0186	88,705.64
19	102	1,008.3419	102,850.87
20	87	1,008.6704	87,754.33
21	90	1,015.7959	91,421.63
22	87	1,022.3582	88,945.17
23	90	1,028.0720	92,526.48
24	98	1,034.6343	101,394.16
25	88	1,034.7040	91,053.95
26	97	1,035.3558	100,429.51
27	82	1,046.9801	85,852.36
28	92	1,047.6319	96,382.13
29	78	1,054.7574	82,271.08
30	91	1,061.3197	96,580.09
31	92	1,067.0335	98,167.08
32	97	1,073.5958	104,138.79

Part of the differences list is presented in Appendix B. Only differences less than zero that occur 8 ($= n_1 \times n_2 = n_1 \times n_3$) or 16 ($= n_2 \times n_3$) times are listed.

Examining the list in Appendix B we see, for instance, that the difference $W_1 - W_{25}$ occurs 8 times. So, W_1 and W_{25} have 2 categories in common. Because $W_1 - W_{25} = W_2 - W_{26} = W_3 - W_{27} = W_4 - W_{28} = W_5 - W_{29} = W_6 - W_{30} = W_7 - W_{31} = W_{10} - W_{32}$, weights W_1, W_2,

W_3, W_4, W_5, W_6, W_7 and W_{10} have the same category in common. In this way we can determine all the groups that have the same category in common. These groups are listed in Appendix C. Apart from the indices of the sampling weights the products of these sampling weights and their frequencies in the sample are also listed in Appendix C. Moreover, the sum of these products for each group is listed. Such a sum is equal to the frequency of a category of the auxiliary variables in the population in the case that all the weights in the corresponding group have this category in common.

Comparing the sums of each group in Appendix C to Table 19.6 yields that $A_x = A_1$, $A_y = A_2$, $X_1 = C_1$, $X_2 = C_4$, $X_3 = C_3$, $X_4 = C_2$, $Y_1 = B_4$, $Y_2 = B_3$, $Y_3 = B_2$ and $Y_4 = B_1$. So, we have matched the weights to the strata. For instance, weight W_1 corresponds to categories A_1, B_4 and C_1.

In example 19.3 it is very easy to apply method 2 described in Section 19.3. If method 2 were applied to the second example, then some groups would have been constructed that do not have the same category in common. The sum of the products of the sampling weights in such a group and their frequencies in the sample would, however, differ from any frequency of a category of the auxiliary variables in the population. So, it is still possible to match the sampling weights to the categories of the auxiliary variables in the same way as above.

19.4. Possible SDC-measures

In the previous sections we have shown that sampling weights can provide additional identifying information to an intruder. Therefore, sampling weights should not be released without taking specific SDC-measures.

Two techniques to reduce the risk of disclosure caused by sampling weights can be applied. The aim of these techniques is to prevent the successful application of either of the two methods available to an intruder described in Section 19.2.

The first SDC technique is subsampling the records with a relatively low weight in the microdata set and then re-calculate the weights for the remaining records. As a consequence the weights of the remaining records with originally low weights are increased. In this way one can make all the weights of the records approximately equal and then discard them. In that case the weights do not provide any information. One can also make all the weights of the records almost equal in a first step and then apply the second method that we will describe, namely adding noise to the (adapted) weights. Subsampling leads to a loss of information, of course, because several records are not published.

The second SDC technique to reduce the risk of disclosure caused by sampling weights is adding noise to these weights. Suppose that the statistical office decides to add noise to the sampling weights. In other words, instead of releasing the true sampling weight W_i of record i the statistical office publishes values $W_i' = W_i + \varepsilon_i$ where ε_i is a random value. The question is how much noise should be added. To avoid having to add too much noise, one should first of all make sure that the differences between the weights are not too large. In particular, suppose there are k different weights W_i (i=1,...,k) such that

$W_1 < W_2 < ... < W_k$. In this case, the difference between W_{i+1} and W_i (i=1,...,k-l) should not be too large. For instance, one could demand that $W_{i+1} - W_i < \delta$ for i=1...,k-l, where δ is a well-chosen threshold value. If this condition is not satisfied one should subsample the microdata set until it is satisfied.

Only when the above condition is fulfilled one should consider adding noise to the sampling weights. Unfortunately, it is not clear at the moment how the value of δ should be chosen.

When noise is added to the sampling weights these weights are perturbed. Because generally all these perturbed weights will have a different value, the methods described in Section 19.2 cannot be applied immediately. So, obtaining additional identifying information is made (much) more difficult for an intruder. When an intruder wants to apply one of the methods of Section 19.2 to extract identifying information from the perturbed weights he must first cluster the perturbed weights into groups that are likely to consist of perturbed weights obtained from an equal original sampling weight, and he must estimate the original sampling weights. So, given the published perturbed weights W_i' an intruder must estimate the original sampling weights and he must estimate the corresponding frequencies in the microdata set. When the intruder succeeds in obtaining good estimates for these numbers, then he can apply one of the methods of Section 19.2. In that case the intruder will only obtain estimates for the population frequencies of the categories of the auxiliary variables after the method has been applied. So, only if the estimates of the original sampling weights, of their corresponding frequencies in the microdata set and of the frequencies of the categories of the auxiliary variables in the population are close enough to the true values the intruder will finally be able to derive additional identifying information.

Note that the sampling weights should be perturbed more in the case of post-stratification than in the case of multiplicative or linear weighting. To obtain additional identifying information in the case of post-stratification perturbed weights that are approximately equal have to clustered. The number of clusters constructed in this way should be equal to the number of strata. Based on these clusters an attacker can estimate the true sampling weights. The size of a cluster is the estimated frequency of the corresponding sampling weight in the sample. By multiplying the estimated sampling weights by their estimated frequencies in the sample an attacker may be able to match the clusters to the strata.

In the case of multiplicative or linear weighting, however, the situation is more difficult for an attacker. Again he may begin by constructing clusters of perturbed weights that are approximately equal. Based on these clusters he can estimate the true sampling weights. The size of a cluster is again the estimated frequency of the corresponding sampling weight in the sample. Subsequently, he should use these estimates for the true sampling weights to make groups of sampling weights that have at least one particular category in common. So, compared to the situation in the case of post-stratification an attacker must make an additional step in order to obtain identifying information from the sampling weights. Moreover, this step is based on estimated, and thus unreliable, sampling weights. Finally, by multiplying each estimated sampling weight in such a group by its estimated frequency an attacker may be able to match the sampling weights to the strata.

As reliable estimates for the true sampling weights are essential in order to construct groups of weights that have at least one particular category in common in the case of multiplicative or linear weighting, one could consider adding (slightly) *biased* noise to the sampling weights. As a result the estimates for the true sampling weights based on clusters of perturbed weights that are approximately equal will be biased. Even when these clusters are constructed perfectly, i.e. each cluster consists of perturbed weights that are obtained from the same true sampling weight, the attacker is unlikely to be able to match the sampling weights to the strata. The reason for this is that he will presumably not succeed in making groups of sampling weights that have a particular category in common, because the estimates for the true sampling weights are rather bad.

It is hard to quantify the probability that an intruder will be able to obtain additional identifying information from the sampling weights when noise has been added. In the case of post-stratification it is clear, however, that the perturbed weights should be sufficiently mingled. Suppose the sampling weights are ordered according to their values. When W_1 and W_2 are two consecutive sampling weights, where $W_2 \geq W_1$, then a sufficiently large number of perturbed weights $W_1 + \varepsilon_{1i}$ should be larger than a sufficiently large number of perturbed weights $W_2 + \varepsilon_{2j}$. As already noted, to quantify what is meant by 'sufficiently' is hard, though. In the case of linear/multiplicative weighting the differences, respectively ratios, of the perturbed weights should be sufficiently mingled in order to avoid the possibility that additional identifying information is derived from the (perturbed) sampling weights by means of one of the two methods described in Section 19.2. Again it is hard to quantify 'sufficiently mingled'.

Part of the problem can be solved by discriminant analysis theory. Suppose that an intruder is able to estimate the weights W_i and their frequencies π_i correctly. The only remaining problem for him to solve would be to allocate the perturbed weights to the correct weights. Suppose, furthermore, that the intruder knows the probability density function of the added noise to weight W_i. So, he also knows the probability density function $f_i(x)$ of the perturbed weights $\hat{W}_i$ that have been obtained from weight W_i. In this case, the intruder can apply the 'Bayes discriminant rule' to allocate the perturbed weights to the correct weights (cf. Mardia, Kent and Bibby, 1979, p. 304-307). As prior probabilities he can use the π_i.

When we define $\varphi_j(x)$ by

$$\varphi_j(x) = \begin{cases} 1 & \text{if } \pi_j f_j(x) = \max_i \pi_i f_i(x) \\ 0 & \text{otherwise.} \end{cases} \tag{19.9}$$

then the probability of misallocating a perturbed weight $\hat{W}_i$ that is obtained from weight W_i is given by

$$P_i = 1 - \int \varphi_i(x) f_i(x) dx \,. \tag{19.10}$$

As a lot of (somewhat unrealistic) conditions have to be fulfilled before an intruder can apply the 'Bayes discriminant rule' the probability of misallocation will generally be larger than P_i. However, in practice one may use (1 - P_i) as an upper bound on the probability that an intruder will be able to obtain additional identifying information from the sampling weights. When the numbers P_i are sufficiently high, i.e. higher than a certain threshold value, then the sampling weights are considered safe, and can be released. Note that the threshold value may be lower in the case of linear/multiplicative weighting than in the case of post-stratification, because in the former case the conditions that have to be fulfilled before an intruder can apply the 'Bayes discriminant rule' are more unrealistic than in the latter case. The number of steps that have to be made by an intruder before he can allocate the perturbed sampling weights to the strata is simply larger in the case of linear/multiplicative weighting than in the case of post-stratification.

We have already noted that in the case of linear/multiplicative weighting it is sufficient that the differences, respectively ratios, of the perturbed weights are "sufficiently mingled" in order to avoid the successful application of the methods of Section 19.2. This can be achieved in two steps. First, determine how much noise should be added to the differences, respectively ratios, to guarantee that they are sufficiently mingled. Second, determine how much noise should be added to the sampling weights in order to obtain the noise, determined in the first step, on the differences or ratios, respectively. We will illustrate the second step of this procedure below. It is assumed that the first step of the procedure is completed. This first step can be made by using (1 - P_i), where P_i is given by (19.10), as an upper bound on the probability that an intruder will be able to obtain additional identifying information from the sampling weights, for instance.

Consider the case of multiplicative weighting. Suppose we need to add the noise ε_R to the ratio $R = W_1/W_2$ in order to safeguard this ratio. So, instead of releasing R we release $R + \varepsilon_R$. We want to achieve this noise ε_R on the ratio R by adding noise to the weights W_1 and W_2, i.e. we want the following relation to hold

$$R + \varepsilon_R = \frac{W_1 + \varepsilon_1}{W_2 + \varepsilon_2}. \tag{19.11}$$

The right-hand side of (19.11) can be written as follows.

$$\frac{W_1 + \varepsilon_1}{W_2 + \varepsilon_2} = \frac{W_1 + \varepsilon_1}{W_2(1 + \varepsilon_2/W_2)} = \frac{W_1 + \varepsilon_1}{W_2}(1 - \varepsilon_2/W_2 + (\varepsilon_2/W_2)^2 - ...). \tag{19.12}$$

Assuming ε_1 and ε_2 to be small in comparison to W_1 and W_2, respectively, we find

$$\frac{W_1 + \varepsilon_1}{W_2 + \varepsilon_2} \approx \frac{W_1}{W_2} + \frac{\varepsilon_1 W_1 - \varepsilon_2 W_2}{W_2^2}. \tag{19.13}$$

Therefore,

$$\varepsilon_R \approx \frac{\varepsilon_1 W_1 - \varepsilon_2 W_2}{W_2^2}. \tag{19.14}$$

Now suppose that the noises ε_1 and ε_2 are realisations of stochastic variables $\underline{\varepsilon}_1$ and $\underline{\varepsilon}_2$ respectively, where $\underline{\varepsilon}_i$ is $N(0,\sigma_i^2)$ distributed. The stochastic variable $\underline{\varepsilon}_R$ is then approximately $N(0,\tau^2)$ distributed, where τ^2 is given by

$$\tau^2 = \frac{W_2^2\sigma_1^2 + W_1^2\sigma_2^2}{W_2^4}. \tag{19.15}$$

We have assumed that the first step of the procedure is completed, i.e. we know how large τ^2 should be. In this case we can determine (an infinite number of) pairs of values of σ_1^2 and σ_2^2 that satisfy (19.15). We can prescribe additional relations that have to be satisfied by σ_1^2 and σ_2^2. A possible choice for these relations is given by

$$\sigma_i^2 = \lambda W_i^2 \qquad (i=1,2), \tag{19.16}$$

where λ is a constant. Note that other choices for σ_i^2, such as $\sigma_i^2 = \lambda W_i$ are also possible. The relation expressed by (19.16), however, seems to be rather natural to us. Moreover, it leads to a simple expression for the constant λ, namely combining (19.15) and (19.16) yields that λ is given by

$$\lambda = \tau^2/(2R^2). \tag{19.17}$$

So, when the noise that is added to weight W_i has a $N(0,\tau^2W_i^2/(2R^2))$ distribution (i=1,2), then the noise ε_R that is actually added to the ratio $R = W_1/W_2$ is approximately $N(0,\tau^2)$ distributed. In other words, by adding a noise that is $N(0,\tau^2W_i^2/(2R^2))$ distributed to weights W_i (i=1,2) the corresponding noise that is added to the ratio $R = W_1/W_2$ has the distribution that has been determined in the first step of the procedure.

A practical problem remains. Suppose noise has to be added to the ratios W_1/W_2 and W_3/W_2, then we find two different values for the variance of the noise that has to be added to the weight W_2. In that case we take the larger of these two values. As a result both ratios W_1/W_2 and W_3/W_2 will be protected sufficiently.

Another problem when noise is added to the sampling weights, apart from evaluating the probability that an intruder can obtain additional identifying information from these weights, is the quality of the perturbed weights. When much noise is added to the sampling weights the resulting perturbed weights will hardly be useful for subsequent analysis. When little noise is added, the weights will remain useful for analysis, but the probability that an intruder can obtain additional information will be relatively high. How much noise should be added in order to obtain 'safe' and useful weights is a problem that remains to be solved.

19.5. Conclusions

The main conclusion of this chapter is that sampling weights can provide additional information in most practical cases when SDC-measures have not been taken. It has been proved that this holds true for any combination of numbers of categories of the auxiliary variables.

Therefore, SDC-measures have to be taken to prevent the derivation of additional identifying information from the sampling weights. Two techniques, subsampling and adding noise, have been discussed in general. These techniques should usually be applied in combination in the case of post-stratification. To quantify the protection offered by these two techniques and the quality of the sampling weights is hard. In practice one can use the probabilities P_i given by (19.9) as a lower bound on the probability that an intruder will be unable to match a weight W_i to its corresponding stratum. Better measures for the probability that an intruder is able to derive additional identifying information from the sampling weights should be developed in due course.

A module to deal with sampling weights has been built into version 3.1 of μ-ARGUS (see Hundepool et al., 2002b). This module allows one to add noise to the sampling weights. The amount of noise that has to be added has to be determined by the user himself.

20. Cell Suppression: Problem Formulation and a Practical Solution

20.1. Introduction

Sande (2000b) gives several examples of blunders made by statistical offices while trying to protect their tables against disclosure of confidential data. He shows that some of the tables that have been published by statistical offices still contain sensitive information and were not adequately protected. Considering the fact that ample attention has been given to the area of confidentiality of tabular data it is quite remarkable that Sande is able to point out so many mistakes and blunders. It even raises the question whether the most commonly used disclosure limitation technique for tabular data, cell suppression, might be too difficult for statistical offices to apply. It also raises the question if it is possible at all to develop cell suppression software that completely protects tables. In this chapter we will examine this latter question.

One of the reasons why cell suppression is so hard to apply correctly in practice is that the cell suppression problem itself is almost always described incorrectly in literature (see for example Kelly, Golden and Assad, 1992; Duarte De Carvalho, Dellaert and De Sanches Osório, 1994; Cox, 1995a; Dellaert and Luijten, 1999; Fischetti and Salazar-González, 2000). Even the author of the present book is guilty in this respect (see Willenborg and De Waal, 1996 and 2001). This incorrect description is based on an incorrect definition of safe suppression patterns. Suppression patterns that are safe according to this definition should not always be considered safe, and vice versa. Not everyone is to blame for giving an incorrect description of the cell suppression. For instance, Sande already gave a (nearly) accurate description of the cell suppression problem in the 1970's.

In Section 20.2 of this chapter we give a formulation for the cell suppression problem that is correct in our point of view. As far as we are aware this is the first time that this formulation for the cell suppression problem is given in literature. The incorrect formulation that is commonly used is a simplification of the correct formulation. This simplification is treated in Section 20.3. To check whether a table is safe according to the correct definition so-called elementary aggregations have to be generated and checked. Section 20.4 briefly discusses these elementary aggregations. Elementary aggregations have already been introduced by Sande (1977, 1978b and 1978c). He, however, restricted attention to elementary aggregations with only non-negative coefficients. We show that elementary aggregations involving also negative coefficients are often required, and extend the concept of elementary aggregations accordingly. Section 20.5 shows that it is indeed sufficient to check these elementary aggregations for safety. In particular this section shows that if the elementary aggregations are safe, then the so-called suppression intervals of individual sensitive cells have a certain minimum width. The proof in that section has essentially been given by Sande (1978c). We have restated this proof in order to make it more easily accessible. Section 20.6 examines how to protect unsafe cells and unsafe elementary aggregations. The major part of that section has been described in literature before; our contribution is restricted to clearly stating how the same method can be applied

to general aggregations. Section 20.7 discusses a way to construct completely safe cell suppression software. Finally, Section 20.8 concludes the chapter with a brief discussion.

20.2. The cell suppression problem

The basis for cell suppression is a sensitivity measure for individual cells. Such a sensitivity measure determines whether a cell is safe, and hence whether its value may be published. If a cell is considered unsafe (sensitive), its value has to be suppressed. There are several classes of sensitivity measures. The best-known ones are the dominance rule and the prior/posterior rule (see for example Section 14.8 and Willenborg and De Waal, 1996 and 2001). In this chapter we will restrict ourselves to these classes of sensitivity rules. In particular, we will assume that all tables in this chapter have non-negative cell values.

The values of the sensitive cells are suppressed. This is referred to as primary suppression. In addition, usually a number of non-sensitive cells has to be suppressed in order to prevent the possibility of precise re-calculation of the suppressed sensitive cell values by an intruder. This is called secondary suppression. The cell suppression problem amounts to finding a good set of secondarily suppressed cells.

Given a suppression pattern of primarily and secondarily suppressed cells, one can determine for each sensitive cell an interval of feasible values that this suppressed sensitive value can assume, the so-called suppression interval of the sensitive cell. In the usual, incorrect description of the cell suppression problem one only demands that the suppression interval contains an interval of prescribed width. Example 20.2.1 below illustrates the procedure.

Example 20.2.1:
Suppose we consider a cell to be unsafe if three or less respondents contribute at least 70% to the cell total. This is the (3,70)-dominance rule. We also demand that the suppression interval of each unsafe cell should have a width of at least 50% of the cell value.

We apply these rules to the table below.

	I	*II*	*III*	Total
A	100	200	150	450
B	250	150	300	700
C	600	450	500	1150
Total	950	800	950	2700

Figure 20.2.1. Unsafe table T_1

Suppose the unsafe cells are *A*×*I* and *A*×*III*. We also suppose that to each of these cells only one respondent contributes. We protect the table by suppressing these cells, and a number of additional cell values. Suppose we obtain the following table.

	I	*II*	*III*	Total
A	×	200	×	450
B	×	150	×	700
C	600	450	500	1150
Total	950	800	950	2700

Figure 20.2.2. Table T_1 "protected version"

The suppression intervals are given in Figure 20.2.3.

	I	*II*	*III*	Total
A	*[0, 250]*	[200, 200]	*[0, 250]*	[450, 450]
B	*[100, 350]*	[150, 150]	*[200, 450]*	[700, 700]
C	[600, 600]	[450, 450]	[500, 500]	[1150, 1150]
Total	[950, 950]	[800, 800]	[950, 950]	[2700, 2700]

Figure 20.2.3. Suppression intervals corresponding to Table T_1 "protected version".

Neither of the unsafe cell values can be determined to within 50% of its actual cell value. The "protected version" of Table T_1 is hence considered safe according to the applied rule for the widths of the suppression intervals. ■

Applying both a sensitivity measure as well as a rule for the widths of the suppression intervals may, however, lead to aggregations of cells that are not safe according to the sensitivity rule. The rule for the width of the suppression intervals may hence be inconsistent with the sensitivity measure.

Example 20.2.2:
From the "protected version" of Table T_1 we can derive the total value of the aggregation of cells *A*×*I* and *A*×*III*. The value of this aggregation, which we can consider to be an ad-hoc cell, equals 250. We therefore know the exact sum of the contributions of the two respondents in cells *A*×*I* and *A*×*III*. So, the "protected version" of Table T_1 is not protected

at all according to our sensitivity measure that says that a cell is unsafe if three or less respondents contribute at least 70% to the total cell value!

Gordon Sande (2000b) calls this phenomenon an "ad-hoc roll up", and gives a number of examples in publications of various statistical offices. ■

To avoid inconsistency between the sensitivity measure and the rule for the widths of the suppression intervals of sensitive cells, one should abandon the idea of using a rule for the widths of the suppression intervals of sensitive cells all together. One should exclusively use a sensitivity measure.

Informally, we now define a table to be safe if and only if for no respondent too much information can be derived according to the applied sensitivity measure. Equivalent to this definition is the following, a bit more practical definition.

Definition 20.2.1: A table is safe if and only if all aggregations of suppressed cells of which the exact value can be derived from the table are safe according to the sensitivity measure.

When this definition is used to determine whether a suppression pattern is safe or not, inconsistency between different safety rules obviously cannot occur.

Most sensitivity measures as described in literature only make sense for sums of cells. Therefore, to apply Definition 20.2.1 the sensitivity measure used has to be extended in order to make sense for more general combinations of cells, for example for differences of sums of cells. Daalmans (2002) describes several sensible possibilities to extend the (n,k)-dominance rule and the prior-posterior rule to general linear combinations of cells. In the sequel of this chapter, we will assume that the applied sensitivity measures are indeed defined for general combinations of cells, and we will use Definition 20.2.1 to determine whether a table is safe.

The cell suppression problem now consists of determining a safe suppression pattern for the table, or set of linked tables, under consideration while keeping the information loss due to suppression as low as possible. To determine the information loss due to cell suppression a suitable information loss measure has to be specified. Examples of such measures are: the total number of suppressed cells, the total value of the suppressed cells, and a weighted mix of the number of suppressed cells and the total value of the suppressed cells.

Determination of a suppression pattern that protects a given table and leads to a minimum information loss is a very complicated problem in general. Even determining whether a table is sufficiently protected given a certain suppression pattern turns out to be a complicated mathematical problem. Section 20.4 very briefly considers the problem of determining whether a table is safe, while Sections 20.6 and 20.7 consider how to solve the cell suppression problem to suboptimality.

20.3. The standard simplification of the problem

In the standard approach to the cell suppression problem this problem is simplified considerably by replacing the condition for a safe table mentioned in Definition 20.2.1 by the condition that the upper bound on the suppression interval of each sensitive cell is at least equal to that value for which the cell would be safe according to the sensitivity measure. Implicitly, this rule assumes that a sensitive cell is protected by suppressing some additional cells with relatively small contributions. In many cases this rule is sufficient for protecting the sensitive cell at hand.

Example 20.3.1:
Suppose we use the same sensitivity measure as in the preceding examples. That is, we consider a cell to be unsafe if three or less respondents contribute at least 70% to the total cell value. If a cell is unsafe, we suppress its value and demand that the upper bound on its suppression interval is at least equal to that value for which the cell would be safe according to the dominance rule.

Consider the table below in which only cell $A \times I$ is unsafe. To this cell 16 respondents contribute, of which the largest three contribute 133 (so more than 70% of the total cell value) to the cell value.

	I	*II*	*III*	Total
A	160	380	340	880
B	40	80	60	180
C	610	800	270	1680
Total	810	1260	670	2740

Figure 20.3.1. Unsafe table T_2

According to the sensitivity measure, the upper bound on the suppression interval should be at least 190 (= 133/0.7). This is achieved by the following suppression pattern.

	I	*II*	*III*	Total
A	×	×	340	880
B	×	×	60	180
C	610	800	270	1680
Total	810	1260	670	2740

Figure 20.3.2. Table T_2 "protected version"

If each of the three largest contributions to $A \times I$ is larger than the largest contribution to $A \times II$, the largest contribution to $B \times I$, and the largest contribution to $B \times II$, then the suppression pattern is safe according to Definition 20.2.1, as can be demonstrated quite easily. ■

Unfortunately, the above rule is not always sufficient. This can be the case for the (n,k)-dominance rule if not all n largest contributions to an unsafe cell are larger than the largest contribution to another suppressed cell. For the prior-posterior rule the above rule can be insufficient if the second largest contribution to the sensitive cell is less than the largest contribution to another suppressed cell.

Example 20.3.2:
We continue Example 20.3.1. Suppose the largest three contributions to cell $A \times I$ in Table T_2 equal 74, 50 and 9, respectively. Suppose, furthermore, that the largest three contributions to cell $B \times I$ equal 18, 5 and 3. If we consider the suppression pattern of Figure 20.3.2, we see that the aggregation of cells $A \times I$ and $B \times I$ has a total value of 200. The largest three contributions to this aggregation sum up to 142 (74+50+18), more than 70% of the total value. According to our sensitivity measure the "protected version" of Table T_2 would hence be unsafe. ■

Although the rule that the upper bound on the suppression interval of an unsafe cell should at least be equal to that value for which this cell would be safe according to the sensitivity measure used is not always sufficient, it does give a good approximation for the correct, more stringent rule. In several cell suppression software packages, such as CONFID (see Robertson, 1992, 1995, and 2000) and ACS (see Sande, 1984 and 1999), this approximation is used instead of the correct, more stringent rule. The approximation is improved by adding some additional constraints to the optimisation problem. The final result is a table that is (almost) completely protected. In Section 20.6 we briefly sketch the CONFID/ACS approach.

Note that in some cases the approximating rule cannot be applied immediately. This is, for instance, the case if we have an unsafe cell with only one contribution (say with value 140), and we apply a (3,70)-dominance rule. For no cell value would this cell ever be considered safe, simply because the cell contains only one contribution. Without knowledge about which other cells are suppressed we cannot specify the upper bound on the suppression interval of the unsafe cell, because the required upper bound depends on the additional cell suppressions.

For such cases we use another approximation suggested by Daalmans: we do not neglect the total protection offered by suppressing a cell, but we do neglect each individual contribution to that cell. For instance, in the above example of the unsafe cell with only one contribution, we would require that the upper bound on the suppression interval should be at least 200 (=140/0.7). Note that an upper bound of 200 will always be too small, because we have neglected the individual values of the additionally suppressed

contributions. However, an upper bound of 200 is the best valid bound we can define easily. In the sequel we will use similar upper bounds.

20.4. Elementary aggregations

As we mentioned before, even determining whether a table is sufficiently protected is a complicated mathematical problem. The problem here is to determine, in principle, all aggregations for which the exact value can be derived from the table. The number of such aggregations may be extremely high. The good news is that not all aggregations have to be actually derived. For certain, natural classes of sensitivity measures it suffices to determine so-called *elementary* aggregations (see Sande, 1977, 1978b, and 1978c; Daalmans, 2002). An aggregation A for which the exact value can be derived from the table is called an elementary aggregation if it cannot be split into aggregations A_1 and A_2 for which the exact values can be derived from the table such that both A_1 and A_2 involve a strict subset of the variables involved in A.

Example 20.4.1:

In Figure 20.3.2 we have a number of elementary aggregations, for example the four aggregations consisting of *A*×*I* and *A*×*II*, *A*×*I* and *B*×*I*, *A*×*II* and *B*×*II*, and *B*×*I* and *B*×*II*. For example, the aggregation "*A*×*I* and *A*×*II*" is an elementary one, because this aggregation can only be split into "*A*×*I*" and "*A*×*II*" and the value of neither "*A*×*I*" nor "*A*×*II*" can be derived exactly from the table. ■

The bad news is that elementary aggregations can involve cells that occur throughout the table. That is, an elementary aggregation does not necessarily involve only cells that occur in the same row or column, but can in principle involve cells from any row or column. An example of a table in which the cells involved in an elementary aggregation are not restricted to a single row or column is given below.

Example 20.4.2:

	I	*II*	*III*	Total
A	×	190	×	440
B	20	×	×	90
C	×	×	135	840
Total	405	630	335	1370

Figure 20.4.1. Elementary aggregations in a table

It is easy to verify that the sum of *A*×*I* and *B*×*II* forms an elementary aggregation. The sum of these two cell values exactly equals 120 (this can be shown by, for example, adding the

first two rows to each other and subtracting the third column). Without further knowledge about the data of these two cells, we cannot decide whether this aggregation is safe or not. ■

As mentioned before, for certain classes of sensitivity measures, a table turns out to safe if and only if all its elementary aggregations are safe according to the sensitivity measure (see Daalmans, 2002). From now on we restrict ourselves to these classes of sensitivity measures. To check whether a table is safe the sensitivity measure has to be applied to all its elementary aggregations. If all elementary aggregations are safe, automatically all aggregations (elementary and non-elementary ones) are safe, and hence the table is safe. Otherwise, the table is unsafe and may not be published in this form.

To check whether a table with suppressed cell values is safe, it remains to specify a method to determine all elementary aggregations. Sande (1977, 1978b and 1978c) describes a method to determine all elementary aggregations with non-negative coefficients corresponding to a given suppression pattern. Elementary aggregations with non-negative coefficients are characterised as extremal directions of a certain cone. To determine these extremal directions Chernikova's algorithm is used (see Chernikova, 1964 and 1965; see also Chapter 5 of this book).

However, also elementary aggregations with negative coefficients exist. An example is given below.

Example 20.4.3:

	I	*II*	*III*	Total
A	70	×	×	440
B	×	50	×	90
C	×	×	135	840
Total	405	630	335	1370

Figure 20.4.2. Elementary aggregations with negative coefficients

We denote the value of a cell in the *i*-th row and *j*-th column by x_{ij}. We can derive that $x_{12} - x_{23} = 170$ (for example by subtracting the third column from the first row). *A*×*II* **minus** *B*×*III* is hence an elementary aggregation. ■

The safety of these elementary aggregations with negative coefficients also needs to be checked. For instance, if in Example 20.4.3 only one respondent contributes to cell *A*×*II* and only one respondent to *B*×*III*, then these two respondents can derive each other's contribution. This would then be a typical example of an unsafe table.

In principle, Sande's method can be used to find all elementary aggregations (also the ones with some negative coefficients) for many tables by taking differences of elementary aggregations involving only non-negative coefficients. For instance, in Figure 20.4.2 Sande's method will, amongst others, identify the following elementary aggregations with non-negative coefficients: $x_{12} + x_{13} = 370$ and $x_{13} + x_{23} = 200$. By taking the difference of these elementary aggregations, we find the elementary aggregation with negative coefficients mentioned in Example 20.4.3: $x_{12} - x_{23} = 170$.

There are two problems with Sande's method. The first problem is that there are tables for which Sande's method cannot generate all elementary aggregations. Admittedly, such tables seem to be rare. A simple example is given below.

Example 20.4.4:

	I	*II*	Total
A	10	×	×
B	10	40	50
Total	20	×	×

Figure 20.4.3. Only elementary aggregations with negative coefficients

In Figure 20.4.3 we have the following elementary aggregations: $x_{13} - x_{12} = 10$, $x_{33} - x_{32} = 20$, $x_{32} - x_{12} = 40$ and $x_{33} - x_{13} = 50$. In all elementary aggregations negative coefficients occur so Sande's method will not find any elementary aggregations.

■

The second problem is that it is unclear – at least for us – how to determine all elementary aggregations from the elementary aggregations with non-negative coefficients only in a simple way. Daalmans (2002) describes an alternative method, which is based on a tree search, to generate all elementary aggregations that does not suffer from the problems mentioned, but may require a considerable amount of computing time.

20.5. Widths of suppression intervals

One may wonder whether it is sufficient to ensure only that all (elementary) aggregations for which the value can be derived exactly from the table are safe according to the sensitivity measure. In particular, one may wonder whether if all such aggregations are safe according to the sensitivity measure, suppression intervals for sensitive cells might still be very narrow, thereby allowing individual contributions to be estimated accurately. Fortunately, this cannot really occur. This has been demonstrated by Sande (1977 and 1978c). The theorem below is based on Sande (1978c).

Theorem 20.1. If the only a-priori knowledge of each cell is that its value is non-negative, then the maximum of the suppression interval of a suppressed cell equals the smallest value of all elementary aggregations with non-negative coefficients in which this suppressed cell occurs.

Proof. The wording of the proof given here differs slightly from Sande's proof, but the argument is the same. To prove the assertion we use Fourier-Motzkin elimination (see Fourier, 1826; Duffin, 1974; Chvátal, 1983; Schrijver, 1986; see also Chapter 8 of this book), extended to equations.

The standard form of Fourier-Motzkin elimination can only be applied to inequalities. The technique is quite simple. Suppose two inequalities are given by

$$x_s \geq \sum_{i \neq s} a_i x_i + b \tag{20.1}$$

and

$$\sum_{i \neq s} a'_i x_i + b' \geq x_s \ , \tag{20.2}$$

then we can construct the valid inequality

$$\sum_{i \neq s} a'_i x_i + b' \geq \sum_{i \neq s} a_i x_i + b \tag{20.3}$$

in which variable x_s does not occur. We say that variable x_s has been eliminated from (20.1) and (20.2). In a natural way the technique can be extended to equations. For instance, we can eliminate variable x_s from (20.1) and

$$x_s = \sum_{i \neq s} a''_i x_i + b'' \tag{20.4}$$

We then obtain the following valid inequality for the remaining variables:

$$\sum_{i \neq s} a''_i x_i + b'' \geq \sum_{i \neq s} a_i x_i + b \ . \tag{20.5}$$

Analogously, we can eliminate a variable from two equations. We then obtain a valid equation for the remaining variables.

When we eliminate all but one variable from a set of (in)equalities, we obtain a set of lower and upper bounds on the value of this variable (for ease of speaking: if equalities restrict the variable to take a particular value, we consider this value to be both a lower as well as an upper bound on the value of this variable). Fourier (1826) showed the following theorem.

If the lower and upper bounds on the value of a variable are contradicting each other, then the original set of (in)equalities is inconsistent. If the lower and upper bounds do not contradict each other, the variable can attain any value between the maximum of the lower bounds and the minimum of the upper bounds.

For the cell suppression problem, we have a set of equations given by

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,, \tag{20.6}$$

where **x** denotes a vector consisting of an unknown for each suppressed cell. The matrix **A** and the vector **b** are determined by the additivity constraints for the table and by the non-suppressed cell values. Besides (20.6) we have non-negativity constraints given by

$$\mathbf{x} \geq \mathbf{0}\,. \tag{20.7}$$

Suppose we want to determine the suppression interval of x_1. We can do this by applying Fourier-Motzkin elimination. We eliminate all variables except x_1, and examine the maximal lower bound and minimal upper bound on x_1. These bounds cannot be inconsistent, because the original, unprotected table is consistent.

To prove Sande's statement we apply Fourier-Motzkin elimination (extended to include equations) to the system given by (20.6) and (20.7). We do this in two steps. In the first step we apply Fourier-Motzkin elimination to (20.6) to eliminate all possible combinations of variables except those involving x_1 from (20.6). For instance, we eliminate variable x_2 from (20.6). This yields a set of equations. Similarly, we eliminate the pair x_2 and x_3 from (20.6). This also yields a set of equations. We do the same for all possible combinations except those involving x_1. Each combination of variables yields a set of equations without these variables. All these sets of equations are combined into one large system of equations. For convenience, we scale the equations such that the coefficient of x_1 equals 1 in each equation.

In the second step we use the inequalities (20.7) to eliminate all variables except x_1 from each equation of this large system. This results in lower and upper bounds on x_1. We examine which equations in the large system of equations lead to the strictest upper bounds on x_1. Note that we only have to examine the equations involving only non-negative coefficients. Namely, if a variable with a negative coefficient were involved in such an equation, the upper bound on x_1 would be infinite. So, the strictest upper bounds on x_1 are derived from equations involving only non-negative coefficients.

Now, suppose that the strictest upper bound is obtained from an equation

$$x_1 + \sum_{i \neq 1} c_i x_i = d_i\,, \tag{20.8}$$

where $c_i \geq 0$. If this aggregation were not an elementary one, it could be split into several elementary aggregations, at least one of which involving x_1. We select one of those elementary aggregations that would give the strictest upper bound on x_1. This upper bound on x_1 is at least as strict as the upper bound obtained from (20.8). We therefore conclude that the strictest upper bound on x_1 is obtained from an elementary aggregation. This concludes the proof. ■

Sande (1978c) proves an even more general result. This result is stated below as Theorem 20.2, which can be proved in the same manner as Theorem 20.1.

Theorem 20.2. If the only a-priori knowledge of each cell is that its value is non-negative, then the maximum of the suppression interval of an aggregation which is a sum of suppressed entries equals the minimum – over all decompositions into pieces consisting of elementary aggregations (with non-negative coefficients) – of the sum of the maximum values for each piece.

For the case where the a-priori knowledge of potential intruders is not limited to just non-negativity, but the potential intruders a-priori also know certain intervals in which the values of the cells lie, we can derive a similar result. We have to replace by (20.7) by

$$\mathbf{b}^{\mathbf{L}} \leq \mathbf{x} \leq \mathbf{b}^{\mathbf{U}} . \tag{20.9}$$

Now, lower and upper bounds for a suppressed cell i can be found in the second step by substituting the values b_j^L or b_j^U for the variables x_j occurring in the elementary aggregations obtained after the first step mentioned in the above proof.

20.6. Protecting unsafe objects

In this section and the next we explore the possibility of developing cell suppression software that generates completely safe suppression patterns. To explain how such software could be constructed, we will use the linear programming (LP) approach of CONFID (see e.g. Robertson, 1992, 1995 and 2000) or ACS (see e.g. Sande, 1984 and 1999) rather than a more complicated integer programming approach (see for example Kelly, Golden and Assad, 1992; Geurts, 1992; Fischetti and Salazar-González, 1998b and 2000). The approach sketched in this section is also implemented in a prototype computer program developed at Statistics Netherlands (see Traa, 2001). The approach is not (yet?) included in τ-ARGUS (see Hundepool et al., 2002a).

Because the cell suppression problem is (too) difficult to solve at once, in the LP approach one first tries to solve the standard simplification described in Section 20.3. To solve this simplified problem, several LP problems are constructed and solved. How the simplified problem can be solved is sketched in Subsection 20.6.1. The simplified problem is in general not solved to optimality.

Besides solving the simplified problem one also need to protect additional sensitive aggregations, for example sensitive aggregations in a single row or column. An approach is sketched in Subsection 20.6.2.

20.6.1. Protecting individual sensitive cells

In the LP approach, the sensitive cells are protected separately. At first, only the upper bound on the suppression interval of the sensitive cell to be protected is taken into account.

Other aspects of the cell suppression problem may be taken into account by later adding additional constraints to the mathematical model (see Subsection 20.6.2).

Protecting the individual sensitive cells is done in two phases, in the first phase for all sensitive cells a pattern of potentially suppressed cells is determined. The cells involved in this pattern are only *eligible* for suppression. In a second phase some of the cells eligible for suppression may be removed from the suppression pattern, the values of these cells are then published after all.

Suppose the minimal value for which the sensitive cell currently under consideration would have been safe equals u_0. That is, the sensitive cell would have been safe if there had been some extra contributions, each of which small in comparison to the actual contributions to the cell, such that the total value of the cell is at least u_0. This value u_0 will act as the minimum value for the upper bound on the suppression interval of the sensitive cell.

We now demand that

$$V + \Delta V \geq u_0 , \tag{20.10}$$

where V is the actual cell value, and ΔV a change in cell value. The values V and u_0 are fixed, and ΔV is an unknown that needs to be determined.

Note that Theorem 20.1 guarantees that if all elementary aggregations are sufficiently protected and u_0 as in (20.10) exists, then the upper bound on the suppression interval is at least u_0. That is, if all elementary aggregations are sufficiently protected, then (20.10) is automatically satisfied.

We have already mentioned that it can happen that there is no value u_0 for which the sensitive cell would have been safe, simply because there are not enough contributions to the sensitive cell. Following Daalmans' suggestion mentioned in Section 20.3 we then temporarily neglect the individual contributions to the additionally suppressed cells, but not the protection offered by the additionally suppressed cells. This suggestion allows us to compute a value u_0 that can be used in (20.10).

After constructing the constraint (20.10), we modify, in principle, all other cell values subject to the constraint that the table remains additive and that the cell values lie within bounds assumed to be known a priori by a potential intruder. For instance, if a dominance rule is used we demand that the lower bound on each cell value is zero, and if a prior-posterior rule with parameters p (allowed worst-case posterior knowledge) and q (prior knowledge) is used we demand that the value of each cell after adaptation lies within q per cent of the actual cell value. In the additivity constraints, a possible hierarchical structure is taken into account.

Subject to the above constraints we minimise the information loss. In the first phase we assume that this information loss is given by

$$\sum_i f_i(V_i) \times |\Delta V_i| , \tag{20.11}$$

where V_i is the value of the *i*-th cell in the table, ΔV_i the change in value of cell *i*, and $f_i(V_i)$ a function of cell value V_i that may depend on cell *i*. The ΔV_i are the unknowns of the LP-problem. The sum is taken over all cells in the table. A non-sensitive cell is potentially suppressed if $\Delta V_i \neq 0$. It is then added to the set of cells that are eligible for suppression.

In CONFID and ACS the following function f_i is used

$$f_i(V) = \log(V + 1) . \tag{20.12}$$

Marking a cell with a large value as eligible for suppression is hence considered to lead to more information loss than marking a cell with a small value as eligible for suppression. For sensitive cells the value of f_i is by definition set to zero, because sensitive cells have to be suppressed any way. Similarly, for non-sensitive cells that have already been marked as eligible for suppression while protecting another sensitive cell f_i is also set to zero.

As is standard in cell suppression software, cells with value zero are never suppressed by CONFID or ACS, because potential intruders may know that the value of such a cell equals zero. This can, for instance, be the case if there are simply no entities in the population who could contribute to this particular cell. A potential intruder may know this. If the cell had been suppressed to "protect" another cell, this knowledge might be used to undo the "protection".

After the first-phase LP problems have been solved for all sensitive cells, we have determined a pattern of potentially suppressed cells for each sensitive cell. Together these patterns of potentially suppressed cells form one large pattern of potentially suppressed cells, i.e. a set of cells that are eligible for suppression. Before we actually suppress any values, we first clean up the cells that are eligible for suppression. By combining several patterns of potentially suppressed cells into one large pattern of cells that are eligible for suppression, some of these cell values that were considered eligible for suppression may be published after all, because enough protection is already offered by other cells that are eligible for suppression.

During the second phase, the clean-up phase, only the cells that are eligible for suppression are considered. Again all sensitive cells are examined separately. For each sensitive cell the information loss given by

$$\sum_j g_j(V_j) \times |\Delta V_j| \tag{20.13}$$

is minimised, where V_j denotes the value of the *j*-th cell in the pattern of potentially suppressed cells after the first phase, ΔV_j the change in value of cell *j*, and $g_j(V_j)$ a function of the cell value V_j that may be depend on cell *j*. The sum is taken over all cells in the large pattern of potentially suppressed cells obtained after the first phase. A non-sensitive cell is secondarily suppressed to protect the sensitive cell under consideration if and only if $\Delta V_j \neq 0$. All sensitive cells are suppressed.

In CONFID and ACS the following function g_j is used:

$$g_j(V) = \log(V)/V \,. \tag{20.14}$$

In the clean-up phase suppression of a cell with a large value leads to a relatively small information loss. The rationale is that a cell with a large value was already eligible for suppression any way, and suppressing this cell may have the effect that cells with smaller values eligible for suppression do not have to be suppressed. In other words, in the second phase one tries to avoid having to suppress many cells with small values by suppressing some cells with relatively large values.

We have now explained how to protect sensitive cells. We have not considered how to protect sensitive aggregations. This question is considered in the next subsection.

20.6.2. Protecting aggregations

20.6.2.1. Protecting aggregations in a single row/column

To protect sensitive aggregations in a single row or column, we extend the set of LP problems to be solved. We simply construct an LP problem for each such aggregation.

Before we start solving the set of LP problems of the first phase we first examine for all sensitive cells which elementary aggregations involving this sensitive cell and cells in the same row, or column respectively, are sensitive as well. This examination can be performed fast by constructing a binary tree (see Robertson, 2000). Robertson (2000) notes that the total number of unsafe cells and unsafe aggregations involving cells in a single row or column is not much larger than the number of sensitive cells. For each unsafe aggregation we demand that the upper bound on the suppression interval of this aggregation is at least equal to that value for which the aggregation would have been safe according to the sensitivity measure. That is, we demand that a similar condition as for individual unsafe cells holds true.

For all cells in the table we introduce unknowns, and demand that

$$\sum_k (V_k + \Delta V_k) \geq u_0 \,, \tag{20.15}$$

where the sum is taken over all cells involved in the unsafe aggregation, and u_0 is that value for which this aggregation would have been safe if there had been some additional respondents, each with a relatively small, or even negligible, contribution.

Note that Theorem 20.2 guarantees that if all elementary aggregations are sufficiently protected and u_0 as in (20.15) exists, then (20.15) is automatically satisfied.

Apart from a slightly different constraint, sensitive aggregations involving only cells in a single row or column are treated similarly as sensitive individual cells. For all sensitive objects, i.e. sensitive cells as well as sensitive aggregations (in a single row or column), the cells eligible for suppression are determined in the first phase as described in the previous subsection. Next, for all sensitive objects the second, clean-up phase is performed.

In many cases the approach sketched in this subsection and the previous one will suffice to protect a table completely. Nevertheless, in some cases there may still be unsafe aggregations left in the table. According to our intuition such unsafe aggregations will only rarely occur in tables produced by national statistical institutes. For the sake of completeness we explain how to protect these unsafe aggregations in the next subsection.

20.6.2.2. Protecting general aggregations

Suppose one wants to protect an unsafe aggregation given by

$$\sum_k a_k x_k = b_k \ . \tag{20.16}$$

In the previous subsections we only had to solve one additional first-phase LP problem and a corresponding second-phase LP problem, because only a minimum value for the upper bound on the suppression interval had to be required. Here we may need to solve two additional first-phase and two corresponding second-phase problems, because both a maximum value for the lower bound on the suppression interval of (20.16) as well as a minimum value for the upper bound on the suppression interval of (20.16) may need to be demanded. Alternatively stated, we could say that we need to protect two unsafe objects: for one object the upper bound on (20.16) needs to be sufficiently high, for the other object the lower bound on (20.16) needs to be sufficiently low.

For all cells in the table we introduce unknowns, and demand that

$$\sum_k a_k (V_k + \Delta V_k) \geq u_0 \ , \tag{20.17}$$

where the sum is taken over all cells involved in the unsafe aggregation (20.16), and u_0 is that value for which aggregation (20.16) would have been safe if there had been some additional respondents, each with a relatively small, or negligible, contribution. If no upper bound on (20.16) is demanded, u_0 equals b_k and the corresponding LP problems can be discarded.

We also demand that

$$-\sum_k a_k (V_k + \Delta V_k) \geq u_0' \ , \tag{20.18}$$

where the sum is taken over all cells involved in the unsafe aggregation (20.16), and u_0' is that value for which the negative of aggregation (20.16) would have been safe if there had been some additional respondents, each with a relatively small, or negligible, contribution. If no lower bound on (20.16) is demanded, u_0' equals $-b_k$ and the corresponding LP problems can be discarded.

Note that Theorem 20.2 guarantees that if all elementary aggregations are sufficiently protected and u_0 as in (20.17) exists, then (20.17) is automatically satisfied. Similarly, Theorem 20.2 also guarantees that if all elementary aggregations are sufficiently protected and u_0' as in (20.18) exists, then (20.18) is automatically satisfied.

For each sensitive elementary aggregation we construct a constraint given by (20.17). Together with the additivity constraints and bounds for individual cells this constraint forms a system of constraints for an LP problem corresponding to this sensitive elementary aggregation. For each sensitive elementary aggregation we also construct a constraint given by (20.18). Together with the additivity constraints and bounds for individual cells this constraint forms a system of constraints for another LP problem corresponding to this sensitive elementary aggregation.

The aggregations (20.16) that need to be protected may either be specified by a user of the cell suppression software or by the cell suppression software itself (see Section 20.7). All specified sensitive objects are treated during the first phase to determine one large pattern of cells eligible for suppression, and later during the second phase to determine the final suppression pattern.

20.7. Completely safe cell suppression software

In principle, one could specify many, or possibly all, sensitive aggregations as objects to protect before the actual cell suppression process starts. There are at least two drawbacks to specifying many, or possibly all, sensitive aggregations as objects to protect before the cell suppression process starts. First, determining all elementary aggregations that require protection for an unprotected table may be (too) time-consuming. Because no suppression pattern is suggested, all possible elementary aggregations have to be considered. This in contrast to the situation where a suppression pattern is suggested, and it is checked whether the elementary aggregations occurring in this pattern are safe. In that case only the elementary aggregations that actually occur in the suppression pattern have to be determined. Second, specifying many aggregations that need to be protected will lead to an increase in the computing time required for the LP problems. For these reasons we will not explore this possibility of obtaining a completely safe table any further.

Alternatively, one could consider applying an iterative procedure. First, a suppression pattern would be determined such that all individual sensitive cells, all aggregations in the same row or column, and additional, e.g. user-specified, aggregations would be sufficiently protected. This suppression pattern would then be checked for the presence of remaining unsafe aggregations. For this, elementary aggregations would have to be generated. Subsequently, these elementary aggregations would be checked to decide whether they are sensitive or not.

Then a new iteration would start where all individual sensitive cells, all sensitive elementary aggregations involving cells in a single row or column, and all other sensitive elementary aggregations detected so far, would be protected. This process would continue until no sensitive aggregations occur in the table anymore. That table could then be published.

This approach can, however, only be applied for small tables. The problem is that a table of a moderate or large size may contain an enormous amount of elementary aggregations. Determining all these elementary aggregations and checking their safety may therefore require an excessive amount of computing time.

Fortunately, we can follow a third approach. To ensure that a table is completely protected by cell suppression, we could dynamically update the set of unsafe objects. At first a set of unsafe objects is determined or specified by the user. This set of unsafe objects will involve the unsafe individual cells, the unsafe aggregations involving cells in the same row/column and possibly some other unsafe aggregations. We construct the LP problems for these unsafe objects in the usual way.

Once, in the first phase, a suppression pattern has been determined to protect a particular unsafe object, we immediately check whether the elementary aggregations corresponding to this suppression pattern are safe or not. In case some elementary aggregations turn out to be unsafe, they are dynamically added to the set of unsafe objects. The LP problems corresponding to these unsafe elementary aggregations are constructed in the normal way described earlier.

Note that the suppression pattern corresponding to a dynamically created unsafe elementary aggregation may itself also contain an unsafe elementary aggregation. Such an unsafe elementary aggregation is also added to the set of unsafe objects that require protection. This process goes on all suppression patterns corresponding to all unsafe objects do not contain any unsafe elementary aggregation anymore.

In the second, clean-up phase, all unsafe objects, i.e. the original unsafe objects plus the dynamically created ones, are considered. All these objects have to be sufficiently protected. For each unsafe object we again check whether the corresponding suppression pattern determined during the second phase is safe. If not, we add the corresponding unsafe elementary aggregations to the set of unsafe objects. The difference between the first and second phase is that during the first phase all cell values are considered, whereas during the second phase only the cells eligible for suppression determined during the first phase are considered.

20.8. Discussion

In this chapter we showed that by using an approach where unsafe elementary aggregations are dynamically added to the set of unsafe objects one can, in principle, guarantee that a completely safe cell suppression pattern is determined. Each time a suppression pattern corresponding to an unsafe object has been determined, it is checked whether this suppression pattern contains any unsafe elementary aggregations. If so, constraints are added to prevent the occurrence of these aggregations. Subsequently, suppression patterns are determined to protect the dynamically created unsafe elementary aggregations. This process goes on until no unsafe aggregations occur in a suppression pattern corresponding to an unsafe object any more. The determined table can be published without fear for disclosure. At present it is, however, not yet known whether determination of unsafe elementary aggregations can be carried out sufficiently fast for the above approach to be applicable in practice.

At the US Bureau of the Census an alternative approach is followed to obtain safe suppression patterns (see Jewett, 1993). The idea of this approach is to determine to what extent a cell can protect a sensitive cell before a suppression pattern is determined. The extent to which a cell can protect a certain sensitive cell is called the protection capacity of

the former cell. If protection capacities are calculated correctly, one can ensure in this way that the final suppression pattern is completely safe.

Using protection capacities has the drawback that this may lead to a bit too much information loss, because these capacities are determined so that even in the worst case a completely safe suppression pattern results.

A more important, practical problem is that calculating protection capacities is no trivial exercise. For tables with a simple underlying structure it is not too hard to calculate these capacities. For tables with a more complicated structure, for example hierarchical tables and tables with respondents that contribute to multiple cells, calculating protection intervals becomes difficult. The technical aspects of calculating protection capacities in general are not (well) documented. Jewett (1993) does not make an attempt to describe how he calculates these protection intervals in general. Moreover, he notes that he may be the only person who completely understands how the calculation is done. Jewett refers persons who are interested in how the calculation is performed to his source code.

We have used an LP approach in this chapter to illustrate how completely safe cell suppression software can be constructed. Instead of an LP approach other approaches are possible as well. In particular, one could use a mixed integer programming (MIP) approach. In comparison to the LP approach the MIP approach has the advantage it may lead to better, i.e. closer to optimal, solutions. The LP approach does not lead to optimal solutions to even the simplified cell suppression problem, because all sensitive cells and aggregations are treated separately rather than simultaneously. In the MIP approach all sensitive cells and aggregations can, in principle, be treated simultaneously.

Another advantage of using an MIP approach over using an LP approach is that the class of possible objective functions is larger for the MIP approach. For the LP approach the class of objective functions for the two phases is limited to linear expressions in the ΔV_i. For the MIP approach one may also assign fixed weights to each cell.

Unfortunately, the MIP approach also has several disadvantages. First, the complexity, and hence the computing time, of MIP algorithms depends in an exponential manner on their input data (number of unknowns involved, and number of constraints). The theoretically best LP algorithms on the other hand have a computing time that only depend in a polynomial manner on the input data. So, from a purely theoretical point of view LP algorithms are much faster than MIP algorithms.

Second, practical experience has confirmed that the computing time of MIP algorithms really is clearly larger than the computing time of LP algorithms for similar problems. In practice, to apply MIP algorithms successfully they are usually designed especially for certain problems. The algorithms then extensively exploit the structure of the problem. The structure of the cell suppression problem as sketched in this chapter might be too "fuzzy" to apply MIP techniques with success.

Third, MIP algorithms and software are more complicated than LP algorithms and software. Cell suppression software based on LP algorithms may be developed and maintained by a statistical office itself. For cell suppression software based on MIP algorithms the statistical office is dependent on external help from experts on operations research.

Fourth, cell suppression software based on the LP approach has been in practical use for over 20 years now. Cell suppression software based on the MIP approach has not yet, or hardly, been in practical use.

Instead of an MIP or LP approach, one could also opt for simpler methods for solving the cell suppression problem. One such approach is the "hypercube method" developed by Repsilber (see for example Repsilber, 1993; De Waal, 1994b; Willenborg and De Waal, 1996 and 2001). This approach has been implemented in the software package GHQUAR. The advantage of this approach is that it is very fast, and hence that it can protect very large tables in an acceptable amount of time.

A drawback of the GHQUAR approach is that the found solutions are less close to optimal than the solutions found by an LP approach, let alone an MIP approach. For a comparison of the results of the three approaches and an approach based on a network formulation (see Cox, 1993 and 1995b) we refer to Giessing (1998, 1999, and 2000). Giessing concludes that for her test tables the LP approach performs best overall. Unfortunately, the implementation of the MIP approach (τ -ARGUS version 2.0) could not be applied to the test tables because it could not handle tables with a hierarchical structure. A comparison of the LP approach and the MIP approach is therefore not available.

In general, a drawback of "simple" approaches is that they may be too simple to handle all, possibly unforeseen, problems. At the moment it is not clear to us whether the approach of GHQUAR is powerful enough to solve the cell suppression problem for hierarchical and linked tables without losing too much information.

All in all it is not clear whether an MIP approach, an LP approach, or a simpler approach would be preferable to solve the cell suppression problem. We personally prefer an LP approach, possibly in combination with the "hypercube method" to protect very large tables, but others are, of course, free to disagree.

Appendix A

The numbers in columns 1 and 2 are the indices of the weights in Table 19.4. The number in the third column is the ratio of the two weights in column 1 and column 2. Only ratios less than one that occur more than once are listed.

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
2	36	0.91751	5	33	0.93973	11	36	0.95152
1	34	0.91752	3	29	0.93973	9	34	0.95152
			2	24	0.93973			
1	33	0.91856	1	20	0.93973	9	33	0.95260
2	35	0.91856	7	35	0.93973	11	35	0.95260
			4	31	0.93973			
2	32	0.92484				4	26	0.95269
1	30	0.92484	6	33	0.93987	3	22	0.95269
			8	35	0.93987			
1	29	0.92590				12	36	0.95283
2	31	0.92590	2	23	0.94254	10	34	0.95283
			1	19	0.94254			
4	36	0.93122				4	25	0.95369
3	34	0.93122	4	30	0.94438	2	17	0.95369
			2	22	0.94438	5	27	0.95369
1	27	0.93220	7	34	0.94438	3	21	0.95369
2	28	0.93220				7	28	0.95369
			4	29	0.94546	1	14	0.95369
3	33	0.93228	7	33	0.94546			
4	35	0.93228	2	20	0.94546	4	24	0.95377
						3	20	0.95377
3	32	0.93297	3	27	0.94613			
5	36	0.93297	4	28	0.94613	6	27	0.95383
1	26	0.93297				8	28	0.95383
			7	32	0.94616			
1	24	0.93404	5	30	0.94616	10	33	0.95391
5	35	0.93404				12	35	0.95391
3	31	0.93404	6	30	0.94630			
			8	32	0.94630	10	32	0.95462
4	32	0.93866				13	36	0.95462
7	36	0.93866	5	29	0.94723	6	26	0.95462
2	26	0.93866	7	31	0.94724			

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
1	22	0.93866				2	16	0.95531
3	30	0.93866	6	29	0.94737	1	13	0.95531
5	34	0.93866	8	31	0.94738			
						6	24	0.95570
8	36	0.93880	1	17	0.94791	13	35	0.95570
6	34	0.93880	3	25	0.94791	10	31	0.95570
			5	28	0.94791			
2	25	0.93965				4	23	0.95662
1	21	0.93965	2	18	0.95007	3	19	0.95662
			1	15	0.95007			
5	23	0.95842	8	23	0.96441	9	23	0.97155
3	18	0.95842	6	19	0.96441	14	28	0.97155
1	11	0.95842				6	16	0.97155
			15	34	0.96573	22	36	0.97155
9	30	0.95912	18	36	0.96573	1	7	0.97155
11	32	0.95912				20	35	0.97155
			12	30	0.96629			
7	27	0.95950	8	22	0.96629	7	18	0.97197
4	21	0.95950	16	34	0.96629	5	15	0.97197
2	14	0.95950						
			9	27	0.96675	8	18	0.97211
9	29	0.96021	11	28	0.96675	6	15	0.97211
11	31	0.96021						
			12	29	0.96739	9	22	0.97345
5	22	0.96029	16	33	0.96739	15	30	0.97345
7	26	0.96029	8	20	0.96739	11	26	0.97345
						18	32	0.97345
12	32	0.96044	15	32	0.96755	23	36	0.97345
10	30	0.96044	9	26	0.96755	19	34	0.97345
6	22	0.96044	19	36	0.96755			
16	36	0.96044				11	25	0.97447
8	26	0.96044	17	34	0.96794	9	21	0.97447
13	34	0.96044	4	17	0.96794			
						15	29	0.97456
7	25	0.96130	10	27	0.96808	9	20	0.97456
5	21	0.96130	12	28	0.96808	11	24	0.97456
						19	33	0.97456
5	20	0.96139	16	32	0.96811	23	35	0.97456
7	24	0.96139	13	30	0.96811	18	31	0.97456

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
8	25	0.96145	9	24	0.96865	12	26	0.97479
6	21	0.96145	15	31	0.96865	10	22	0.97479
			19	35	0.96865			
10	29	0.96153				7	17	0.97567
13	33	0.96153	13	29	0.96921	5	14	0.97567
6	20	0.96153	16	31	0.96921			
8	24	0.96153				12	25	0.97581
16	35	0.96153	4	16	0.96958	13	27	0.97581
12	31	0.96153	3	13	0.96958	8	17	0.97581
						10	21	0.97581
14	34	0.96207	14	30	0.96975	16	28	0.97581
17	36	0.96207	17	32	0.96975	6	14	0.97581
1	10	0.96294	6	17	0.96990	12	24	0.97590
2	12	0.96294	10	25	0.96990	10	20	0.97590
			13	28	0.96990			
4	18	0.96426				24	36	0.97636
7	23	0.96426	7	19	0.97014	20	34	0.97636
5	19	0.96426	4	15	0.97014			
2	11	0.96427	2	9	0.97014	21	34	0.97645
1	9	0.96427				25	36	0.97645
3	15	0.96427	1	8	0.97141			
			5	16	0.97141			
			3	12	0.97141			
2	8	0.97733	11	20	0.98050	11	18	0.98528
4	12	0.97733	18	29	0.98050	14	21	0.98528
7	16	0.97733	23	33	0.98050	26	32	0.98528
5	13	0.97733				20	29	0.98528
1	6	0.97733	13	23	0.98066	1	3	0.98528
3	10	0.97733	10	18	0.98066	9	15	0.98528
			6	11	0.98066	22	30	0.98528
14	27	0.97747				6	10	0.98528
11	23	0.97747	19	30	0.98122	17	25	0.98528
9	19	0.97747	23	32	0.98122	2	4	0.98528
20	33	0.97747				8	12	0.98528
26	36	0.97747	16	27	0.98176	24	31	0.98528
2	7	0.97747	12	21	0.98176			
6	13	0.97747	8	14	0.98176	14	20	0.98536
1	5	0.97747				21	29	0.98536
8	16	0.97747	13	22	0.98257	28	35	0.98536

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
22	34	0.97747	16	26	0.98257	27	33	0.98536
24	35	0.97747				17	24	0.98536
17	28	0.97747	9	17	0.98304	25	31	0.98537
			15	25	0.98304			
21	33	0.97756				15	23	0.98607
25	35	0.97756	4	10	0.98329	10	16	0.98607
			2	6	0.98329	30	36	0.98607
14	26	0.97828	7	13	0.98329	3	7	0.98607
21	32	0.97828				21	28	0.98607
27	36	0.97828	11	19	0.98343	29	35	0.98607
			24	33	0.98343			
22	33	0.97859	2	5	0.98343	22	29	0.98640
26	35	0.97859	17	27	0.98343	26	31	0.98640
			8	13	0.98343			
4	11	0.97867	26	34	0.98343	7	11	0.98649
3	9	0.97867				5	9	0.98649
			16	25	0.98360			
12	23	0.97882	13	21	0.98360	16	23	0.98663
10	19	0.97882				12	18	0.98663
			13	20	0.98369	13	19	0.98663
14	25	0.97931	16	24	0.98369	8	11	0.98663
9	18	0.97931				10	15	0.98663
22	32	0.97931	24	32	0.98416	6	9	0.98663
6	12	0.97931	20	30	0.98416			
1	4	0.97931				15	22	0.98799
20	31	0.97931	14	22	0.98424	18	26	0.98799
			28	36	0.98424			
11	22	0.97938	21	30	0.98424	14	19	0.98831
18	30	0.97938	25	32	0.98424	17	23	0.98831
23	34	0.97938	17	26	0.98424			
			27	34	0.98424	11	17	0.98903
14	24	0.97939				15	21	0.98903
21	31	0.97939	5	10	0.98513	19	27	0.98903
27	35	0.97939	7	12	0.98513	18	25	0.98903
						9	14	0.98903
						23	28	0.98903
15	20	0.98912	29	32	0.99281	19	20	0.99702
18	24	0.98912	20	26	0.99281	23	24	0.99702
			33	36	0.99281			
28	34	0.99024				31	33	0.99812
25	30	0.99024	22	27	0.99312	4	5	0.99812

Appendix A

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
17	22	0.99024	26	28	0.99312	12	13	0.99812
						25	27	0.99812
12	17	0.99039	30	33	0.99321	18	19	0.99812
10	14	0.99039	32	35	0.99321	32	34	0.99812
9	13	0.99071	27	29	0.99323	16	17	0.99830
11	16	0.99071	28	31	0.99323	13	14	0.99830
24	29	0.99128	14	17	0.99394	9	10	0.99863
2	3	0.99128	15	18	0.99394	11	12	0.99863
26	30	0.99128	21	25	0.99394			
8	10	0.99129	34	36	0.99394	31	32	0.99886
11	15	0.99129	19	23	0.99394	35	36	0.99886
17	21	0.99129	9	11	0.99394	24	26	0.99886
			6	8	0.99394	20	22	0.99886
28	33	0.99137	30	32	0.99394	29	30	0.99886
25	29	0.99137	13	16	0.99394	33	34	0.99886
17	20	0.99137	5	7	0.99394			
			22	26	0.99394	21	22	0.99895
4	8	0.99193	20	24	0.99394	25	26	0.99895
3	6	0.99193	1	2	0.99394			
			10	12	0.99394	7	8	0.99985
20	27	0.99199	3	4	0.99394	5	6	0.99985
24	28	0.99199	33	35	0.99394			
			27	28	0.99394	24	25	0.99991
15	19	0.99208	29	31	0.99394	20	21	0.99991
21	27	0.99208						
31	35	0.99208	16	18	0.99451	23	25	0.99693
4	7	0.99208	13	15	0.99451	19	21	0.99693
18	23	0.99208						
12	16	0.99208	23	27	0.99506			
10	13	0.99208	11	14	0.99506			
29	33	0.99208	18	21	0.99506			
3	5	0.99208						
32	36	0.99208	22	24	0.99507			
30	34	0.99208	34	35	0.99507			
25	28	0.99208	30	31	0.99507			
28	32	0.99210	19	22	0.99588			
27	30	0.99210	23	26	0.99588			

Nr.	Nr.	Ratio	Nr.	Nr.	Ratio	Nr.	Nr.	Ratio
16	19	0.99265	14	15	0.99621			
12	15	0.99265	17	18	0.99621			
8	9	0.99265						

In the above ratios list two ratios are considered equal when they differ less than 2×10^{-6}. This can be justified as follows. A weight W_i in Table 19.4 is given by $W_i = W_i' + \varepsilon_i$, where $|\varepsilon_i| < 0.5\times10^{-4}$ and W_i' is the true sampling weight. A ratio W_1/W_2 between two weights from Table 19.4 is therefore given by $W_1/W_2 = W_1'/W_2' + \delta$, where $\delta \approx (W_1\varepsilon_1 - W_2\varepsilon_2)/W_2^2$. Taking $W_1 \approx W_2 \approx 100$, $|\delta|$ is less than approximately 10^{-6}. Two ratios that should actually be equal therefore differ less than 2×10^{-6} in this ratios list.

Appendix B

The numbers in columns 1 and 2 are the indices of the weights in Table 19.7. In the third column the difference of the weights in columns 1 and 2 is listed. Only differences less than 0 that occur 8 or 16 times are listed.

Nr.	Nr.	Difference	Nr.	Nr.	Difference	Nr.	Nr.	Difference
2	26	-102.2163	11	19	-26.6158	2	4	-12.2761
3	27	-102.2163	3	10	-26.6157	6	10	-12.2761
4	28	-102.2163	14	22	-26.6157	9	12	-12.2761
6	30	-102.2163	25	30	-26.6157	14	18	-12.2761
7	31	-102.2163	27	32	-26.6157	15	19	-12.2761
10	32	-102.2163	1	6	-26.6157	25	27	-12.2761
1	25	-102.2163	8	15	-26.6157	26	28	-12.2761
5	29	-102.2163	18	24	-26.6157	29	31	-12.2761
						30	32	-12.2761
11	27	-65.2540	11	18	-26.2925	1	3	-12.2761
9	26	-65.2539	9	16	-26.2924	5	7	-12.2761
12	28	-65.2539	15	22	-26.2924	13	17	-12.2761
15	30	-65.2539	8	14	-26.2924	16	20	-12.2761
17	31	-65.2539	12	20	-26.2924	21	23	-12.2761
19	32	-65.2539	13	21	-26.2924	22	24	-12.2761
8	25	-65.2539	17	23	-26.2924	8	11	-12.2760
13	29	-65.2539	19	24	-26.2924			
						7	10	-6.5623
2	16	-63.2548	2	6	-25.9639	17	19	-6.5623
3	18	-63.2548	4	10	-25.9639	21	22	-6.5623
6	22	-63.2548	9	15	-25.9639	23	24	-6.5623
1	14	-63.2548	12	19	-25.9639	29	30	-6.5623
4	20	-63.2548	16	22	-25.9639	31	32	-6.5623

Nr.	Nr.	Difference	Nr.	Nr.	Difference	Nr.	Nr.	Difference
5	21	-63.2548	20	24	-25.9639	5	6	-6.5623
7	23	-63.2548	26	30	-25.9639	13	15	-6.5623
10	24	-63.2548	28	32	-25.9639			
						11	12	-0.6519
20	28	-38.9615	11	17	-20.0535	3	4	-0.6518
23	31	-38.9615	1	5	-20.0534	14	16	-0.6518
24	32	-38.9615	3	7	-20.0534	25	26	-0.6518
14	25	-38.9615	8	13	-20.0534	27	28	-0.6518
16	26	-38.9615	14	21	-20.0534	1	2	-0.6518
18	27	-38.9615	25	29	-20.0534	8	9	-0.6518
21	29	-38.9615	27	31	-20.0534	18	20	-0.6518
22	30	-38.9615	18	23	-20.0534			
1	8	-36.9624	2	5	-19.4016			
2	9	-36.9624	9	13	-19.4016			
4	12	-36.9624	4	7	-19.4016			
5	13	-36.9624	12	17	-19.4016			
6	15	-36.9624	16	21	-19.4016			
7	17	-36.9624	20	23	-19.4016			
10	19	-36.9624	26	29	-19.4016			
3	11	-36.9623	28	31	-19.4016			

In the above differences list two differences are considered equal when they differ less than 2×10^{-4}. This can be justified as follows. A weight W_i in Table 19.7 is given by $W_i = W_i' + \varepsilon_i$, where $|\varepsilon_i| < 0.5\times10^{-4}$ and W_i' is the true sampling weight. A difference between two weights from Table 19.7 is therefore given by $\Delta W = \Delta W' + \delta$, where $|\delta| < 10^{-4}$. Two differences that should actually be equal therefore differ less than 2×10^{-4} in this differences list.

Appendix C

Group 1:

Nr.	Weight ×Frequency
1	89,518.82
2	83,049.42
3	91,642.09
4	80,360.33
5	99,064.28
6	100,705.86
7	93,587.27
10	97,137.95
Total	735,066.02

Group 2:

Nr.	Weight × Frequency
25	91,053.95
26	100,429.51
27	85,852.36
28	96,382.13
29	82,271.08
30	96,580.09
31	98,167.08
32	104,138.79
Total	754,874.99

Group 3:

Nr.	Weight × Frequency
8	94,036.66
9	90,219.48
11	107,008.15
12	100,202.55
13	100,929.36
15	92,634.12
17	87,154.83
19	102,850.87
Total	775,036.02

Group 4:

Nr.	Weight × Frequency
14	87,625.34
16	96,650.25
18	88,705.64
20	87,754.33
21	91,421.63
22	88,945.17
23	92,526.48
24	101,394.16
Total	735,023.00

Group 5:

Nr.	Weight × Frequency
1	89,518.82
3	91,642.09
8	94,036.66
11	107,008.15
14	87,625.34
18	88,705.64
25	91,053.95
27	85,852.36
Total	735,443.01

Group 6:

Nr.	Weight × Frequency
6	100,705.86
10	97,137.95
15	92,634.12
19	102,850.87
22	88,945.17
24	101,394.16
30	96,580.09
32	104,138.79
Total	784,387.01

Group 7:

Nr.	Weight × Frequency
2	83,049.42
4	80,360.33
9	90,219.48
12	100,202.55
16	96,650.25
20	87,754.33
26	100,429.51
28	96,382.13
Total	735,048.00

Group 8:

Nr.	Weight × Frequency
5	99,064.28
7	93,587.27
13	100,929.36
17	87,154.83
21	91,421.63
23	92,526.48
29	82,271.08
31	98,167.08
Total	745,122.01

Appendix C

Group 9:

Nr.	Weight × Frequency
1	89,518.82
2	83,049.42
5	99,064.28
6	100,705.86
8	94,036.66
9	90,219.48
13	100,929.36
14	87,625.34
15	92,634.12
16	96,650.25
21	91,421.63
22	88,945.17
25	91,053.95
26	100,429.51
29	82,271.08
30	96,580.09
Total	1,485,135.02

Group 10:

Nr.	Weight × Frequency
3	91,642.09
4	80,360.33
7	93,587.27
10	97,137.95
11	107,008.15
12	100,202.55
17	87,154.83
18	88,705.64
19	102,850.87
20	87,754.33
23	92,526.48
24	101,394.16
27	85,852.36
28	96,382.13
31	98,167.08
32	104,138.79
Total	1,514,865.01

Samenvatting (Summary in Dutch)

Het statistische proces zoals dat op nationale statistische bureaus wordt gepraktiseerd kan worden onderverdeeld in een groot aantal stappen. Bijvoorbeeld, Willeboordse (1998) onderscheidt de volgende fases in het statistische proces voor bedrijfsstatistieken (voor sociale statistieken kan dezelfde onderverdeling worden gemaakt):

- vaststellen van onderzoeksdoelen;
- ontwerpen van vragenformulier en steekproef;
- verzamelen en invoeren van data;
- verwerken en analyseren van data;
- publiceren van data.

Ieder van deze fases kan zelf worden onderverdeeld in een aantal stappen. Bij het vaststellen van de onderzoeksdoelen worden gebruikersgroepen onderscheiden, gebruikerswensen geïnventariseerd, beschikbare gegevensbronnen in kaart gebracht, potentiële respondenten benaderd, het onderzoek ingebed in het algemene raamwerk voor bedrijfsstatistieken, de doelpopulatie en doelvariabelen vastgesteld, en wordt de outputtabel ontworpen.

Bij het ontwerpen van het vragenformulier en de steekproef wordt het potentiële nut van beschikbare administratieve registers onderzocht, de kaderpopulatie in het zogeheten Algemeen Bedrijfsregister vergeleken met de doelpopulatie, het steekproefkader gedefinieerd, het steekproefontwerp en de schattingsmethode geselecteerd, en het vragenformulier ontworpen.

Tijdens het verzamelen en invoeren van data wordt de steekproef getrokken, en worden data verzameld en ingevoerd in het computersysteem op het statistische bureau. Het statistische bureau tracht tijdens deze stap de responslast voor bedrijven zoveel mogelijk te beperken en de non-respons te minimaliseren. Om data te verzamelen worden verschillende mogelijkheden, zoals dataverzameling via papieren vragenlijsten, persoonlijke interviews, telefonische interviews, en Electronic Data Interchange, in overweging genomen.

Tijdens het verwerken en analyseren van data worden foutieve data opgespoord en gecorrigeerd (gaafgemaakt), ontbrekende gegevens geïmputeerd (dat wil zeggen geschat en ingevuld), ophooggewichten bepaald, populatiecijfers geschat, de data in het integratieraamwerk geïntegreerd, en de data geanalyseerd (bijvoorbeeld om te corrigeren voor seizoensinvloeden).

Bij het publiceren wordt de publicatiestrategie bepaald, worden de data beveiligd tegen onthulling van gevoelige informatie, en ten slotte worden de uiteindelijke data gepubliceerd.

In dit boek onderzoeken we twee verschillende, maar gerelateerde, bovengenoemde onderwerpen. Het eerste onderwerp is gaafmaken (Engels: statistical data editing), ook wel controle en correctie genaamd, dat plaatsvindt tijdens het verwerken en analyseren van data. Het doel van gaafmaken is het opsporen en corrigeren van incorrecte data. Om dit doel te bereiken worden de geobserveerde data verrijkt door middel van vakinhoudelijke kennis en statistische analyses. We proberen in feite meer informatie te creëren dan we hebben geobserveerd.

Het tweede onderwerp is statistische beveiliging, dat plaatsvindt aan het eind van het statistische proces. Het doel van statistische beveiliging is het verhinderen dat gevoelige informatie over individuele respondenten, of kleine groepen respondenten, uit de gepubliceerde data kan worden afgeleid. Om dit doel te bereiken worden vaak gegevens verwijderd, of wordt de informatie in de data gereduceerd door het toevoegen van ruis of het indikken (hercoderen) van variabelen. We proberen hier dus in feite de informatie in de data te verminderen.

Op het eerste gezicht lijken gaafmaken en statistische beveiliging elkaars tegengestelde. Bij nadere beschouwing blijken de onderwerpen echter sterk gerelateerd te zijn. Bij beide onderwerpen trachten we zoveel mogelijk informatie te behouden. Het voornaamste verschil tussen de twee onderwerpen wordt gevormd door de randvoorwaarden waaraan voldaan moeten worden. Bij gaafmaken worden foutieve data meestal opgespoord met behulp van bepaalde regels, edits genaamd. Vaak wordt verondersteld dat er zo min mogelijk fouten in de geobserveerde data aanwezig zijn. Gegeven deze veronderstelling ligt het voor de hand zo min mogelijk waarden aan te passen zodanig dat aan alle edits wordt voldaan. Bij statistische beveiliging veronderstellen we vaak dat veilige microdata aan bepaalde frequentieregels moeten voldoen. We proberen dan zo min mogelijk geobserveerde data te verwijderen zodanig dat aan de frequentieregels wordt voldaan. In beide gevallen kunnen we het resulterende wiskundige probleem formuleren als een optimalisatieprobleem. Bij statistische beveiliging kan het optimalisatieprobleem worden geformuleerd als een zogeheten set-covering probleem, bij gaafmaken kan het optimalisatieprobleem deels worden opgelost door middel van algoritmen voor het set-covering probleem.

We hebben dit boek als volgt opgebouwd. Hoofdstuk 1 geeft een korte inleiding op de twee onderwerpen van dit boek. In ditzelfde hoofdstuk geven we tevens een overzicht van het proefschrift.

In hoofdstuk 2 gaan we in op het gaafmaken in het algemeen. We beschrijven een aantal bekende technieken om het gaafmaakproces efficiënt te laten verlopen. In het bijzonder gaan we kort in op automatisch gaafmaken, de gaafmaaktechniek waarop we ons in latere hoofdstukken richten.

In die latere hoofdstukken concentreren we ons vooral op het zogeheten foutenlokalisatieprobleem, dat wil zeggen het probleem van het opsporen van incorrecte data. Gezien de complexiteit van dit probleem wordt het in de literatuur hetzij uitsluitend voor categoriale (discrete) data zonder rekenkundige structuur, zoals “Geslacht” of “Beroep”, of uitsluitend voor continue data beschouwd. In dit boek beschouwen we het foutenlokalisatieprobleem echter voor een mix van categoriale en continue data.

In hoofdstuk 3 geven we een gedetailleerde wiskundige formulering van het foutenlokalisatieprobleem voor een mix van categoriale en continue data. Deze formulering is gebaseerd op eenvoudige concepten uit de wiskundige logica. In het bijzonder maakt de formulering gebruik van de implicatieoperator (IF/THEN-statement). Wij zijn de eersten die een dergelijke formulering voor het foutenlokalisatieprobleem, die natuurlijker lijkt dan een formulering als geheeltallig programmeringsprobleem, geven. In hetzelfde hoofdstuk formuleren we het foutenlokalisatieprobleem ook als een geheeltallig programmeringsprobleem. In principe kan dit optimalisatieprobleem worden opgelost door gebruik te maken van standaard branch-and-bound methoden zoals deze zijn geïmplementeerd in commercieel verkrijgbare software. Een nadeel van dergelijke commerciële software is echter dat slechts één optimale oplossing voor het wiskundige optimalisatieprobleem wordt bepaald, terwijl wij graag alle optimale oplossingen willen bepalen om daarvan later één, op grond van een additioneel criterium, uit te kiezen. Om alle optimale oplossingen met behulp van commerciële software te bepalen zijn speciale maatregelen noodzakelijk. Het is nog onduidelijk of commerciële software met dergelijke speciale maatregelen voldoende snel is voor praktische gevallen van het foutenlokalisatieprobleem.

Hoofdstukken 4 tot en met 10 beschrijven verschillende methoden om het wiskundige foutenlokalisatieprobleem op te lossen. Hoofdstuk 4 beschrijft een methode ontwikkeld door Fellegi en Holt (1976) gebaseerd op het genereren van zogeheten impliciete edits. Dit hoofdstuk is grotendeels gebaseerd op uit de literatuur bekende resultaten, en bevat nauwelijks nieuwe resultaten. Uitzonderingen zijn een eenvoudig bewijs dat de methode van Fellegi en Holt ook werkt voor continue data, en de opmerking dat de methode, in principe, ook gebruikt kan worden voor een mix van categoriale en continue data.

Hoofdstuk 5 beschrijft hoe methoden, in het bijzonder het algoritme van Chernikova (Chernikova, 1964 en 1965) voor het genereren van hoekpunten van polyhedra gebruikt kunnen worden voor het oplossen van het foutenlokalisatieprobleem. Dergelijke methoden worden in praktijk toegepast voor het oplossen van het foutenlokalisatieprobleem voor continue data. In de literatuur wordt de mogelijkheid om deze methoden uit te breiden tot een mix van categoriale en continue data kort aangestipt (zie Sande, 1978a). In dat rapport worden echter geen details over de uitgebreide methode verschaft. In hoofdstuk 5 worden dergelijke details wel beschreven, en wordt gedemonstreerd dat een aantal uit de literatuur bekende resultaten voor continue data ook van toepassing is op een mix van categoriale en continue data. Ten slotte merken we in dit hoofdstuk op dat naast Chernikova's algoritme ook andere algoritmen voor het genereren van hoekpunten van polyhedra gebruikt kunnen worden voor het oplossen van het foutenlokalisatieprobleem.

In hoofdstuk 6 wordt het foutenlokalisatieprobleem als een zogeheten dynamisch disjunctive-facet probleem beschreven. Glover, Klingman and Stutz (1974) beschrijven een snedemethode voor het oplossen van het disjunctive-facet probleem. In eerste instantie trachten we het foutenlokalisatieprobleem als een dergelijk disjunctive-facet probleem te formuleren. Dit blijkt helaas niet geheel mogelijk te zijn. Het gewone disjunctive-facet probleem breiden we daarom uit naar een *dynamisch* disjunctive-facet probleem. De snedemethode van Glover, Klingman and Stutz breiden we dienovereenkomstig uit. Bovendien laten we zien dat het algoritme van Glover, Klingman and Stutz voor het

oorspronkelijke disjunctive-facet probleem niet noodzakelijkerwijs na eindig veel stappen stopt. Ten slotte breiden we ons algoritme uit zodanig dat eindigheid gegarandeerd is.

Hoofdstuk 7 beschrijft een methode ontwikkeld door Pergamentsev (1998) gebaseerd op lokale zoektechnieken. Pergamentsev was een tweedefase student aan de Technische Universiteit Eindhoven (TUE), en liep stage bij het Centraal Bureau voor de Statistiek (CBS) waar hij aan het foutenlokalisatieprobleem werkte. In hoofdstuk 7 stellen wij een aantal mogelijke verbeteringen op Pergamentsev's methode voor.

Hoofdstuk 8 beschrijft een branch-and-bound methode. Quere, wederom een tweedefase student aan de TUE die tijdens een stage bij het CBS aan het foutenlokalisatieprobleem werkte, ontwikkelde deze branch-and-bound methode voor continue data. Later hebben we in samenwerking met Quere deze methode uitgebreid naar een mix van categoriale en continue data.

Het foutenlokalisatieprobleem breiden we in hoofdstuk 9 uit tot een mix van categoriale, continue en geheeltallige data. Daarnaast breiden we het algoritme van hoofdstuk 8 uit zodat het geschikt wordt voor het oplossen van dit nieuwe probleem. Het ontwikkelde algoritme maakt gebruik van Fourier-Motzkin eliminatie in geheeltallige data zoals ontwikkeld door Pugh (1992). Ons algoritme combineert Fourier-Motzkin eliminatie in geheeltallige data op efficiënte wijze met het algoritme van hoofdstuk 8.

In hoofdstuk 10 beginnen we met het beschrijven van snedemethoden ontwikkeld door Garfinkel, Kunnathur en Liepins (1986 en 1988). De oorspronkelijke algoritmen van Garfinkel, Kunnathur en Liepins zijn geschikt voor uitsluitend categoriale, respectievelijk uitsluitend continue data. In hoofdstuk 10 breiden we deze algoritmen uit tot een algoritme voor een mix van categoriale en continue data. Vervolgens gebruiken we de in hoofdstukken 8 en 9 ontwikkelde theorie om deze snedemethoden en een soortgelijke snedemethode van Ragsdale en McKeown (1996) te verbeteren, en verder uit te breiden naar een mix van categoriale, continue en geheeltallige data.

Resultaten van evaluatie-experimenten voor de algoritmen van hoofdstuk 3 (gebaseerd op een formulering als geheeltallig programmeringsprobleem), hoofdstuk 5 (gebaseerd op generatie van hoekpunten van polyhedra), hoofdstuk 8 (gebaseerd op branch-and-bound), en hoofdstuk 10 (gebaseerd op snedemethoden) op zes numerieke bestanden worden beschreven in hoofdstuk 11.

In hoofdstuk 12 beschouwen we imputatie, het invullen van ontbrekende en incorrecte data. In het bijzonder beschouwen we het probleem van consistente imputatie, dat wil zeggen het probleem van het zodanig invullen van ontbrekende en incorrecte gegevens dat de uiteindelijke data zowel statistisch acceptabel als intern consistent zijn. We stellen een algoritme, geïnspireerd door het algoritme ontwikkeld in hoofdstukken 8 en 9, voor om dit probleem op te lossen. Dit algoritme is geïmplementeerd en verder uitgewerkt door Kartika, een tweedefase student aan de Technische Universiteit Delft die tijdens een stage bij het CBS aan dit probleem werkte. In hetzelfde hoofdstuk beschrijven we ook WAID, een imputatieprogramma dat we in het kader van een Europees project hebben ontwikkeld.

Het gedeelte van dit boek over het gaafmaakproces wordt afgesloten door hoofdstuk 13. Dit hoofdstuk beschrijft een aantal praktische aspecten van het gaafmaakproces. De laatste

paragraaf van hoofdstuk 13 beschrijft kort de invloed van de ontwikkelde methodologie en software op de dagelijkse gang van zaken op het CBS.

Het foutenlokalisatieprobleem dat we in dit boek uitgebreid onderzoeken is gerelateerd aan veel andere problemen, zowel wiskundige als niet-wiskundige. Voorbeelden van gerelateerde wiskundige problemen zijn: het imputatieprobleem (het probleem van het schatten van ontbrekende en incorrecte data), het uitbijterprobleem (het probleem van het opsporen van univariate of multivariate uitbijters in de data), en het ophoogprobleem (het probleem van het bepalen van ophooggewichten die nodig zijn voor het schatten van populatiegegevens). Voorbeelden van gerelateerde niet-wiskundige problemen zijn: het logistieke probleem van het omgaan met grote hoeveelheden gegevens, en het ICT probleem van het ontwikkelen van professionele software voor het gaafmaakproces als onderdeel van de softwaresuite voor het gehele statistische proces. Deze problemen worden niet of nauwelijks aangestipt in dit boek.

In het tweede gedeelte van dit boek, hoofdstukken 14 tot en met 20, concentreren we ons op statistische beveiliging (Engels: statistical disclosure control). Hoofdstuk 14 geeft een overzicht van het vakgebied. In dit overzicht gaan we vooral in op een algemene aanpak voor de statistische beveiliging van microdata, dat wil zeggen de data van individuele respondenten, van met name sociale statistieken die wordt toegepast op diverse statistische bureaus waaronder het CBS. Volgens deze aanpak moeten regels voor de statistische beveiliging van microdata van sociale statistieken voornamelijk gebaseerd zijn op het eisen van drempelwaarden voor populatiefrequenties van bepaalde karakteristieken. Deze algemene aanpak wordt in aansluitende hoofdstukken nader uitgewerkt.

Hoofdstuk 15 beschrijft de – vanuit het oogpunt van de statistische beveiliging – ergst mogelijke populaties. Zulke populaties leiden tot het hoogst mogelijke verwachte aantal unieke individuen in willekeurig gekozen steekproeven. Wij hebben dit wiskundige probleem voor het eerst voorgesteld en opgelost.

De populatiefrequenties van karakteristieken zijn meestal onbekend. Om deze frequenties voor grote gebieden te schatten kunnen standaard statistische methoden worden gebruikt. Voor kleine gebieden moet men echter gebruikmaken van zogeheten small area estimation. Voorbeelden van small area schatters zijn synthetische en gecombineerde schatters. In hoofdstuk 16 gebruiken we deze synthetische en gecombineerde schatters om populatiefrequenties voor kleine gebieden te schatten. Deze schatters zijn in samenwerking met Pannekoek ontwikkeld en getest.

Indien de populatiefrequentie van een bepaalde karakteristiek beneden de vereiste drempelwaarde ligt, wordt deze karakteristiek te onveilig voor publicatie geacht. De karakteristiek mag daarom niet op deze wijze worden gepubliceerd. In zo'n geval moeten maatregelen getroffen worden om individuele eenheden met deze karakteristiek te beveiligen. Een mogelijke maatregel is onderdrukking, waarbij (een gedeelte van) de karakteristiek wordt verwijderd. Aangezien een statistisch bureau zoveel mogelijk informatie als wettelijk en moreel gezien is toegestaan wil publiceren, proberen we, bijvoorbeeld, zo min mogelijk waarden te onderdrukken onder de randvoorwaarde dat het resulterende databestand veilig is. In samenwerking met Willenborg hebben we optimalisatiemodellen voor een aantal van dergelijke problemen opgesteld. De wiskundige formuleringen van deze optimalisatiemodellen worden beschreven in hoofdstuk 17.

Naast onderdrukking kan men gebruikmaken van andere statistische beveiligingstechnieken om het onthullingsrisico te beperken. Voorbeelden van dergelijke technieken zijn hercodering en perturbatie. Al deze technieken, onderdrukking, hercodering en perturbatie, leiden tot informatieverlies. Aangezien we zoveel mogelijk informatie willen publiceren, onder de randvoorwaarde dat het resulterende databestand veilig is, willen we het informatieverlies ten gevolge van de ene techniek kunnen vergelijken met het informatieverlies ten gevolge van een andere techniek. In hoofdstuk 18 beschrijven we een algemeen raamwerk gebaseerd op het entropiebegrip om zulke vergelijkingen te kunnen maken. Ook dit raamwerk is in samenwerking met Willenborg ontwikkeld.

Het publiceren van ophooggewichten samen met het desbetreffende microdatabestand vereist enige extra aandacht. Deze ophooggewichten kunnen een potentiële onthuller mogelijkerwijs namelijk in staat stellen om meer informatie uit de gepubliceerde data af te leiden dan is toegestaan volgens de toegepaste statistische beveiligingsregels. In hoofdstuk 19 gaan we in op dit gevaar bij de publicatie van ophooggewichten door methoden te beschrijven volgens welke een potentiële onthuller te werk zou kunnen gaan. Tevens stellen we technieken voor om dit gevaar te bestrijden. Wederom hebben we deze onthullingsmethoden en beveiligingstechnieken in samenwerking met Willenborg ontwikkeld.

Ten slotte beschrijven we in hoofdstuk 20 het zogeheten celonderdrukkingsprobleem voor tabeldata. Dit hoofdstuk laat in het bijzonder zien dat de gewoonlijk gebruikte definitie voor een veilige tabel inconsistent is. In plaats van deze inconsistente definitie, stellen we een alternatieve definitie voor die gebaseerd is op zogeheten elementaire aggregaten. Het gebruik van elementaire aggregaten werd al aan het eind van de zeventiger jaren door Sande voorgesteld (zie, bijvoorbeeld, Sande, 1977). Diens concept van een veilige tabel is echter niet volledig consistent. Wij hebben Sande's concept van een veilige tabel aangepast zodanig dat onze definitie wel consistent is. We laten zien dat onze definitie niet kan leiden tot onderdrukte cellen waarvan de waarden nauwkeurig teruggerekend kunnen worden.

Hoofdstuk 20 is een mooi slothoofdstuk voor dit boek aangezien het teruggrijpt op concepten uit het eerste deel van dit boek, zoals impliciete edits en Fourier-Motzkin eliminatie. Elementaire aggregaten kunnen worden beschouwd als impliciete edits. Ze kunnen worden afgeleid door middel van Fourier-Motzkin eliminatie. Hoofdstuk 20 laat zien dat de studie van statistische beveiliging in zekere zin hetzelfde is als de studie van het automatisch opsporen van foutieve waarden.

References

Albert, J.H. and Gupta, A.K., 1983, Estimation in Contingency Tables Using Prior Information. *Journal of the Royal Statistical Society, Series (B)*, **45**, 60-69.

Bakker, T., 1997, Rounding in Tables. Report (research paper 9738), Statistics Netherlands, Voorburg.

Bankier, M., 1999, Experience with the New Imputation Methodology Used in the 1996 Canadian Census with Extensions for Future Censuses. *UN/ECE Work Session on Statistical Data Editing*, Rome.

Bankier, M., P. Poirier, M. Lachance and P. Mason, 2000, A Generic Implementation of the Nearest-Neighbour Imputation Methodology (NIM). *Proceedings of the Second International Conference on Establishment Surveys*, Buffalo, 571-578.

Barcaroli, G., C. Ceccarelli, O. Luzi, A. Manzari, E. Riccini and F. Silvestri, 1995, The Methodology of Editing and Imputation of Qualitative Variables Implemented in SCIA. Internal Report, ISTAT, Rome.

Barcaroli, G. and M. Venturi, 1996, The Probabilistic Approach to Automatic Edit and Imputation: Improvements of the Fellegi-Holt Methodology. *UN/ECE Work Session on Statistical Data Editing*, Voorburg.

Barcaroli, G. and M. Venturi, 1997, DAISY (Design, Analysis and Imputation System): Structure, Methodology and First Applications. *Statistical Data Editing (Volume 2)*; *Methods and Techniques*. United Nations, Geneva.

Ben-Ari, M., 2001, *Mathematical Logic for Computer Science* (second edition). Springer-Verlag, London.

Benedetti, R., and L. Franconi, 1998, An Estimation Method for Individual Risk of Disclosure Based on Sampling Design. Report, ISTAT, Rome.

Bethlehem, J.G. and W.J. Keller, 1987, Linear Weighting of Sample Survey Data. *Journal of Official Statistics*, **3**, 141-153.

Bethlehem, J.A., W.J. Keller and J. Pannekoek, 1989, Disclosure Control of Microdata. *Journal of the American Statistical Association,* **85**, 38-45.

Bishop, Y.M.M., Fienberg, S.E. and Holland, P.W., 1975, *Discrete Multivariate Analysis; Theory and Practice*. MIT Press, Cambridge, Massachusetts.

Blaise Reference Manual, 2002, Department of Statistical Informatics, Statistics Netherlands, Heerlen.

Blaise Developer's Guide, 2002, Department of Statistical Informatics, Statistics Netherlands, Heerlen.

Blien, U., H. Wirth and M. Müller, 1992, Disclosure Risk for Microdata Stemming from Official Statistics. *Statistica Neerlandica,* **46**, 69-82.

Bremner, D., K. Fukuda and A. Marzetta, 1998, Primal-Dual Methods for Vertex and Facet Enumeration. *Discrete and Computational Geometry*, **20**, 333-357.

Cabot, A.V., 1975, On the Generalized Lattice Point Problem. *Operations Research*, **23**, 565-571.

Carlin, B.P. and Louis, T.A., 1996, *Bayes and Emperical Bayes Methods for Data Analysis*. Chapman and Hall, New York.

Central Statistical Office, 2000, Editing and Calibration in Survey Processing. Report SMD-37, Ireland.

Chambers, R.L., T. Crespo, S. Laaksonen, P. Piela, P. Tsai and T. De Waal, 2001a, The AUTIMP-Project: Evaluation of WAID. Report (research paper 0121), Statistics Netherlands, Voorburg.

Chambers, R.L., J. Hoogland, S. Laaksonen, D.M. Mesa, J. Pannekoek, P. Piela, P. Tsai and T. De Waal, 2001b, The AUTIMP-Project: Evaluation of Imputation Software. Report (research paper 0122), Statistics Netherlands, Voorburg.

Chandru, V., 1993, Variable Elimination in Linear Constraints. *The Computer Journal*, **36**, 463-472.

Chandru, V. and J. N. Hooker, 1999, *Optimization Methods for Logical Inference*. John Wiley & Sons, New York.

Chaudhuri, A., 1994, Small Domain Statistics: A Review. *Statistica Neerlandica*, **48**, 215-236.

Chen, G. and S. Keller-McNully, 1998, Estimation of Identification Disclosure Risk in Microdata. *Journal of Official Statistics*, **14**, 79-95.

Chernikova, N.V., 1964, Algorithm for Finding a General Formula for the Non-Negative Solutions of a System of Linear Equations. *USSR Computational Mathematics and Mathematical Physics*, **4**, 151-158.

Chernikova, N.V., 1965, Algorithm for Finding a General Formula for the Non-Negative Solutions of a System of Linear Inequalities. *USSR Computational Mathematics and Mathematical Physics*, **5**, 228-233.

Chung, W.H., 2003, Efficient Automatic Error Localisation. (in Dutch), Internal report (BPA number: 1057-03-TMO), Statistics Netherlands, Voorburg.

Chvátal, V., 1983, *Linear Programming*. W.H. Freeman and Company, New York.

Citteur, C.A.W. and L.C.R.J. Willenborg, 1993, Public Use Microdata Files: Current Practices at National Statistical Bureaus. *Journal of Official Statistics*, **9**, 783-794.

Coccia, G., 1992, Disclosure Risk in Italian Current Population Surveys. International Seminar on Statistical Confidentiality, Dublin.

Cox, L.H., 1981, Linear Sensitivity Measures in Statistical Disclosure Control. *Journal of Statistical Planning and Inference*, **5**, 153-164.

References

Cox, L.H., 1986, Comment on Duncan and Lambert (1986). *Journal of the American Statistical Association*, **81**, 19-21.

Cox, L.H., 1987, A Constructive Procedure for Unbiased Controlled Rounding. *Journal of the American Statistical Association*, **82**, 520-524.

Cox, L.H., 1993, Solving Confidentiality Problems in Tabulations Using Network Optimization: A Network Model for Cell Suppression in the U.S. Economic Censuses. *Proceedings of the International Seminar on Statistical Confidentiality*, Dublin, 229-245.

Cox, L.H., 1995a, Protecting Confidentiality in Business Surveys. In: *Business Survey Methods* (ed. Cox, Binder, Chinnappa, Christianson & Kott), John Wiley & Sons, Inc., New York, 443-473.

Cox, L.H., 1995b, Network Models for Complementary Cell Suppression. *Journal of the American Statistical Association*, **90**, 1453-1462.

Cox, L.H. and L.R. Ernst, 1983, Controlled Rounding. In: *Some Recent Advances in the Theory of Computation and Application of Network Flow Models*, University of Toronto Press, 139-148.

Crescenzi, F., 1992, Estimating Population Uniques; Methodological Proposals and Applications on Italian Census Data. *Proceedings of the International Seminar on Statistical Confidentiality*, Dublin, 247-260.

Daalmans, J., 2000, Automatic Error Localisation of Categorical Data. Report (research paper 0024), Statistics Netherlands, Voorburg.

Daalmans, J., 2002, Deriving and Protecting Elementary Aggregations in the Cell Suppression Problem. Report (research paper 0203), Statistics Netherlands, Voorburg.

Dalenius, T.E., 1981, *Release of Microdata - The International Picture*. Selbstverlag GMD, Sankt Augustin.

Dandekar, R.A., M. Cohen, and N. Kirkendall, 2001, Applicability of Latin Hypercube Sampling Techniques to Create Multivariate Synthetic Microdata. *Proceedings of the NTTS&ETK 2001*, 839-847.

Dantzig, G.B. and B. Curtis Eaves, 1973, Fourier-Motzkin Elimination and Its Dual. *Journal of Combinatorial Theory (A)*, **14**, 288-297.

De Boer, P., B. Schouten and L. Willenborg, 2000, Remote Access: A Preliminary Study (in Dutch). Report (research paper 0041), Statistics Netherlands, Voorburg.

Defays, D. and P. Nanopoulos, 1993, Panels of Enterprises and Confidentiality: The Small Aggregates Method. *Proceedings of the 1992 Symposium on Design and Analysis of Longitudinal Surveys,* Ottawa, 195-204.

De Jong, A., 2002, Uni-Edit: Standardized Processing of Structural Business Statistics in the Netherlands. *UN/ECE Work Session on Statistical Data Editing*, Helsinki.

De Jong, W.A.M., 1992, ARGUS: An Integrated System for Data Protection. *Proceedings of the International Seminar on Statistical Confidentiality*, Dublin, 317-322.

De Jong, W.A.M., 1996, Designing a Complete Edit Strategy; Combining Techniques. Report (research paper 9639), Statistics Netherlands, Voorburg.

De Jonge, G., 1990, The Estimation of Population Unicity from Microdata Files (in Dutch), Internal report (BPA number: 10785-90-M1), Statistics Netherlands, Voorburg.

Dellaert, N.P. and W.A. Luijten, 1999, Statistical Disclosure in General Three-Dimensional Tables. *Statistics Neerlandica*, **53**, 197-221.

Deville, J.C. and C.E. Särndal, 1992, Calibration Estimators in Survey Sampling. *Journal of the American Statistical Association*, **87**, 376-382.

De Vries, R.E., 1993, Disclosure Control of Tabular Data Using Subtables. Report (BPA number: 11741-93-M1), Statistics Netherlands, Voorburg.

De Waal, T., 1994a, The Maximum Expected Number of Unique Individuals in a Population. *Kwantitatieve Methoden*, **15**, nr. 46, 15-33.

De Waal, T., 1994b, The Hypercube Method for Suppression in Tables. Report (BPA number: 9248-94-RSM), Statistics Netherlands, Voorburg.

De Waal, T., 1996, CherryPi: A Computer Program for Automatic Edit and Imputation. *UN/ECE Work Session on Statistical Data Editing*, Voorburg.

De Waal, T., 1997a, Cutting Plane Algorithms for Optimal Automatic Error Localisation. Report (research paper 9729), Statistics Netherlands, Voorburg.

De Waal, T., 1997b, Identifying Balance Restrictions. Report (research paper 9730), Statistics Netherlands, Voorburg.

De Waal, T., 1997c, The Error Localisation Problem as a Dynamic Disjunctive-Facet Problem. Report (research paper 9743), Statistics Netherlands, Voorburg.

De Waal, T., 1998a, Mathematical Programming Techniques for Solving the General Error Localisation Problem. Report (research paper 9809), Statistics Netherlands, Voorburg.

De Waal, T., 1998b, An Introduction to CherryPi and MacroView. Report (research paper 9836), Statistics Netherlands, Voorburg.

De Waal, T., 1998c, Improvements on Pergamentsev's Algorithm for Automatic Error Localisation in General Data. Report (research paper 9831), Statistics Netherlands, Voorburg.

De Waal, T., 2000a, An Optimality Proof of Statistics Netherlands' New Algorithm for Automatic Editing of Mixed Data. Report (research paper 0028), Statistics Netherlands, Voorburg.

References

De Waal, T., 2000b, Vertex Generation Methods for Solving the Error Localisation Problem in Mixed Data. Report (research paper 0036), Statistics Netherlands, Voorburg.

De Waal, T., 2000c, Using Subject-Matter Knowledge in a Fellegi-Holt Based Automatic Editing Program. Report (research paper 0037), Statistics Netherlands, Voorburg.

De Waal, T., 2000d, SLICE: A Software Framework for Statistical Data Editing and Imputation. In: *CAPTOR 2000: Qualità della Didattica e Sistemi Computer-Assisted*, 195-206.

De Waal, T., 2000e, A Brief Overview of Imputation Methods Applied at Statistics Netherlands. *Netherlands Official Statistics*, **15**, 23-27.

De Waal, T., 2001a, Automatic Edit and Imputation in Categorical, Continuous and Integer Data. Report (research paper 0107), Statistics Netherlands, Voorburg.

De Waal, T., 2001b, SLICE: Generalised Software for Statistical Data Editing. *Proceedings in Computational Statistics* (ed. J.G. Bethlehem and P.G.M. Van der Heijden), Physica-Verlag, New York, 277-282.

De Waal, T., 2001c, WAID 4.1: A Computer Program for Imputation of Missing Values. *Research in Official Statistics*, **4**, 53-70.

De Waal., T., 2002a, Algorithms for Automatic Error Localisation and Modification. Report (research paper 0226), Statistics Netherlands, Voorburg.

De Waal, T., 2002b, An Algorithm for Solving the Error Localisation Problem in General Data Based on Cutting Planes. Report (research paper 0232), Statistics Netherlands, Voorburg.

De Waal, T., 2003, Solving the Error Localization Problem by Means of Vertex Generation. To be published in *Survey Methodology*.

De Waal, T., J. De Waard and R. Plomp, 2001, Manual WAID (4.1), Report (research paper 0118), Statistics Netherlands, Voorburg.

De Waal, T. and A.J. Pieters, 1995, ARGUS User's Guide. Report, Statistics Netherlands, Voorburg.

De Waal, T. and R. Quere, 2003, A Fast and Simple Algorithm for Automatic Editing of Mixed Data. Submitted to *Journal of Official Statistics*.

De Waal, T., R. Renssen and F. Van de Pol, 2000, Graphical Macro-Editing: Possibilities and Pitfalls. *Proceedings of the Second International Conference on Establishment Surveys*, Buffalo, 579-588.

De Waal, T. and F. Van de Pol, 1997, A Recipe for Applying CherryPi in the Edit Process. *UN/ECE Work Session on Statistical Data Editing*, Prague.

De Waal, T. and L.C.R.J. Willenborg, 1994a, Minimising the Number of Local Suppressions in a Microdata Set. Report (BPA number: 5997-94-M1), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1994b, Development of ARGUS: Past, Present, Future. Report (BPA number: 7065-94-M1), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1994c, Principles of Statistical Disclosure Control. Report (BPA number: 2901-94-M1), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1995a, Statistical Disclosure Control and Sampling Weights. Report (BPA number: 3216-95-RSM), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1995b, Local Suppression in Statistical Disclosure Control and Data Editing. Report (BPA number: 3441-95-RSM), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1995c, Optimal Global Recoding and Local Suppression. Report (BPA number: 5286-95-RSM), Statistics Netherlands, Voorburg.

De Waal, T. and L.C.R.J. Willenborg, 1996, A View on Statistical Disclosure Control for Microdata. *Survey Methodology*, **22**, 95-103.

De Waal, T. and L.C.R.J. Willenborg, 1997, Statistical Disclosure Control and Sampling Weights. *Journal of Official Statistics*, **13**, 417-434.

De Waal, T. and L.C.R.J. Willenborg, 1998, Optimal Local Suppression in Microdata. *Journal of Official Statistics*, **14**, 421-435.

De Waal, T. and L.C.R.J. Willenborg, 1999a, Exact Disclosure in a Super-Table, *Netherlands Official Statistics*, **14**, 11-16.

De Waal, T. and L.C.R.J. Willenborg, 1999b, Information Loss through Global Recoding and Local Suppression. *Netherlands Official Statistics*, **14**, 17-20.

De Waal, T. and H. Wings, 1999, From CherryPi to SLICE. Report (research paper 9908), Statistics Netherlands, Voorburg.

De Wolf, P.P., 1999, A Heuristic Approach to Cell Suppression in Hierarchical Tables. Report (research paper 9913), Statistics Netherlands, Voorburg.

De Wolf, P.P., J. Gouweleeuw, P. Kooiman and L. Willenborg, 1997, Reflections on PRAM. Report (research paper 9742), Statistics Netherlands, Voorburg.

Dines, L.L., 1927, On Positive Solutions of a System of Linear Equations. *Annals of Mathematics*, **28**, 386-392.

Domingo-Ferrer, J. (ed.), 2002, *Inference Control in Statistical Databases – From Theory to Practice*. Lecture Notes in Computer Science, **2316**, Springer-Verlag, New York.

Domingo-Ferrer, J., J.M. Mateo-Sanz and V. Torra, 2001, Comparing SDC Methods for Microdata on the Basis of Information Loss and Disclosure Risk. *Proceedings of the NTTS&ETK 2001*, 807-826.

Domingo-Ferrer, J. and J.M. Mateo-Sanz, 2002, Practical Data-Oriented Microaggregation for Statistical Disclosure Control. *IEEE Transactions on Knowledge and Data Engineering*, 189-201.

References

Doyle, P., J. Lane, J. Theeuwes and L. Zayatz (eds.), 2001, *Confidentiality, Disclosure and Data Access*. North-Holland, Amsterdam.

Duarte De Carvalho, F., N.P. Dellaert and M. De Sanches Osório, 1994, Statistical Disclosure in Two-Dimensional Tables: General Tables. *Journal of the American Statistical Association*, **89**, 1547-1557.

Duffin, R.J., 1974, On Fourier's Analysis of Linear Inequality Systems. *Mathematical Programming Studies*, **1**, 71-95.

Duncan, G.T. and D. Lambert, 1986, Disclosure-Limited Data Dissemination (with discussion). *Journal of the American Statistical Association*. **81**, 10-28.

Duncan, G.T. and D. Lambert, 1989, The Risk of Disclosure for Microdata. *Journal of the Business and Economic Statistics*. **7**, 207-217.

Duncan, G.T. and S. Mukherjee, 2000, Optimal Disclosure Limitation Strategy in Statistical Databases: Deterring Tracker Attacks Through Additive Noise. *Journal of the American Statistical Association*, **95**, 720-729.

Fagan, J., B. Greenberg and B. Hemming, 1988, Controlled Rounding of Three-Dimensional Tables, Statistical Research Division Report Series, Bureau of the Census, Washington D.C.

Federal Committee on Statistical Methodology, 1990, Data Editing in Federal Statistical Agencies. Statistical Policy Working Paper 18, Washington DC: US Office of Management and Budget.

Fellegi, I.P., 1975, Controlled Random Rounding. *Survey Methodology*, **1**, 123-133.

Fellegi, I.P. and D. Holt, 1976, A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association*, **71**, 17-35.

Ferguson, D.P., 1994, An Introduction to the Data Editing Process. *Statistical Data Editing (Volume 1); Methods and Techniques*, United Nations, Geneva.

Fienberg, S.E., 1994, Conflicts Between the Needs for Access to Statistical Information and Demands for Confidentiality. *Journal of Official Statistics*, **10**, 115-132.

Fienberg, S.E., U.E. Makov and RJ. Steele, 1998, Disclosure Limitation Using Perturbation and Related Methods for Categorical Data. *Journal of Official Statistics*, **14**, 485-511.

Fienberg, S.E. and U.E. Makov, 2001, Uniqueness, Urn Models and Disclosure Risk. *Research in Official Statistics*, **4**, 23-40.

Fillion, J.M. and I. Schiopu-Kratina, 1993, On the Use of Chernikova's Algorithm for Error Localisation. Report, Statistics Canada.

Fischetti, M. and J.J. Salazar-González, 1998a, Experiments with Controlled Rounding for Statistical Disclosure Control in Tabular Data with Linear Constraints. Report, University of Padova.

Fischetti, M. and J.J. Salazar-González, 1998b, Models and Algorithms for Optimizing Cell Suppression in Tabular Data with Linear Constraints. Report, University of Padova.

Fischetti, M. and J.J. Salazar-González, 1998c, Partial Cell Suppression: A New Methodology for Statistical Disclosure Control. Report, University of La Laguna, Tenerife.

Fischetti, M. and J.J. Salazar-González, 2000, Models and Algorithms for Optimizing Cell Suppression in Tabular Data with Linear Constraints. *Journal of the American Statistical Association*, **95**, 916-928.

Fourier, J.B.J., 1826, Solution d'une Question Particulière du Calcul des Inégalités. In: *Oeuvres II*, Paris.

Franconi, L., 1999, Level of Safety in Microdata: Comparisons Between Different Definitions of Disclosure Risk and Estimation Models. *Eurostat/UN-ECE Work Session on Statistical Data Confidentiality*.

Freund, R.J. and H.O. Hartley, 1967, A Procedure for Automatic Data Editing. *Journal of the American Statistical Association*, **62**, 341-352.

Fuller, W.A., 1993, Masking Procedures for Microdata Disclosure Limitation. *Journal of Official Statistics,* **9**, 383-406.

Garfinkel, R.S., A.S. Kunnathur and G.E. Liepins, 1986, Optimal Imputation of Erroneous Data: Categorical Data, General Edits. *Operations Research*, **34**, 744-751.

Garfinkel, R.S., A.S. Kunnathur and G.E. Liepins, 1988, Error Localization for Erroneous Data: Continuous Data, Linear Constraints. *SIAM Journal on Scientific and Statistical Computing*, **9**, 922-931.

Gelman, A., J.B. Carlin, H.S. Stern and D.B. Rubin, 1995, *Bayesian Data Analysis*. Chapman and Hall, New York.

Geurts, J., 1992, Heuristics for Cell Suppression in Tables. Report (BPA number: 11450-92-M1), Statistics Netherlands, Voorburg.

Giessing, S., 1998, Looking for Efficient Automated Secondary Cell Suppression Systems: A Software Comparison. *Research in Official Statistics*, **2**, 69-85.

Giessing, S., 1999, A Survey on Software Packages for Automated Secondary Cell Suppression. *Joint ECE/Eurostat Work Session on Statistical Data Confidentiality*, Thessaloniki.

Giessing, S., 2000, Transferable Software for Automated Secondary Cell Suppression. *Research in Official Statistics*, **3**, 119-132.

Giessing, S. and A. Hundepool, 2001, The CASC Project: Integrating Best Practice Methods for Statistical Confidentiality. *Proceedings of the NTTS&ETK 2001*, 775-793.

Glover, F., 1973, Convexity Cuts and Cut Search. *Operations Research*, **21**, 123-134.

References

Glover, F. and D. Klingman, 1973, The Generalized Lattice-Point Problem. *Operations Research*, **21**, 141-155.

Glover, F., D. Klingman and J. Stutz, 1974, The Disjunctive-Facet Problem: Formulation and Solution Techniques. *Operations Research*, **22**, 582-601.

Gouweleeuw, J., P. Kooiman, L.C.R.J. Willenborg and P.P. De Wolf, 1998, Post Randomisation for Statistical Disclosure Control: Theory and Implementation. *Journal of Official Statistics*, **14**, 463-478.

Granquist, L. 1984, Data Editing and its Impact on the Further Processing of Statistical Data. *Workshop on Statistical Computing*, Budapest.

Granquist, L., 1990, A Review of some Macro-editing Methods for Rationalizing the Editing Process. *Proceedings of the Statistics Canada Symposium*, 225-234.

Granquist, L., 1995, Improving the Traditional Editing Process. In: *Business Survey Methods* (ed. Cox, Binder, Chinnappa, Christianson & Kott), John Wiley & Sons, Inc., New York, 385-401.

Granquist, L., 1997, The New View on Editing. *International Statistical Review*, **65**, 381-387.

Granquist, L. and J. Kovar, 1997, Editing of Survey Data: How Much is Enough?. In: *Survey Measurement and Process Quality* (ed. Lyberg, Biemer, Collins, De Leeuw, Dippo, Schwartz & Trewin), John Wiley & Sons, Inc., New York, 415-435.

Greenberg, B., 1990, Disclosure Avoidance Research at the U.S. Census Bureau. *Proceedings of the Annual Research Conference*, U.S. Bureau of the Census, 144-166.

Greenberg, B.V. and L.V. Zayatz, 1992, Strategies for Measuring Risk in Public Use Microdata Files. *Statistica Neerlandica,* **46**, 33-48.

Hadley, G., 1962, *Linear Programming*. Addison-Wesley, Reading, Massachusetts.

Harte, P., 2000, Testing Automatic Editing by Means of the Simplex Method. Internal report (BPA number: 3296-00-RSM), Statistics Netherlands, Voorburg.

Hidiroglou, M.A., and J.-M. Berthelot, 1986, Statistical Editing and Imputation for Periodic Business Surveys. *Survey Methodology*, **12**, 73-83.

Hillier, F.S. and G.J. Lieberman, 1995, *Introduction to Mathematical Programming* (second edition). McGraw-Hill, Inc., New York.

Hirsch, W.M. and G.B. Dantzig, 1968, The Fixed Charge Problem. *Naval Research Logistics Quarterly*, **15**, 413-424.

Hoogland, J., 1994, Protecting Microdata sets Against Statistical Disclosure by Means of Compound Poisson Distributions (in Dutch). Report (BPA number: 4244-94-M1), Statistics Netherlands, Voorburg.

Hoogland, J., 2002, Selective Editing by Means of Plausibility Indicators. *UN/ECE Work Session on Statistical Data Editing*, Helsinki.

Hoogland, J., Houbiers, M. and T. De Waal, 2002, Workbook for the Data Editing Course – Version 2. Report (research paper 0214), Statistics Netherlands, Voorburg.

Hooker, J., 2000, *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. John Wiley & Sons, New York.

Houbiers, M., 1999a, Automatic Editing of the 1996 Industrial Waste Survey by Means of CherryPi (in Dutch). Internal report (BPA number: 3230-99-RSM), Statistics Netherlands, Voorburg.

Houbiers, M., 1999b, Application of Duffin's Analysis of Linear Inequality Systems to the Error Localisation Problem and Chernikova's Algorithm. Report (research paper 9922), Statistics Netherlands.

Houbiers, M., 1999c, Edit Checking in Error Localisation. Report (research paper 9932), Statistics Netherlands, Voorburg.

Houbiers, M., R. Quere and T. De Waal, 1999, Automatically Editing the 1997 Survey on Environmental Costs. Internal report (BPA number: 4917-99-RSM), Statistics Netherlands, Voorburg.

Hox, J.J., 1999, A Review of Current Software for Handling Missing Data. *Kwantitatieve Methoden*, **62**, 123-138.

Hundepool, A., A. Van de Wetering, P.P. De Wolf, S. Giessing, M. Fischetti, J.J. Salazar and A. Caprara, 2002a, τ-ARGUS User's Manual (Version 2.1). Report, Statistics Netherlands, Voorburg.

Hundepool, A., A. Van de Wetering, L. Franconi, A. Capobianchi, P.P. De Wolf, 2002b, μ-ARGUS User's Manual (Version 3.1). Report, Statistics Netherlands, Voorburg.

Hurkens, C.A.J. and Tiourine, S.R., 1998a, On Solving Huge Set-Cover Models of the Microdata Protection Problem. Paper presented at SDP '98, 25-27 March 1998, Lisbon, Portugal.

Hurkens, C.A.J. and Tiourine, S.R., 1998b, Models and Methods for the Microdata Protection Problem. *Journal of Official Statistics*, **14**, 437-447.

ILOG CPLEX 7.5 Reference Manual, 2001, ILOG, France.

Imbert, J.L., 1993, Variable Elimination for Disequations in Generalized Linear Constraint Systems. *The Computer Journal*, **36**, 473-484.

Jewett, R., 1993, Disclosure Analysis for the 1992 Economic Census. Unpublished Manuscript, Economic Programming Division, U.S. Bureau of the Census, Washington, DC.

Kalton, G. and D. Kasprzyk, 1986, The Treatment of Missing Survey Data. *Survey Methodology*, **12**, 1-16.

Kartika, W., 2001, Consistent Imputation for Categorical and Numerical Data. Report (research paper 0113), Statistics Netherlands, Voorburg.

References

Keller, W.J. and J.A. Bethlehem, 1992, Disclosure Protection of Microdata: Problems and Solutions. *Statistica Neerlandica,* **46**, 5-19.

Keller, W.J. and L.C.R.J. Willenborg, 1993, Microdata Release Policy at the Netherlands CBS. *Proceedings of the International Seminar on Statistical Confidentiality*, Dublin, 439-444.

Kelly, J.P., 1990, *Confidentiality Protection in Two- and Three-Dimensional Tables*. Ph.D. Thesis, University of Maryland, College Park, Maryland.

Kelly, J.P., B.L. Golden and A.A. Assad, 1992, Cell Suppression: Disclosure Protection for Sensitive Tabular Data. *Networks*, **22**.

Kirkendall, N. and G. Sande, 1998, Comparison of Systems Implementing Automated Cell Suppression for Economic Statistics. *Journal of Official Statistics*, **14**, 513-535.

Kohler, D.A., 1973, Translation of a Report by Fourier on His Work on Linear Inequalities. *Opsearch*, **10**, 38-42.

Kim, J.J., 1986, A Method for Limiting Disclosure in Microdata Based on Random Noise and Transformation. *Proceedings of the Section on Survey Research Methods, American Statistical Association*, Washington, DC, 303-308.

Kooiman, 1998, One Person's Noise in Another Person's Signal: A Comment on Fienberg at al. (1998). Report (research paper 9827), Statistics Netherlands, Voorburg.

Kooiman, P., L. Willenborg and J. Gouweleeuw, 1997, PRAM: a Method for Disclosure Control of Microdata. Report (research paper 9705), Statistics Netherlands, Voorburg.

Korte, B. and J. Vygen, 2000, *Combinatorial Optimization – Theory and Algorithms*. Springer-Verlag, Berlin.

Kovar, J. and P. Whitridge, 1990, Generalized Edit and Imputation System; Overview and Applications. *Revista Brasileira de Estadistica*. **51**, 85-100.

Kovar, J. and P. Whitridge, 1995, Imputation of Business Survey Data. In: *Business Survey Methods* (ed. Cox, Binder, Chinnappa, Christianson & Kott), John Wiley & Sons, Inc., New York, 403-423.

Kovar, J. and W.E. Winkler, 1996, Editing Economic Data. *UN/ECE Work Session on Statistical Data Editing*, Voorburg.

Laflamme, F., Barrett, C., Johnson, W., and Ramsay, L., 1996, Experiences in Re-engineering the Approach to Editing and Imputing Canadian Imports Data. *Proceedings of the Bureau of the Census Annual Research Conference and Technology Interchange*, 1025-1037.

Lambert, D., 1993, Measures of Disclosure Control and Harm. *Journal of Official Statistics*, **9**, 313-331.

Liepins, G.E., 1980, Refinements to the Boolean Approach to Automatic Data Editing. Report, Oak Ridge National Laboratory.

Liepins, G.E., 1981, A Rigorous, Systematic Approach to Automatic Data Editing and Its Statistical Basis. Report, Oak Ridge National Laboratory.

Liepins, G.E., 1983, Fourier-Motzkin Elimination for Mixed Systems. Report, Oak Ridge National Laboratory.

Liepins, G.E., 1984, Algorithms for Error Localization of Discrete Data. Report, Oak Ridge National Laboratory.

Liepins, G.E., R.S. Garfinkel and A.S. Kunnathur, 1982, Error Localization for Erroneous Data: A Survey. *TIMS/Studies in the Management Sciences*, **19**, 205-219.

Mardia, K.V., J. T. Kent and J.M. Bibby, 1979, *Multivariate Analysis*. Academic Press, London.

Marriott, K. and P.J. Stuckey, 1998, *Programming with Constraints – An Introduction*. MIT Press, Cambridge, Massachusetts.

Marsh, C., A. Dale and C.J. Skinner, 1994, Safe Data Versus Safe Settings: Access to Microdata from the British Census. *International Statistical Review*, **62**, 32-54.

McKeown, P.G., 1975, A Vertex Ranking Procedure for Solving the Linear Fixed-Charge Problem. *Operations Research*, **23**, 1183-1191.

McKeown, P.G., 1981, A Branch-and-Bound Algorithm for Solving Fixed Charge Problems. *Naval Research Logistics Quarterly*, **28**, 607-617.

McKeown, P.G., 1984, A Mathematical Programming Approach to Editing of Continuous Survey Data. *SIAM Journal on Scientific and Statistical Computing*, **5**, 784-797.

McKeown, P.G. and C.T. Ragsdale, 1990, A Computational Study of Using Preprocessing and Stronger Formulations to Solve Large General Fixed Charge Problems. *Computers and Operations Research*, **17**, 9-16.

McMullen, P., 1970, The Maximum Number of Faces of a Convex Polytope. *Mathematika*, **17**, 179-184.

Mokken, R.J., P. Kooiman, J. Pannekoek and L.C.R.J. Willenborg, 1992, Disclosure Risks for Microdata. *Statistica Neerlandica,* **46**, 49-67.

Mokken, R.J., J. Pannekoek, and L.C.R.J. Willenborg, 1989, Microdata and Disclosure Risks, *CBS Select*, **5**, Statistical Essays, Staatsuitgeverij, The Hague, 181-200.

Moore, R.A., 1996a, Controlled Data-Swapping Techniques for Masking Public Use Microdata Sets. Report, Statistical Research Division RR 96/04, U.S. Bureau of the Census, Washington, DC.

Moore, R.A., 1996b, Analysis of the Kim-Winkler Algorithm for Masking Microdata Files – How Much Masking is Necessary and Sufficient? Conjectures for the Development of a Controllable Algorithm. Report, Statistical Research Division RR 96/05, U.S. Bureau of the Census, Washington, DC.

Motzkin, T.S., 1936, *Contributions to the Theory of Linear Inequalities* (in German). Dissertation, University of Basel.

Müller, W., U. Blien, P. Knoche, H. Wirth et al., 1991, *The Factual Anonymity of Microdata* (in German). Metzler-Poeschel Verlag, Stuttgart.

Nemhauser, G.L. and L.A. Wolsey, 1988, *Integer and Combinatorial Optimisation.* Wiley, New York.

Paass, G., 1988, Disclosure Risk and Disclosure Avoidance for Microdata. *Journal of Business and Economic Studies*, **6**, 487-500.

Paass G. and U. Wauschkuhn, 1985, Data Access, Data Protection and Anonymisation – Analysis Potential and Identifiability of Anonymised Individual Data (in German). Gesellschaft fur Mathematik und Datenverarbeitung, Oldenbourg- Verlag, Munich.

Pannekoek, J., 1992, Disclosure Control of Extreme Values of Continuous Identifiers (in Dutch). Report (BPA number: 7787-92-M1), Statistics Netherlands, Voorburg.

Pannekoek, J., 1999, Statistical Methods for Some Simple Disclosure Limitation Rules. *Statistica Neerlandica*, **53**, 55-67.

Pannekoek, J. and T. De Waal, 1995, Synthetic and Combined Estimators in Statistical Disclosure Control. Report (BPA number: 9613-94-RSM), Statistics Netherlands, Voorburg.

Pannekoek, J. and T. De Waal, 1998, Synthetic and Combined Estimators in Statistical Disclosure Control. *Journal of Official Statistics*, **14**, 399-410.

Pergamentsev, S.Y., 1998, Automatic Statistical Data Correction. Report (research paper 9826), Statistics Netherlands, Voorburg.

Pfeffermann, D., 2002, Small Area Estimation – New Developments and Directions. *International Statistical Review*, **70**, 125-143.

Pierzchala, M., 1995, Editing Systems and Software. In: *Business Survey Methods* (ed. Cox, Binder, Chinnappa, Christianson & Kott), John Wiley & Sons, Inc., New York, 425-441.

Pieters, A.J., and T. De Waal, 1995, A Demonstration of ARGUS. Report (BPA number: 3947-95-RSM), Statistics Netherlands, Voorburg.

Prékopa, A., 1972, On the Number of Vertices of Random Convex Polyhedra. *Periodica Mathematica Hungarica*. **2**, 259-282.

Pugh, W., 1992, The Omega Test: A Fast and Practical Integer Programming Algorithm for Data Dependence Analysis. *Communications of the ACM*, **35**, 102-114.

Pugh, W. and D. Wonnacott, 1994, Experiences with Constraint-Based Array Dependence Analysis. In: *Principles and Practice of Constraint Programming, Second International Workshop*. Lecture Notes in Computer Science, **768**, Springer-Verlag, Berlin.

Quere, R., 2000, Automatic Editing of Numerical Data. Report (research paper 0016), Statistics Netherlands, Voorburg.

Quere, R. and T. De Waal, 2000, Error Localisation in Mixed Data Sets. Report (research paper 0017), Statistics Netherlands, Voorburg.

Ragsdale, C.T. and P.G. McKeown, 1991, An Algorithm for Solving Fixed-Charge Problems Using Surrogate Constraints. *Computers & Operations Research*, **18**,87-96.

Ragsdale, C.T. and P.G. McKeown, 1996, On Solving the Continuous Data Editing Problem. *Computers & Operations Research*, **23**, 263-273.

Repsilber, D., 1993, Safeguarding Secrecy in Aggregate Data. *Proceedings of the International Seminar on Statistical Confidentiality*, Dublin.

Rinott, Y., 2003, On Models for Statistical Disclosure Risk Estimation. *Joint UN/ECE and Eurostat Work Session on Statistical Data Confidentiality*, Luxembourg.

Robertson, D., 1992, Cell Suppression at Statistics Canada. *Proceedings of the Annual Research Conference*, US Bureau of the Census, Washington DC, 107-135.

Robertson, D., 1995, Automated Disclosure Control at Statistics Canada. *Proceedings of the Second International Seminar on Statistical Confidentiality*, Luxembourg.

Robertson, D., 2000, Improving Statistics Canada's Cell Suppression Software (CONFID). *Proceedings in Computational Statistics 2000* (Eds. J.G. Bethlehem and P.G.M. Van der Heijden), Physica-Verlag, New York, 403-408.

Robinson, J.A., 1965, A Machine-Oriented Logic Based on the Resolution Principle. *Journal Assoc. Comput. Mach*, **12**, 23-41.

Robinson, J.A., 1968, The Generalized Resolution Principle. In: *Machine Intelligence 3* (Eds. Dale and Michie). 77-93, Oliver and Boyd, Edinburgh.

Rubin, D.B., 1993, Discussion Statistical Disclosure Control. *Journal of Official Statistics*, **9**, 461-468.

Rubin, D.S., 1975, Vertex Generation and Cardinality Constrained Linear Programs. *Operations Research*, **23**, 555-565.

Rubin, D.S., 1977, Vertex Generation Methods for Problems with Logical Constraints. *Annals of Discrete Mathematics*, **1**, 457-466.

Russell, S. and P. Norvig, 1995, *Artificial Intelligence, a Modern Approach*. Prentice-Hall, Englewood Cliffs, New Jersey.

Samuels, S.M., 1998, A Bayesian Species-Sampling-Inspired Approach to the Uniques Problem in Microdata Disclosure Risk Assessment. *Journal of Official Statistics*, **14**, 373-383.

Sande, G., 1977, Towards Automated Disclosure Analysis for Establishment Based Statistics. Report, Statistics Canada.

Sande, G., 1978a, An Algorithm for the Fields to Impute Problems of Numerical and Coded Data. Technical report, Statistics Canada.

Sande, G., 1978b, A Theorem Concerning Elementary Aggregations. Report, Statistics Canada.

Sande, G., 1978c, Confidentiality and Polyhedra – An Analysis of Suppressed Entries and Cross-Tabulations. Report, Statistics Canada.

Sande, G., 1984, Automated Cell Suppression Software to Preserve Confidentiality of Business Statistics. *Statistical Journal of the United Nations ECE*, **2**, 33-41.

Sande, G., 1999, Structure of the ACS Automated Cell Suppression System. *Joint ECE/Eurostat Work Session on Statistical Data Confidentiality*.

Sande, G., 2000a, *Personal communication.*

Sande, G., 2000b, Blunders by Official Statistical Agencies While Protecting the Confidentiality of Business Statistics (unpublished draft paper).

Sande, G., 2001, Methods for Data Directed Microaggregation in One Dimension. *Proceedings of the NTTS&ETK 2001*,

Sande, G., 2002, Exact and Approximate Methods for Data Directed Microaggregation in One or More Dimensions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **10**, 459-476.

Schaffer, J., 1987, Procedure for Solving the Data-Editing Problem with Both Continuous and Discrete Data Types. *Naval Research Logistics*, **34**, 879-890.

Schiopu-Kratina, I. and J.G. Kovar, 1989, Use of Chernikova's Algorithm in the Generalized Edit and Imputation System. Methodology Branch Working Paper BSMD 89-001E, Statistics Canada.

Schrijver, A., 1986, *Theory of Linear and Integer Programming*. John Wiley & Sons, New York.

Shlomo, N., 2003, Accessing Microdata via the Internet. *Joint UN/ECE and Eurostat Work Session on Statistical Data Confidentiality*, Luxembourg.

Skinner, C.J. 1992, On Identification Disclosure and Prediction Disclosure for Microdata. *Statistica Neerlandica,* **46**, 21-32.

Skinner, C.J. and D.J. Holmes, 1992, Modelling Population Uniqueness. International Seminar on Statistical Confidentiality, Dublin.

Skinner, C.J. and D.J. Holmes, 1998, Estimating the Re-identification Risk Per Record in Microdata. *Journal of Official Statistics*, **14**, 361-372.

Skinner, S., C. Marsh, S. Openshaw and C. Wymer, 1990, Disclosure Avoidance for Census Microdata in Great Britain. *Proceedings of the 1990 Annual Research Conference,* U.S. Bureau of the Census, Washington, DC, 131-143.

Skinner, C.J., C. Marsh, S. Openshaw and C. Wymer, 1994, Disclosure Control for Census Microdata. *Journal of Official Statistics*, **10**, 31-51.

Sonquist, J.N., E.L. Baker and J.A. Morgan, 1971, Searching for Structure. Institute for Social Research, University of Michigan.

Special Issue on Confidentiality and Data Access, 1993, *Journal of Official Statistics*, **9**.

Spjøtvoll, E. and I. Thomsen, 1987, Application of Some Empirical Bayes Methods to Small Area Estimation. *Proceedings of the 46th ISI Conference*.

Sullivan, G.R., 1989, *The Use of Added Error to Avoid Disclosure in Microdata Releases*. Ph.D. Thesis, Iowa State University.

Todaro, T.A., 1999, Overview and Evaluation of the AGGIES Automated Edit and Imputation System. *UN/ECE Work Session on Statistical Data Editing*, Rome.

Traa, M., 2001, Prototype Software for Cell Suppression Based on Linear Programming. Report (research paper 0130), Statistics Netherlands, Voorburg.

Tukey, J.W., 1977, *Exploratory Data Analysis*. Addison-Wesley, London.

US Department of Commerce, 1978, Report on Statistical Disclosure and Disclosure Avoidance Techniques. Statistical Policy Working Paper 2, Washington DC.

Van den Broeke, G., 2000, The Omega-Test (in Dutch). Internal Report (BPA number: 4031-00-TMO), Statistics Netherlands, Voorburg.

Van den Broeke, G., 2001, Inconsistencies and Redundancies in Edit Sets. Report (research paper 0112), Statistics Netherlands, Voorburg.

Van de Pol, F., F. Bakker and T. de Waal, 1997, On Principles for Automatic Editing of Numerical Data with Equality Checks. Report (research paper 9736), Statistics Netherlands, Voorburg.

Van de Pol, F. and J. Bethlehem, 1997, Data Editing Perspectives. *Statistical Journal of the United Nations ECE*, **14**, 153-171.

Van de Pol, F., A. Buijs, G. Van der Horst and T. De Waal, 1997, Integrating Automatic Editing, Computer-Assisted Editing and Graphical Macro-Editing. Report (research paper 9739), Statistics Netherlands, Voorburg.

Van de Pol, F., and Diederen, B., 1996, A Priority Index for Macro-Editing the Netherlands Foreign Trade Survey. *Proceedings of the Data Editing Workshop and Exposition*, Bureau of Labor Statistics, Washington D.C.

Van de Pol, F. and Molenaar, W., 1995, Selective and Automatic Editing with CADI-applications. In: V. Kuusela, (ed.), *Essays on Blaise 1995, Proceedings of the Third International Blaise Users' Conference*, Statistics Finland, Helsinki, 159-168.

Van Gelderen, R., 1995, ARGUS: Statistical Disclosure Control of Survey Data. Report (BPA number: 2543-95-RSM), Statistics Netherlands, Voorburg.

Van Kouwen, M., 1993, Aspects of Statistical Disclosure Control for Sampling Weights (in Dutch). Internal Report (BPA number: 5587-93-M1), Statistics Netherlands, Voorburg.

Van Laarhoven, P.J.M and E.H.L Aarts, 1987, *Simulated Annealing: Theory and Applications*. Mathematics and Its Applications, D. Reidel Publishing Company, Dordrecht.

Van Riessen, P., 2002, Automatic Editing by Means of CPLEX. Internal report (BPA number: 975-02-TMO), Statistics Netherlands, Voorburg.

Vasko, F.J. and G.R. Wilson, 1986, Hybrid Heuristics for Minimum Cardinality Set-Covering Problems. *Naval Research Logistics Quarterly*, **33**, 241-249.

Vazirani, V.V., 2001, *Approximation Algorithms*. Springer-Verlag, Berlin.

Verboon, P., 1994, Some Ideas for a Masking Measure for Statistical Disclosure Control. Report (BPA number: 9034-94-RSM), Statistics Netherlands, Voorburg.

Vesseur, J.W., 1998, Addition of Noise to the Sampling Weights in Statistical Disclosure Control. Report (research paper 9829), Statistics Netherlands, Voorburg.

Walker, W.E., 1976, A Heuristic Adjacent Extreme Point Algorithm for the Fixed Charge Problem. *Management Science*, **5**, 587-596.

Walukiewicz, S., 1990, *Integer Programming*. Mathematics and Its Applications / East European Series, Kluwer Academic Publishers, Dordrecht.

Warners, J.P., 1999, *Non-Linear Approaches to Satisfiability Problems*. Ph.D. Thesis, Eindhoven University of Technology.

Weng, S.S., 2002, Elimination in Linear Editing and Error Localization. Report (RDD Research report number RDD-02-06), United States Department of Agriculture & National Agricultural Statistics Service, Fairfax.

Willeboordse, A. (ed.), 1998, *Handbook on the Design and Implementation of Business Surveys*. Office for Official Publications of the European Communities, Luxembourg.

Willenborg, L.C.R.J., 1988, Computational Aspects of Survey Processing. Ph.D. Thesis, Tilburg University.

Willenborg, L.C.R.J., 1990a, Remarks on Disclosure Control of Microdata. Report (BPA number: 11562-90-M1), Statistics Netherlands, Voorburg.

Willenborg, L.C.R.J., 1990b, Disclosure Risks for Microdata Sets: Stratified Populations and Multiple Investigators. Report (BPA number: 10088-90-M1), Statistics Netherlands, Voorburg.

Willenborg, L.C.R.J., 1993, Discussion Statistical Disclosure Limitation. *Journal of Official Statistics,* **9**, 469-474.

Willenborg, L. and T. De Waal, 1996, *Statistical Disclosure Control in Practice*. Lecture Notes in Statistics, **111**, Springer-Verlag, New York.

Willenborg, L. and T. De Waal, 2001, *Elements of Statistical Disclosure Control in Practice*. Lecture Notes in Statistics, **155**, Springer-Verlag, New York.

Willenborg, L.C.R.J., R.J. Mokken and J. Pannekoek, 1990, Microdata and Disclosure Risks. *Proceedings of the 1990 Annual Research Conference,* U.S. Bureau of the Census, Washington DC, 167-180.

Williams, H.P., 1976, Fourier-Motzkin Elimination Extension to Integer Programming. *Journal of Combinatorial Theory (A)*, **21**, 118-123.

Williams, H.P., 1983, A Characterisation of All Feasible Solutions to an Integer Program. *Discrete Applied Mathematics*, **5**, 146-155.

Williams, H.P., 1986, Fourier's Method of Linear Programming and Its Dual. *American Mathematical Monthly*, **93**, 681-695.

Williams, H.P. and S.C. Brailsford, 1996, Computational Logic and Integer Programming. In: *Advances in Linear and Integer Programming* (ed. J.E. Beasley), Clarendon Press, Oxford, 249-281.

Winkler, W.E., 1995, Editing Discrete Data. *UN/ECE Work Session on Statistical Data Editing*, Athens.

Winkler, W.E., 1998, Re-identification Methods for Evaluating the Confidentiality of Analytically Valid Microdata. *Research in Official Statistics*, **2**, 87-104.

Winkler, W.E., 1999, State of Statistical Data Editing and Current Research Problems. *UN/ECE Work Session on Statistical Data Editing*, Rome.

Winkler, W.E. and L.A. Draper, 1997, The SPEER Edit System. *Statistical Data Editing (Volume 2); Methods and Techniques*, United Nations, Geneva.

Winkler, W.E. and T.F. Petkunas, 1997, The DISCRETE Edit System. *Statistical Data Editing (Volume 2)*; *Methods and Techniques*. United Nations, Geneva.

Wolsey, L.A., 1998, *Integer Programming*. John Wiley & Sons, New York.

Zaslavsky, A.M. and N.J. Horton, 1998, Balancing Disclosure Risk Against the Loss of Nonpublication. Journal of Official Statistics, **14**, 411-419.

Curriculum Vitae

Ton de Waal werd op 22 oktober 1963 te Den Haag geboren. In 1982 behaalde hij zijn VWO-diploma aan het Segbroekcollege in Den Haag. Aansluitend begon hij met de studies Wiskunde en Natuurkunde aan de Rijksuniversiteit Leiden. Voor beide studies behaalde hij in 1983 de propedeuse. Nadien hij heeft zich uitsluitend op de studie Wiskunde geconcentreerd. In 1987 behaalde hij hiervoor het doctoraalexamen. Vervolgens heeft hij een lerarenopleiding ter verkrijging van de lerarenbevoegdheid (eerste graad) aan dezelfde universiteit gevolgd. Het diploma voor deze opleiding werd in januari 1989 behaald. Vanaf maart 1989 volgde hij de tweedefase opleiding "Wiskunde voor de Industrie" aan de Technische Universiteit Eindhoven. Deze opleiding werd in 1991 succesvol afgesloten.

Na afronding van zijn studies is Ton enige tijd werkzaam geweest als docent/SAS-programmeur aan de Technische Universiteit Eindhoven. In 1993 trad hij in dienst als methodoloog bij het Centraal Bureau voor de Statistiek. Momenteel is hij daar nog steeds werkzaam bij de Sector Methoden en Ontwikkeling.

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT (ERIM)

ERIM PH.D. SERIES

RESEARCH IN MANAGEMENT

ERIM Electronic Series Portal: http://hdl.handle.net/1765/1

Title:	**Operational Control of Internal Transport**
Author:	J. Robert van der Meer
Promotor(es):	Prof.dr. M.B.M. de Koster, Prof.dr.ir. R. Dekker
Defended:	September 28, 2000
Series number:	1
ISBN:	90-5892-004-6

Title:	**Quantitative Models for Reverse Logistics**
Author:	Moritz Fleischmann
Promotor(es):	Prof.dr.ir. J.A.E.E. van Nunen, Prof.dr.ir. R. Dekker
Defended:	October 5, 2000
Series number:	2
Published:	Lecture Notes in Economics and Mathematical Systems, Volume 501, 2001, Springer Verlag, Berlin,
ISBN:	3540 417 117

Title:	**Optimization Problems in Supply Chain Management**
Author:	Dolores Romero Morales
Promotor(es):	Prof.dr.ir. J.A.E.E. van Nunen, dr. H.E. Romeijn
Defended:	October 12, 2000
Series number:	3
ISBN:	90-9014078-6

Title:	**Layout and Routing Methods for Warehouses**
Author:	Kees Jan Roodbergen
Promotor(es):	Prof.dr. M.B.M. de Koster, Prof.dr.ir. J.A.E.E. van Nunen
Defended:	May 10, 2001
Series number:	4
ISBN:	90-5892-005-4

Title:	**Rethinking Risk in International Financial Markets**
Author:	Rachel Campbell
Promotor(es):	Prof.dr. C.G. Koedijk
Defended:	September 7, 2001
Series number:	5
ISBN:	90-5892-008-9

Title:	**Labour Flexibility in China's Companies: An Empirical Study**
Author:	Yongping Chen
Promotor(es):	Prof.dr. A. Buitendam, Prof.dr. B. Krug
Defended:	October 4, 2001
Series number:	6
ISBN:	90-5892-012-7

Title:	**Strategic Issues Management: Implications for Corporate Performance**
Author:	Pursey P. M. A. R. Heugens
Promotor(es):	Prof.dr.ing. F.A.J. van den Bosch, Prof.dr. C.B.M. van Riel
Defended:	October 19, 2001
Series number:	7
ISBN:	90-5892-009-7

Title:	**Beyond Generics; A Closer Look at Hybrid and Hierarchical Governance**
Author:	Roland F. Speklé
Promotor(es):	Prof.dr. M.A. van Hoepen RA
Defended:	October 25, 2001
Series number:	8
ISBN:	90-5892-011-9

Title:	**Interorganizational Trust in Business to Business E-Commerce**
Author:	Pauline Puvanasvari Ratnasingam
Promotor(es):	Prof.dr. K. Kumar, Prof.dr. H.G. van Dissel
Defended:	November 22, 2001
Series number:	9
ISBN:	90-5892-017-8

Title:	**Outsourcing, Supplier-relations and Internationalisation: Global Source Strategy as a Chinese Puzzle**
Author:	Michael M. Mol
Promotor(es):	Prof.dr. R.J.M. van Tulder
Defended:	December 13, 2001
Series number:	10
ISBN:	90-5892-014-3

Title:	**The Business of Modularity and the Modularity of Business**
Author:	Matthijs J.J. Wolters
Promotor(es):	Prof. mr. dr. P.H.M. Vervest, Prof. dr. ir. H.W.G.M. van Heck
Defended:	February 8, 2002
Series number:	11
ISBN:	90-5892-020-8

Title:	**The Quest for Legitimacy; On Authority and Responsibility in Governance**
Author:	J. van Oosterhout
Promotor(es):	Prof.dr. T. van Willigenburg, Prof.mr. H.R. van Gunsteren
Defended:	May 2, 2002
Series number:	12
ISBN:	90-5892-022-4

Title:	**Information Architecture and Electronic Market Performance**
Author:	Otto R. Koppius
Promotor(es):	Prof.dr. P.H.M. Vervest, Prof.dr.ir. H.W.G.M. van Heck
Defended:	May 16, 2002
Series number:	13
ISBN:	90-5892-023 - 2

Title:	**Planning and Control Concepts for Material Handling Systems**
Author:	Iris F.A. Vis
Promotor(es):	Prof.dr. M.B.M. de Koster, Prof. dr. ir. R. Dekker
Defended:	May 17, 2002
Series number:	14
ISBN:	90-5892-021-6

Title:	**Essays on Agricultural Co-operatives; Governance Structure in Fruit and Vegetable Chains**
Author:	Jos Bijman
Promotor(es):	Prof.dr. G.W.J. Hendrikse
Defended:	June 13, 2002
Series number:	15
ISBN:	90-5892-024-0

Title:	**Analysis of Sales Promotion Effects on Household Purchase Behavior**
Author:	Linda H. Teunter
Promotor(es):	Prof.dr. ir. B. Wierenga, Prof.dr. T. Kloek
Defended:	September 19, 2002
Series number:	16
ISBN:	90-5892-029-1

Title:	**Incongruity between Ads and Consumer Expectations of Advertising**
Author:	Joost Loef
Promotor(es):	Prof.dr. W.F. van Raaij, prof. dr. G. Antonides
Defended:	September 26, 2002
Series number:	17
ISBN:	90-5892-028-3

Title:	**Creating Trust between Local and Global Systems**
Author:	Andrea Ganzaroli
Promotor(es):	Prof.dr. K. Kumar, Prof.dr. R.M. Lee
Defended:	October 10, 2002
Series number:	18
ISBN:	90-5892-031-3

Title:	**Coordination and Control of Globally Distributed Software Projects**
Author:	Paul C. van Fenema
Promotor(es):	Prof.dr. K. Kumar
Defended:	October 10, 2002
Series number:	19
ISBN:	90-5892-030-5

Title:	**Improving the Flexibility and Profitability of ICT-Enabled Business Networks: An Assessment Method and Tool**
Author:	Dominique J.E. Delporte- Vermeiren
Promotor(es):	Prof.mr.dr. P.H.M. Vervest, Prof.dr.ir. H.W.G.M. van Heck
Defended:	May 9, 2003
Series number:	20
ISBN:	90-5892-040-2

Title:	**Organizing Knowledge in Internal Networks. A Multilevel Study**
Author:	Raymond van Wijk
Promotor(es):	Prof.dr.ing. F.A.J. van den Bosch
Defended:	May 22, 2003
Series number:	21
ISBN:	90-5892-039-9

Title:	**Cyclic Railway Timetable Optimization**
Author:	Leon W.P. Peeters
Promotor(es):	Prof. Dr. L.G. Kroon, Prof.dr.ir. J.A.E.E. van Nunen
Defended:	June 6, 2003
Series number:	22
ISBN:	90-5892-042-9