Mining frequent itemsets in memory-resident databases

Wim Pijls, Jan C. Bioch
Department of Computer Science, Erasmus University,
P.O.Box 1738, 3000 DR Rotterdam, The Netherlands.
e-mail {*pijls,bioch*}@*few.eur.nl*

Abstract

Due to the present-day memory sizes, a memory-resident database has become a practical option. Consequently, new methods designed to mining in such databases are desirable.
In the case of disk-resident databases, breadth-first search methods are commonly used. We propose a new algorithm, based upon depth-first search in a set-enumeration tree. For memory-resident databases, this method turns out to be superior to breadth-first search.
Keywords Frequent itemsets, Association rules, Datamining

1 Introduction

Finding frequent itemsets in large amounts of data has become a major research issue over the past few years. Most algorithms assume that the database is stored on disk. Main memory in a computer (also called primary memory as opposed to secondary memory denoting disk memory) is getting larger and larger. In a present-day PC, 128 Mb has become a common size. Even 256 Mb is no longer exceptional. Many data sets arising in practice fit into such amounts of memory. Several papers utilize the synthetic data sets from [1], which were proposed there as suitable benchmarks for data mining algorithms. Those data sets also easily fit into the today's main memories. Given the large memory sizes, it makes sense to construct algorithms which exploit the features of memory-resident databases. In this paper, we propose algorithms which are effective under the assumption the database is stored into a two-dimensional array, in which every entry can be retrieved quickly.

Frequent itemsets. Frequent itemsets are needed to formulate association rules, a central task in the present-day practice of data mining and knowledge discovery. Association rules are applied for instance in the analysis of basket data. A stereotypical form of an association rule derived from basket data is: "40% of the customers who buy product X and Y also buy product Z". In algorithms for discovering association rules, the quest for frequent itemsets is the major task, which largely determines the efficiency of the algorithm. Establishing association rules is a straightforward action afterwards. In this paper, we therefore restrict ourselves to finding frequent itemsets.
Mining frequent itemsets was applied first to transaction data. Of course, the application area is much larger. Frequent items are also relevant in insurance data, census data, medical data etc. They also arise as patterns in episodes[10]. We discuss our theory in terms of

Mining Frequent Itemsets in Memory-Resident Databases

Wim Pijls and Jan C. Bioch

ERIM Report Series *Research in Management*	
ERIM Report Series reference number	ERS-2000-53-LIS
Publication	December 2000
Number of pages	9
Email address first author	Pijls@few.eur.nl
URL (electronic version)	
Address	Erasmus Research Institute of Management (ERIM) Rotterdam School of Management / Faculteit Bedrijfskunde Erasmus Universiteit Rotterdam PoBox 1738 3000 DR Rotterdam, The Netherlands Phone: # 31-(0) 10-408 1182 Fax: # 31-(0) 10-408 9640 Email: info@erim.eur.nl Internet: www.erim.eur.nl

Bibliographic data and classifications of all the ERIM reports are also available on the ERIM website:
www.erim.eur.nl

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT

REPORT SERIES
RESEARCH IN MANAGEMENT

BIBLIOGRAPHIC DATA AND CLASSIFICATIONS		
Abstract	Due to the present-day memory sizes, a memory-resident database has become a practical option. Consequently, new methods designed to mining in such databases are desirable. In the case of disk-resident databases, breadth-first search methods are commonly used. We propose a new algorithm, based upon depth-first search in a set-enumeration tree. For memory-resident databases, this method turns out to be superior to breadth-first search.	
Library of Congress Classification (LCC)	5001-6182	Business
	5201-5982	Business Science
	HD 66.2	Data processing
Journal of Economic Literature (JEL)	M	Business Administration and Business Economics
	M 11 R 4	Production Management Transportation Systems
	C89	Data Collection and Data Estimation Methodology; Computer Progrmams: Other
European Business Schools Library Group (EBSLG)	85 A	Business General
	260 K 240 B	Logistics Information Systems Management
	240 F	Data processing
Gemeenschappelijke Onderwerpsontsluiting (GOO)		
Classification GOO	85.00	Bedrijfskunde, Organisatiekunde: algemeen
	85.34 85.20	Logistiek management Bestuurlijke informatie, informatieverzorging
	54.64	Gegevensbanken
Keywords GOO	Bedrijfskunde / Bedrijfseconomie	
	Bedrijfsprocessen, logistiek, management informatiesystemen	
	Data mining, Algoritmen	
Free keywords	Frequent itemsets, association rules, datamining	

Data set

Nrs.	items
1	A B E F
2	B C D
3	A B E F
4	A B C F
5	A B C E F
6	C D E F

Array representation

	A	B	C	D	E	F
1	1	1	0	0	1	1
2	0	1	1	1	0	0
3	1	1	0	0	1	1
4	1	1	1	0	0	1
5	1	1	1	0	1	1
6	0	0	1	1	1	1

Frequent itemsets with *minsup*= 3

support	frequent itemsets
5	B, F
4	A, AB, AF, ABF, BF, C, E, EF
3	AE, ABE, ABEF, AEF, BC, BE, BEF, CF

Figure 1: An example of a data set along with its frequent itemsets.

transaction data. An example of transaction data is shown in Figure 1. There are six transactions (with numbers 1 to 6) and six items (denoted by the letters A to F). A transaction T is said to support an itemset I, if set I is included in T. The support of an itemset I is defined as the number of transactions supporting I. An itemset is called frequent, if the support of I surpasses a given minimum value (the so-called minimum support, abbreviated as *minsup*). In the example of Figure 1 we have $minsup = 3$. As said earlier, the goal of this paper is to develop algorithms intended to discover frequent itemsets in memory-resident databases.

Previous work. The algorithms for finding frequent itemsets may be divided into two kinds: bottom-up and top-down algorithms respectively. In a bottom-up algorithm the candidate itemsets are examined from small to large. On the other hand, a top-down algorithm starts with a large candidate set, which is reduced step by step until a frequent set has been found. Almost every bottom-up algorithm is a variant of Apriori. Some instances of this family are Apriori[1, 2] (the seminal instance of this family), AprioriTid [1, 2], DIC [6], DHP [8], Max-Miner [5]. The latest one searches for maximal frequent item sets whereas the former ones look for all frequent itemsets. A number of top-down instances is presented in [11]. Another top-down instance is Pincer search[7].
In this paper, we only consider bottom-up algorithms. Almost every above bottom-up algorithm applies breadth-first search. However, our focus is on depth-first search. This search method was ignored in data mining so far, since it is not appropriate in the case of disk-resident databases. An algorithm based upon depth-first search will be presented, which surpasses its breadth-first counterpart. Since top-down algorithms aim at finding only maximal frequent items set (finding all frequent itemsets is performed in a subsequent phase), those algorithms are left out of consideration.

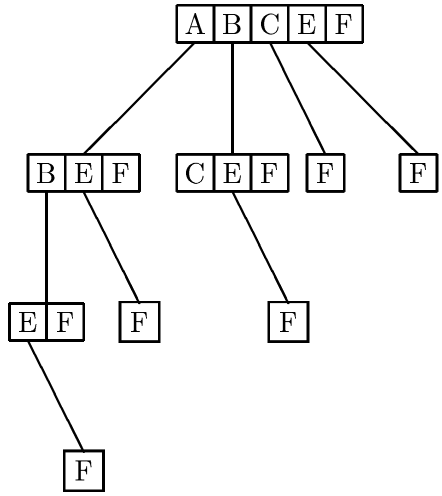

Figure 2: An example of a trie (without support counts).

Overview. In Section 2, we discuss a framework, from which a breadth-first and depth-first search algorithm can be derived. The breadth-first instance is similar to Apriori. In Section 3 we elaborate on the depth-first instance, presenting a new datamining algorithm. Section 4 gives the results of the experiments. Concluding remarks are included in Section 5.

2 A framework based upon a trie

The best-known algorithm for finding frequent patterns is Apriori [1]. In the originating paper, a *hash tree*, a tree with a hash table in each node, was proposed to represent itemsets. We utilize a different data structure which replaces the hash nodes by completely filled arrays with dynamic length. This data structure is equivalent to a trie[3]. In the context of frequent itemsets a trie was applied before in [4]. It stores the full collection of frequent patterns in an efficient and compact way. Figure 2 shows an example of a trie. This trie represents the frequent itemsets of Figure 1 (without mentioning the support counts). Each path from an entry in the root to an entry in another node corresponds to a frequent itemset. So AEF, AE and A are denotations for paths as well as for frequent itemsets. The property that any heading subpattern in a frequent pattern is frequent as well, makes a trie a proper data structure for storing frequent patterns.
The entries (or cells) in a node of a trie are mostly called *buckets*, as is also the case for a hash-tree. Each bucket can be identified with its path to the root and hence with an itemset.

A search framework to look for frequent itemsets is the following code. In the current section, we will discuss later a bread-first instance of this framework. In Section 3, a depth-first instance is discussed.

```
(1) T := any trie of itemsets;
(2) count(T);
(3) stop := false;
(4) while not stop do
(5)    T' := T;
(6)    T := an expansion of T';
(7)    C := T\T';   /* C is the set of candidates */
(6)    count(C);
(8)    if every expansion of T contains
          only infrequent itemsets then stop := true;

(9)  procedure count(C);
(10) for every transaction T do
(11)    for every itemset I ∈ C do
(12)       if T supports I then I.count++;
```

We assume, that it can easily be determined whether the criterion in line 8 is fulfilled. For example, if every expandable bucket in the trie corresponds to an infrequent itemset, the criterion is fulfilled.
The support of an itemset I is commonly stored into the bucket corresponding to I. To count the support of the candidates, a database pass is made. See line 9 through 12. As mentioned before, we assume memory-resident databases. The database is stored into a two-dimensional boolean array. The most obvious storage method is one byte per array entry. However, even one bit per entry turns out to be feasible. Further, we can choose between horizontal and vertical lay-out respectively. In the code of *count* the database is processed transaction by transaction. So, we assume a so-called horizontal lay-out, as opposed to vertical lay-out. In the former the entries of each transaction are stored contiguously, whereas the latter stores the entries of each item contiguously. In line 12 of the above code, backtracking is applied to inspect each path P corresponding to an itemset I of C. Inspecting a path P is aborted as soon as an item i with i outside T is found.

Breadth-First We implemented a breadth-first instance, which is similar to Apriori[1, 2]. Like Apriori, our algorithm builds the trie levelwise: the frequent k-itemsets with $k = 1, 2, 3, \ldots$ are found successively. T is initialized as the collection of all 1-itemsets. After the k-th iteration of the main loop T contains all frequent k-itemsets (itemsets of length k) along with their support. In the $(k+1)$-th iteration, T' is extended to a trie T. The buckets in $T \backslash T'$ (T without T') are new and make up the candidate set C. Each candidate I represents a $(k+1)$-itemset. Our algorithm differs from Apriori in that it has just one data structure to represent itemsets. In the original Apriori version, the candidates are stored into a so-called hash-tree, a data structure equivalent to a trie. Moreover, apart from a hash tree, Apriori maintains a list of frequent itemsets. This list was used to perform a join operation resulting into new candidates and to retrieve subsets of candidates.

3 The depth-first instance

The depth-first instance of the framework proceeds as follows. In a preprocessing step, the support of each single item is counted and the infrequent items are eliminated. Let the frequent items be denoted by $i_1, i_2, \ldots, i_n$. Next, the following code is executed.

(1) $T :=$ *the trie including only bucket* i_n*;*
(2) **for** $m := n - 1$ **downto** *1* **do**
(3) $\quad T' := T$*;*
(4) $\quad T := T'$ *with* i_m *added to the left and*
$\qquad\quad$ *a copy of* T' *appended to* i_m*;*
(5) $\quad C := T \backslash T'$ *(=the subtrie rooted in* i_m*);*
(6) $\quad$ *count*(C)*;*
(7) $\quad$ *delete the infrequent itemsets from* T*;*

On termination, T exactly contains the frequent itemsets. How the algorithm works, is illustrated in Figure 3 using the data set of Figure 1. The single items surpassing the minimum support are $i_1 = A, i_2 = B, i_3 = C, i_4 = E$ and $i_5 = F$. Figure 3 shows the shape of T composed in each iteration of the while loop. Also the infrequent itemsets to be deleted at the end of each iteration are mentioned. At the start of the m-th iteration, the root of trie T consists of the 1-itemsets $i_{m+1}, \ldots, i_n$. (We denote a 1-itemset by the name of the single item.) By the statement in line 3, this trie may also be referred to as T'. A new trie T is composed including the buckets $i_m, i_{m+1}, \ldots, i_n$ in the root and a copy of T' (the former value of T) appended to i_m. The new candidate set C makes up a subtrie consisting of i_m and a copy of T' appended to i_m. In Figure 3, the candidate set C is in the left part of each trie and this set is drawn in bold. Notice that the final shape of the trie (after deleting infrequent itemsets) agrees with Figure 2.

The number of iterations in the *for* loop is equal to the number of frequent 1-itemsets. When the *for* loop parameter is equal to m, the column corresponding to the m-th item in the database array is passed. Consequently the new algorithm is not tractable, if the database under consideration is not in memory.

For the search framework of Section 2, the performance of an instance may be measured by the number of *inspections* into the two-dimensional array A representing the database. Consider the procedure *count* in lines 9 through 12 in the code of Section 2. Given a transaction T and an candidate itemset I, the path corresponding to I is walked through as long as this path contains items included in T. For the items i on this path, the cell at the intersection of row T and column i in array A is inspected. Such an action counts as one inspection.
In the breadth-first version, the trie is built up layer by layer, as discussed in Section 2. A new layer of buckets means a new set C of candidates. Each ancestor bucket of a new candidate I in C corresponds to a frequent subset of I. When the procedure *count* is executed for a new layer, each ancestor bucket of any candidate I is visited as many times as its count value. Each such visit entails one inspection. In fact, in each ancestor bucket of any candidate I, the support is re-counted. In the depth-first instance on the other hand, every bucket or itemset I is counted once, and a bucket is not re-visited after completion of its support count. Hence, we will see in Section 4, that depth-first has considerably fewer inspections than breadth-first.
The depth-first version has a preprocessing step which counts the support for each single item, before executing the above code. After the preprocessing step, the items may be

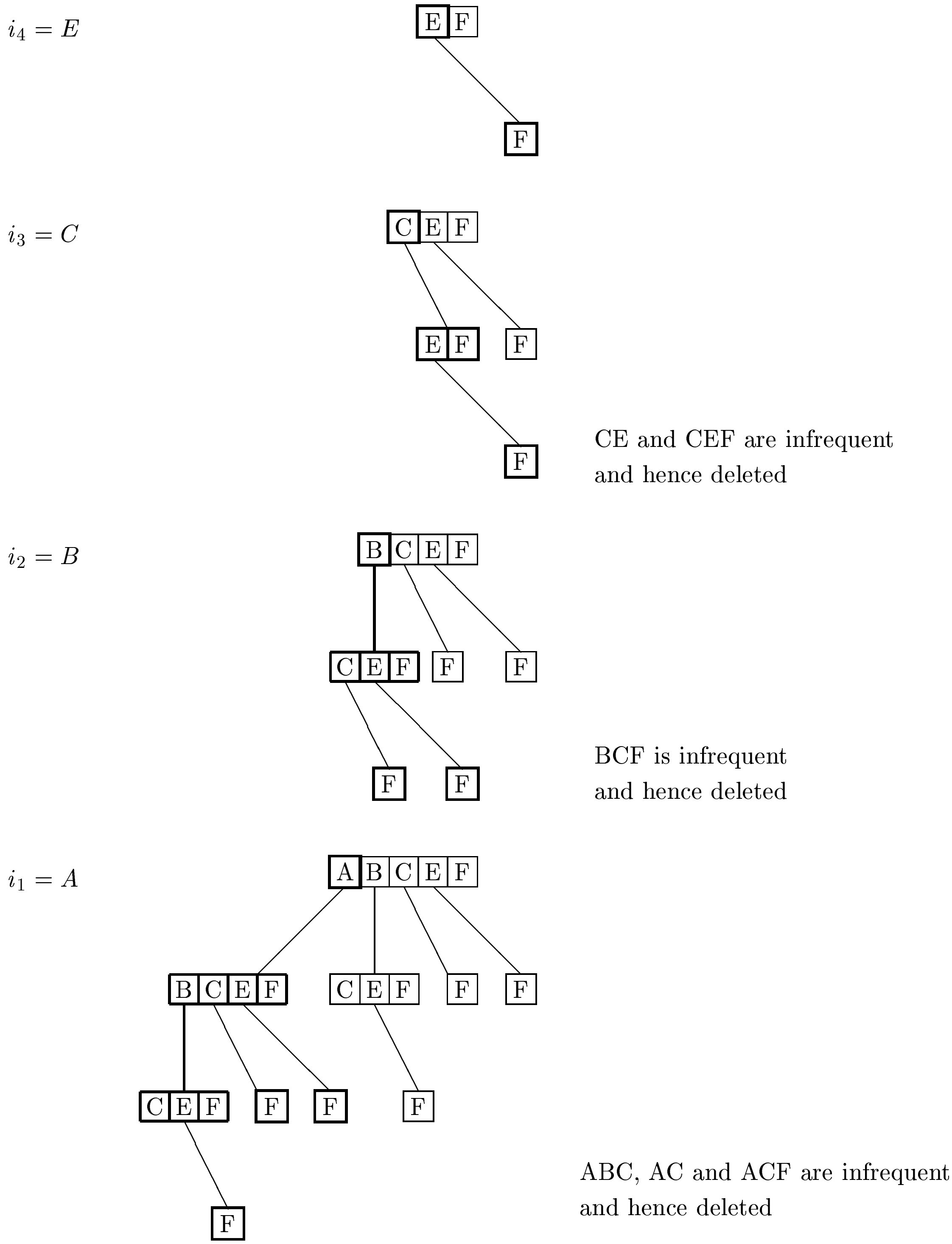

Figure 3: Illustrating the Depth-first algorithm

re-ordered. The most favorable execution time is achieved, if we order the items from right to left by decreasing frequency. This result can be explained as follows. If a bucket in the trie has a support s, then each child bucket is visited s times. In a child bucket the same phenomenon holds. Therefore, it is better to have low support at the top of the deeper side (to the left) of the trie and hence, high support at the top of the shallow part (to the right).

4 Experimental work

We applied the depth-first (DF) and the breadth-first (BF) algorithm to the synthetic databases simulating retail transaction data, as described in [1, 2]. These datasets are used as benchmarks in most papers dealing with frequent itemsets. The parameters for generating a synthetic database are the number of transactions D (in thousands), the average transaction size T and the average length I of maximal frequent itemsets. The number of maximal frequent itemsets was set at $L = 2000$ and the number of items was set at $N = 1000$, following the design in [1, 2, 8, 9]. The experiments were conducted at a Pentium-1 machine with 128 Mb memory at 166Mz, running Windows NT. The programs were developed under the Borland C++ 5.02 environment, but are also usable with the GNU C++ compiler.

We found out that execution times with bitwise storage hardly differ from times with bytewise storage. For reasons of space efficiency, each program involved in the experiments below applies bitwise storage of the two-dimensional database array.
The execution times in seconds for four data sets defined in [1] are displayed in the tables. For the data sets D100T20I6 and D100T20I4, four *minsup*-values (ranging from .5% to 2%) are applied. See Figure 4.

		.5%	1%	1.5%	2%
D100T20I6	DF	273	121	78	65
	BF	561	191	134	98
D100T20I4	DF	219	128	88	72
	BF	455	260	142	104

Figure 4: Examining $D = 100$ and $T = 20$ with $I = 6$ and $I = 4$.

The outcome of the examination of the sets D100T10I4 and D100T5I2 is shown in Figure 5. Those sets do not contain any frequent k-itemsets with $k > 1$ if a *minsup* value $> 1\%$ is taken. Therefore, *minsup* values $> 1\%$ have been omitted in the table.

		.25%	.5%	.75%	1%
D100T10I4	DF	99	71	54	47
	BF	277	165	88	69
D100T5I2	DF	51	37	29	25
	BF	109	66	54	39

Figure 5: Examining D100T10I4 and D100T5I2.

In both above figures, the depth-first algorithm turns out to be superior to breadth-first. As the *minsup* is lower (and hence the workload is greater), the discrepancy is larger. The transition from depth-first to breadth-first reduces the execution time to about 50% in case of low *minsup* values and to about 70% in case of higher *minsup* values.
In Section 3, we introduced the number of inspections as a standard for the performance of the instances of the framework. We measured the number of inspections during the aforementioned experiments. The results are shown in Figure 6. Each value denotes the number of inspections expressed in millions. Clearly, depth-first surpasses breadth-first. This has already been argued theoretically in Section 3.

		.5%	1%	1.5%	2%
D100T20I6	DF	804	427	301	234
	BF	1036	670	503	401
D100T20I4	DF	731	464	315	251
	BF	956	701	521	429
		.25%	.50%	.75%	1%
D100T10I4	DF	295	226	172	142
	BF	577	427	327	282
D100T5I2	DF	139	86	53	35
	BF	281	206	161	139

Figure 6: The number of array inspections.

5 Concluding remarks

Since main memories are very large nowadays and bitwise storage of the boolean database array turns out to be feasible, one may assume memory-resident data-bases in many practical cases. We have applied a depth-first algorithm to memory-resident databases. (Applying this algorithm to a database on disk is not a practical option.) It evidently surpasses the classical algorithms adapted to memory-resident databases. Moreover, it is transparent and easy to implement. Hence, it should be considered promising.

References

[1] R. Agrawal, and R. Srikant, *Fast Algorithms for Mining Association Rules*, Proceedings of the 20th Int'l Conference on Very Large Databases, Santiago, Chili, September 1994.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen and A.I. Verkamo, *Fast Discovery of Association Rules*, Chapter 12 in: U.M Fayyad et al. (eds.), Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, pp. 307-328, 1996.

[3] A. V. Aho, J.E. Hopcroft and J.D. Ullman, *Data Structures and Algorithms*, pp. 163-169, ISBN 0-201-00023-7, Addison-Wesley Publishing Company, 1983.

[4] A. Amir, R. Feldman and R. Kashi, *A New and Versatile Method for Association Generation*, in: Principles of Data Mining and Knowledge Discovery, Proceedings of the First European Symposium, (PKDD'97) Trondheim Norway, pp. 221-231, 1997.

[5] R. J. Bayardo Jr., *Efficiently Mining Long Patterns from Databases*, in Proceedings of the ACM SIGMOD Conference on Management of Data, Seattle, pp. 85-93, 1998.

[6] S. Brin, R. Motwani, J. Ullman and S. Tsur, *Dynamic Itemset Counting and Implication Rules for Market Basket Data* in: Proceedings of the 1997 SIGMOD Conference of Management of Data, pp.255-264.

[7] D. Lin and Z.M. Kedem, *Pincer Search, A New Algorithm for Discovering the Maximum Frequent Set*, in: Proceedings of the Sixth European on Extending Database Technology, pp. 105-119, 1998.

[8] J.S. Park, M.-S. Chen and P.S. Yu, *An Effective Hash Based Algorithm for Mining Association Rules*, in: Proceedings of the 1995 SIGMOD Conference on the Management of Data, pp. 175-186.

[9] A. Savasere, E. Omiecinsky and S. Navathe, *An efficient algorithm for Mining Association rules in Large Databases*, in: Proceedings of the 21st Conference on Very Large Databases. pp. 432-444. 1995.

[10] H. Toivonen, *Discovery of frequent patterns in large data collections,* Ph.D. Thesis. Report A-1996-5, University of Helsinki, Department of Computer Science, November 1996.

[11] M.J. Zaki, S. Parthasarathy, M. Ogihara and W. Li, *New Algorithms for Fast Discovery of Association Rules*, in: Proceedings of the Third International Conference on Knowledge Discovery in Databases and Data Mining, pp. 283-286, 1997.

ERASMUS **R**ESEARCH **I**NSTITUTE OF **M**ANAGEMENT

REPORT SERIES
RESEARCH IN MANAGEMENT

Publications in the Report Series Research* in Management

Impact of the Employee Communication and Perceived External Prestige on Organizational Identification
Ale Smidts, Cees B.M. van Riel & Ad Th.H. Pruyn
ERS-2000-01-MKT

Critical Complexities, from marginal paradigms to learning networks
Slawomir Magala
ERS-2000-02-ORG

Forecasting Market Shares from Models for Sales
Dennis Fok & Philip Hans Franses
ERS-2000-03-MKT

A Greedy Heuristic for a Three-Level Multi-Period Single-Sourcing Problem
H. Edwin Romeijn & Dolores Romero Morales
ERS-2000-04-LIS

Integer Constraints for Train Series Connections
Rob A. Zuidwijk & Leo G. Kroon
ERS-2000-05-LIS

Competitive Exception Learning Using Fuzzy Frequency Distribution
W-M. van den Bergh & J. van den Berg
ERS-2000-06-LIS

Start-Up Capital: Differences Between Male and Female Entrepreneurs, 'Does Gender Matter?'
Ingrid Verheul & Roy Thurik
ERS-2000-07-STR

The Effect of Relational Constructs on Relationship Performance: Does Duration Matter?
Peter C. Verhoef, Philip Hans Franses & Janny C. Hoekstra
ERS-2000-08-MKT

Marketing Cooperatives and Financial Structure: a Transaction Costs Economics Analysis
George W.J. Hendrikse & Cees P. Veerman
ERS-2000-09-ORG

---

* ERIM Research Programs:
LIS Business Processes, Logistics and Information Systems
ORG Organizing for Performance
MKT Decision Making in Marketing Management
F&A Financial Decision Making and Accounting
STR Strategic Renewal and the Dynamics of Firms, Networks and Industries

A Marketing Co-operative as a System of Attributes: A case study of VTN/The Greenery International BV,
Jos Bijman, George Hendrikse & Cees Veerman
ERS-2000-10-ORG

Evaluating Style Analysis
Frans A. De Roon, Theo E. Nijman & Jenke R. Ter Horst
ERS-2000-11-F&A

From Skews to a Skewed-t: Modelling option-implied returns by a skewed Student-t
Cyriel de Jong & Ronald Huisman
ERS-2000-12-F&A

Marketing Co-operatives: An Incomplete Contracting Perspective
George W.J. Hendrikse & Cees P. Veerman
ERS-2000-13– ORG

Models and Algorithms for Integration of Vehicle and Crew Scheduling
Richard Freling, Dennis Huisman & Albert P.M. Wagelmans
ERS-2000-14-LIS

Ownership Structure in Agrifood Chains: The Marketing Cooperative
George W.J. Hendrikse & W.J.J. (Jos) Bijman
ERS-2000-15-ORG

Managing Knowledge in a Distributed Decision Making Context: The Way Forward for Decision Support Systems
Sajda Qureshi & Vlatka Hlupic
ERS-2000-16-LIS

Organizational Change and Vested Interests
George W.J. Hendrikse
ERS-2000-17-ORG

Strategies, Uncertainty and Performance of Small Business Startups
Marco van Gelderen, Michael Frese & Roy Thurik
ERS-2000-18-STR

Creation of Managerial Capabilities through Managerial Knowledge Integration: a Competence-Based Perspective
Frans A.J. van den Bosch & Raymond van Wijk
ERS-2000-19-STR

Adaptiveness in Virtual Teams: Organisational Challenges and Research Direction
Sajda Qureshi & Doug Vogel
ERS-2000-20-LIS

Currency Hedging for International Stock Portfolios: A General Approach
Frans A. de Roon, Theo E. Nijman & Bas J.M. Werker
ERS-2000-21-F&A

Transition Processes towards Internal Networks: Differential Paces of Change and Effects on Knowledge Flows at Rabobank Group
Raymond A. van Wijk & Frans A.J. van den Bosch
ERS-2000-22-STR

Assessment of Sustainable Development: a Novel Approach using Fuzzy Set Theory
A.M.G. Cornelissen, J. van den Berg, W.J. Koops, M. Grossman & H.M.J. Udo
ERS-2000-23-LIS

Creating the N-Form Corporation as a Managerial Competence
Raymond vanWijk & Frans A.J. van den Bosch
ERS-2000-24-STR

Competition and Market Dynamics on the Russian Deposits Market
Piet-Hein Admiraal & Martin A. Carree
ERS-2000-25-STR

Interest and Hazard Rates of Russian Saving Banks
Martin A. Carree
ERS-2000-26-STR

The Evolution of the Russian Saving Bank Sector during the Transition Era
Martin A. Carree
ERS-2000-27-STR

Is Polder-Type Governance Good for You? Laissez-Faire Intervention, Wage Restraint, And Dutch Steel
Hans Schenk
ERS-2000-28-ORG

Foundations of a Theory of Social Forms
László Pólos, Michael T. Hannan & Glenn R. Carroll
ERS-2000-29-ORG

Reasoning with partial Knowledge
László Pólos & Michael T. Hannan
ERS-2000-30-ORG

Applying an Integrated Approach to Vehicle and Crew Scheduling in Practice
Richard Freling, Dennis Huisman & Albert P.M. Wagelmans
ERS-2000-31-LIS

Informants in Organizational Marketing Research: How Many, Who, and How to Aggregate Response?
Gerrit H. van Bruggen, Gary L. Lilien & Manish Kacker
ERS-2000-32-MKT

The Powerful Triangle of Marketing Data, Managerial Judgment, and Marketing Management Support Systems
Gerrit H. van Bruggen, Ale Smidts & Berend Wierenga
ERS-2000-33-MKT

The Strawberry Growth Underneath the Nettle: The Emergence of Entrepreneurs in China
Barbara Krug & Lászlo Pólós
ERS-2000-34-ORG

Consumer Perception and Evaluation of Waiting Time: A Field Experiment
Gerrit Antonides, Peter C. Verhoef & Marcel van Aalst
ERS-2000-35-MKT

Trading Virtual Legacies
Slawomir Magala
ERS-2000-36-ORG

Broker Positions in Task-Specific Knowledge Networks: Effects on Perceived Performance and Role Stressors in an Account Management System
David Dekker, Frans Stokman & Philip Hans Franses
ERS-2000-37-MKT

An NPV and AC analysis of a stochastic inventory system with joint manufacturing and remanufacturing
Erwin van der Laan
ERS-2000-38-LIS

Generalizing Refinement Operators to Learn Prenex Conjunctive Normal Forms
Shan-Hwei Nienhuys-Cheng, Wim Van Laer, Jan Ramon & Luc De Raedt
ERS-2000-39-LIS

Classification and Target Group Selection bases upon Frequent Patterns
Wim Pijls & Rob Potharst
ERS-2000-40-LIS

New Entrants versus Incumbents in the Emerging On-Line Financial Services Complex
Manuel Hensmans, Frans A.J. van den Bosch & Henk W. Volberda
ERS-2000-41-STR

Modeling Unobserved Consideration Sets for Household Panel Data
Erjen van Nierop, Richard Paap, Bart Bronnenberg, Philip Hans Franses & Michel Wedel
ERS-2000-42-MKT

The Interdependence between Political and Economic Entrepeneurship
ERS-2000-43-ORG
Barbara Krug

Ties that bind: The Emergence of Entrepreneurs in China
Barbara Krug
ERS-2000-44-ORG

What's New about the New Economy? Sources of Growth in the Managed and Entrepreneurial Economies
David B. Audretsch and A. Roy Thurik
ERS-2000-45-STR

Human Resource Management and Performance: Lessons from the Netherlands
Paul Boselie, Jaap Paauwe & Paul Jansen
ERS-2000-46-ORG

Average Costs versus Net Present Value: a Comparison for Multi-Source Inventory Models
Erwin van der Laan & Ruud Teunter
ERS-2000-47-LIS

A Managerial Perspective on the Logic of Increasing Returns
Erik den Hartigh, Fred Langerak & Harry Commandeur
ERS-2000-48-MKT

Fuzzy Modeling of Client Preference in Data-Rich Marketing Environments
Magne Setnes & Uzay Kaymak
ERS-2000-49-LIS

The Mediating Effect of NPD-Activities and NPD-Performance on the Relationship between Market Orientation and Organizational Performance
Fred Langerak, Erik Jan Hultink & Henry S.J. Robben
ERS-2000-50-MKT

Extended Fuzzy Clustering Algorithms
Uzay Kaymak & Magne Setnes
ERS-2000-51-LIS

Sensemaking from actions: Deriving organization members' means and ends from their day-to-day behavior
ERS-2000-52-MKT
Johan van Rekom, Cees B.M. van Riel & Berend Wierenga