Multi-objective Optimization Methods for Allocation and Prediction

Multi-objective Optimization Methods for Allocation and Prediction

Multi-objectieve optimalisatiemethoden voor allocatie en voorspelling

Thesis

to obtain the degree of Doctor from the
Erasmus University Rotterdam
by command of the
rector magnificus

Prof.dr. R.C.M.E. Engels

and in accordance with the decision of the Doctorate Board

The public defense shall be held on

Thursday 9 May 2019 at 15:30 hrs

by

Qing Chuan Ye
born in Wageningen, the Netherlands.

Erasmus University Rotterdam

Doctoral Committee

Promotor:	Prof.dr.ir. R. Dekker
Other members:	Prof.dr. A.P.M. Wagelmans Dr. N. Agatz Prof.dr. U. Kaymak
Copromotor:	Dr. Y. Zhang

Erasmus Research Institute of Management - ERIM
The joint research institute of the Rotterdam School of Management (RSM)
and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam
Internet: http://www.erim.eur.nl

ERIM Electronic Series Portal: http://repub.eur.nl/

ERIM PhD Series in Research in Management, 460
ERIM reference number: EPS-2019-460-LIS
ISBN 978-90-5892-539-8

SIKS Dissertation Series No. 2019-10
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Design: PanArt, www.panart.nl

This publication (cover and interior) is printed by Tuijtel on recycled paper, BalanceSilk®.
The ink used is produced from renewable resources and alcohol free fountain solution.
Certifications for the paper and the printing production process: Recycle, EU Ecolabel, FSC®, ISO14001.
More info: www.tuijtel.com

Acknowledgments

First of all, I would like to thank my supervisors, Yingqian Zhang and Rommert Dekker, for the opportunity they gave me to do a PhD, and for guiding and assisting me throughout the PhD trajectory. Rommert, thank you for all the interesting conversations and the many insights into what it takes to be a researcher. Despite your busy schedule, you always found some time for casual chitchat, which made meetings more light-hearted. Yingqian, thank you for all the time spent on meetings, discussions, email correspondence, and of course some small talk now and then. I enjoyed our brainstorming sessions and I learned a lot about a myriad of topics from different expertises throughout the course of these years. Despite having been a teaching assistant for quite a few years at the beginning of my PhD, I had never stood in front of a classroom and given a lecture before. That is, until you needed a replacement for a lecture. Even though I have to admit I was very nervous at first, I ended up enjoying giving lectures. I really appreciate all the opportunities you have given me and your patience throughout these years.

I would also like to thank Prof.dr. Albert Wagelmans, Dr. Niels Agatz and Prof.dr. Uzay Kaymak for being part of my inner committee and giving constructive feedback on my thesis. Furthermore, I would like to thank Prof.dr. Kevin Tierney and Prof.dr. Patrick de Causmaecker for being part of my committee and joining the opposition in the thesis defense.

I would like to give special thanks to Wim Pijls for giving me the opportunity to be a teaching assistant, which made me realize that I enjoy teaching, eventually resulting in my decision to pursue a PhD. I am grateful for your kindness. Just before starting my PhD, I had the privilege to work together with Emiel Maasland

and Tommi Tervonen on a project that eventually resulted in a spin-off company. I enjoyed the time working together and getting a glimpse of consulting and software development. In addition, I would like to thank Sicco Verwer for the cooperation on various papers and the hospitality.

The past few years would not have been as enjoyable if it were not for my fellow colleagues. Judith, thank you for reaching out to me on my first day and introducing me to our colleagues. I enjoyed the talks we had when you dropped by from time to time and your enthusiasm. Rutger, thank you for the discussions on various topics, both research and non-research related. I greatly appreciate all the proofreading and feedback, and the work you put in to help your colleagues with the final steps of their thesis. Kevin, thank you for taking over the responsibility of organizing the game nights and for facilitating various other activities and helping everyone out when you can. I would also like to thank the rest of my colleagues, who all helped to make it a very enjoyable experience: Alexander, Bart, Chiel, Evelot, Frederik, Harwin, Kim, Kristiaan, Marieke, Mathijn, Nemanja, Nick, Paul, Remy, Rowan, Sha, Thomas, Thomas, Tim, Twan, Weina, Willem, Zahra. Thanks for the great times, including, but of course not limited to, lunch breaks, special events, and game nights!

Finally, I would like to thank my family and friends for their support over all these years. Kevin ten Haaf and Ruben Janssen deserve my gratitude for being my paranymphs. And last but definitely not least: Ann, thank you so much for being by my side and for your support. When I started this journey, our journey together also started soon after. However, as this journey will come to an end, ours will continue on.

Charlie Ye
Rotterdam, March 2019

Table of Contents

Chapter 1

Introduction

Auctions are a common occurrence in everyday life. With the rise of the internet online auctions have become increasingly popular, as they are easily accessible to everyone. An auction is a way of selling items, which can be goods or services, that are put up for bid by an auctioneer. Participants, or bidders, can place bids on an item to indicate the price they are willing to pay for an item. The higher the bid, the better the chance the agent will win. Most auctions consist of many items. These items can be auctioned off one after another, or they can be put up on auction all at the same time. Some auctions take place every day, or even multiple times in a day, whereas other only occur once in a while. Some well known examples of auctions are commodities auctions, which usually occur daily, like flower and fish wholesale auctions, and auctions of luxury items, which occur less often, such as fine art and antiques. Online auctions have also given rise to new kinds of auctions, like vacation auctions and advertisement auctions. Besides auctions for obtaining goods, one can also put up a service for bid on which agents can bid their desired compensation. This is also known as a reverse auction. In this case it follows that the lower the bid, the higher the chance the agent will win. This type of auction is often used in procurement, and, what has been on the rise recently, the sharing economy.

In this thesis, we will focus on two different aspects of auctions. First, we consider the task allocation problem. This comes into play at the end of an auction, when participants have submitted all their bids, and the auctioneer needs to determine

which item or task will be assigned to which bidder. Second, we consider the auction design. Auction design already comes into play before the auction even starts. The auctioneer needs to decide how the auction will work, e.g. how the items are presented, and what the starting prices will be. Auction design is very important as it influences how the bidders will bid, and therefore has a big impact on the outcome.

1.1 Task allocation problem

The task allocation problem, or assignment problem, is a combinatorial optimization problem that is well studied and has applications in many fields. In a task allocation problem, there are a number of *agents* and a number of *tasks*. The tasks can usually be done by any of the agents, but an agent might have a limit on the number of tasks it can perform. Letting an agent perform a task incurs a cost, which differs for each combination of tasks and agents. The challenge is to determine which task(s) should be done by which agent, in such a way that the sum of costs is minimized. This problem has been widely studied and can be solved reasonably fast.

However, in a repeated auction setting this minimum cost solution might not be the best solution in the long run. We only know that this solution is optimal for a single task allocation problem. In repeated auctions it can happen that the set of agents is different in between auctions. Agents will participate in auctions when they think they will benefit from it. They put in their time and effort in the hopes of profiting off of it, e.g., winning a task. When an agent participates many times in the same auction, but rarely wins something, the agent can think they are wasting their time and effort, which can be put to better use. Therefore, they might choose to refrain from participating in the auction again. They could start participating in a different auction from a different auctioneer, or even stop participating at all when the bidder is not reliant on these auctions, e.g. people participating in online auctions for luxury items. This results in a smaller pool of participants, resulting in fewer bids, which means that the allocation will have less flexibility. Ultimately this could lead to a situation where there are only a few participants who can dictate the allocation because they are the only agents left, and who may drive up the costs as there is no competition left. This situation is undesirable for the auctioneer.

Therefore, the auctioneer may be interested in another criterion besides only minimizing costs when allocating tasks to agents. To avoid dissatisfaction among agents, the auctioneer may opt to incorporate *fairness* in the task allocation. There are multiple ways to define fairness, and the exact definition of fairness is up to the auctioneer to determine what is considered fair in their application.

We study the fair task allocation problem in the context of an actual transportation situation in the port of Rotterdam in the Netherlands in Chapter 2, and we study its effects in repeated task allocations in Chapter 3, in which we take into account the development of the agents' participation behavior.

1.2 Auction design optimization

Auction design is very important to the way an auction will eventually play out (Klemperer 2002). A different design for an auction with the exact same items and participating agents may result in a completely different outcome. Not only does the design influence the final allocation, but it can also influence the bids that agents submit. An example of an aspect the auctioneer needs to take into account is whether the bids are made public, i.e., all bidders may know the bids of other bidders (open bid), or whether the bids will be secret (sealed bid). Another example would be whether bidders should submit bids that exceed other bids until no participant would like to make a better bid (English auction), or whether the auctioneer should set a price no one is willing to pay for it, and lower it until there is a participant who is prepared to accept the proposed price (Dutch auction). The starting price of an item, and the order of items being shown in the auction, can have an impact on the performance of the auction as well. Therefore, it is important to design an auction in such a way that it optimizes the desired outcome, whether this would be fetching the highest price, or selling as many items as possible.

Designing such an auction is difficult as this requires a lot of knowledge of the specific auction and its properties, and the items that are being auctioned. Hence, this has been primarily the work of experts who have had many years of experience, know a lot about the items at hand, and know how and when to auction them. However, one expert is not the other and they each have their own ideas of what is

important to an auction and what is not. It can also happen that experts do not notice certain aspects that are present in historical auctions and therefore will not act upon. Hence, it is difficult even for experts to design an auction that optimizes the desired outcome. However, with the rise of digitization and the storage of information on conducted auctions, there is a trove of data available on historical auctions. These historical auctions contain a lot of information that can be useful for designing future auctions. In order to obtain this information we make use of *machine learning* techniques.

Machine learning can be used in data analytics to discover possible patterns in data. Historical data can contain information on relationships and trends that might not have been noticed before. Therefore, hidden insights can be uncovered by examining historical data. With the help of machine learning techniques, one can construct models and algorithms that can learn relations and patterns within historical data. Thereafter, these models can be used to predict the outcome of a new instance that is similar to the situation in the examined historical instances. There are two main categories in machine learning, supervised and unsupervised learning. In supervised learning example inputs and the accompanying outcomes are provided to the computer, from which a general rule needs to be learned that maps the inputs to the outcomes. In unsupervised learning no outcomes are provided to the computer. Hence, the computer needs to find structure in the given input by itself.

We make use of supervised learning for regression and classification problems and use these in combination with mathematical optimization models in order to learn what made an item perform well in historical auctions. In Chapter 4 we learn regression models and map these to integer linear programs, which we can optimize by using an existing solver. We then apply this technique in Chapter 5 to help construct a decision support system to aid experts with auction design.

1.3 Thesis outline

In this dissertation we use a combination of techniques from operations research and computer science to tackle different aspects of an auction. It consists of two main

parts. In Chapters 2 and 3 we focus mainly on the task allocation part of the auction. The inspiration for this problem comes from an actual transportation situation in the port of Rotterdam in the Netherlands, where there is a huge increase in inter-terminal transport due to the expansion of the port. We propose a sustainable transportation system that makes use of existing trucks at the port and uses an auction to collect bids and assign tasks to trucks. As this auction would be repeated every day, and participants are free to come and go whenever they would like in this system, it is important to keep the truck companies interested in participating in order to have competition. Therefore, we investigate the effect of adding fairness into the allocation and how it fares with regard to the traditional minimum cost solution. In Chapters 4 and 5 we focus on the auction design. We use a combination of machine learning techniques and mathematical programming models to derive information from historical data that helps with determining the design parameters for a new auction of similar items. Through experiments and a case study on historical data of auctions obtained from an online industrial auction company we show the performance of our methods.

The chapters of this dissertation are based on papers that have been published in or have been submitted to peer-reviewed journals. As a result, all chapters can be read independently from each other. We will outline the topic of each chapter in more detail below.

Chapter 2 considers a fair task allocation problem in transportation where an optimal allocation has low cost and distributes tasks as evenly as possible among heterogeneous participants who have different capacities and costs to execute tasks. We analyze and solve it in two parts using two novel polynomial-time algorithms. Furthermore, we conduct an extensive set of experiments to investigate the trade-off between cost minimization and fairness.

In Chapter 3 we study the effect of the fair task allocation in a repeated setting where we conduct multiple rounds of the auction. We investigate how the allocation influences the agents' decision to participate. The participation behavior of agents is modeled in two different ways, namely using prospect theory, and using a fuzzy connective. Simulations are used to study how agents' participation affects the outcomes throughout the rounds. We compare two task allocation algorithms, the traditional minimum cost allocation, and the fair task allocation.

In Chapter 4 we demonstrate how to apply machine learning techniques to solve the optimal ordering problem in sequential auctions. We learn regression models based on historical auctions, which are subsequently used to predict the expected value of orderings for new auctions. Given the regression models, we introduce a novel white-box approach that maps learned regression models to integer linear programs (ILP), which can then be solved by any ILP-solver. We show that the internal structure of regression models can be efficiently evaluated inside an ILP solver for optimization purposes.

A case study using data from an online industrial auction company can be found in Chapter 5. We propose a method that builds a decision tree to predict how well an auction will perform in terms of a performance indicator, which is the ratio between the selling price and the estimated price of items. We show how to learn a classification tree by combining multiple objectives using integer programming, given the specific user's needs. In addition, we compare the performance of our method with traditional classification tree algorithms, and demonstrate the flexibility of our method.

Finally, in Chapter 6 we conclude the main findings of this dissertation.

1.4 Clarification of contribution

The chapters of this dissertation are based on papers that are the result of a collaboration between myself, my promotor, my co-promotor and various other authors. For each chapter, the reference to the publication and the contribution of each author are given below.

Chapter 2 The research for this chapter has been conducted by the first author in close cooperation with dr. Yingqian Zhang, under the supervision of prof.dr.ir. Rommert Dekker. This chapter spawned from an inter-terminal transport (ITT) project which was part of a collaboration between the Erasmus University Rotterdam and TU Delft. This chapter is based on Ye et al. (2017a).

Chapter 3 The research for this chapter has been conducted by the first author in close cooperation with dr. Yingqian Zhang. Prof.dr.ir. Uzay Kaymak con-

tributed in defining the problem of the fuzzy connective and aided in the review process. This chapter is based on Ye and Zhang (2016) and Ye et al. (2017b).

Chapter 4 The research for this chapter has been conducted by dr. Sicco Verwer and dr. Yingqian Zhang in close cooperation with myself. I was mainly involved in the construction of mathematical models and algorithms, and the running of experiments. This chapter is based on Verwer et al. (2017).

Chapter 5 The research for this chapter has been conducted by the first author in close cooperation with dr. Yingqian Zhang and dr. Sicco Verwer. It follows a collaboration with an online industrial auction company which provided the data. This chapter is based on a working paper.

Chapter 2

Fair task allocation in transportation [1]

2.1 Introduction

Traditionally, optimization of task allocation problems considered only the costs involved in the allocation. However, there has been in recent years more attention to cases where cost should not always be the sole consideration (Campbell et al. 2008). There are circumstances when other criteria need to be taken into account as well during the decision making process. *Fairness* has been considered as one of the important additional criteria in many application domains (Ogryczak et al. 2005, Gopinathan and Li 2011, Bertsimas et al. 2012). Although there is no common definition for the term, there are two fairness criteria that are often used in the literature: the Nash bargaining criterion and the Rawlsian maximin criterion. The former is based on Nash's four axioms of pareto optimality, independency of irrelevant alternatives, symmetry, and invariance to affine transformations or equivalent utility representations (Nash 1950). The latter is based on Rawls' two principles of justice (Rawls 1971). Rawls' maximin criterion maximizes the welfare level of the

[1] This chapter is based on Ye et al. (2017a).

worst-off group member and has therefore been used in allocation problems (Jaffe 1981, Kumar and Kleinberg 2000).

In this chapter, we study task allocation problems in which we take fairness into account in addition to the standard minimum cost criterion. This work was inspired by an actual transportation situation in the port of Rotterdam in the Netherlands. The increase in the number of container terminals in said port will result in a huge increase in inter-terminal transport (ITT). The port authority invited a team of researchers to investigate a sustainable transportation system, called an *asset light solution*, in which trucks that were already present in the port could execute open jobs. The main idea behind this system is that trucks that come from the hinterland to drop off or pick up containers often have spare time in between tasks. Usually, trucks are scheduled to do several jobs to and from various terminals in the port in one day. There may be large gaps between these jobs during which time the truck would be idle due to the nature of the jobs that truck companies agree to do. Terminals could take advantage of these idle trucks by providing them with jobs that they can perform within the port while waiting for their next scheduled job. The trucks will be compensated for these jobs. The compensation from the terminals to the trucking companies would be large enough to cover the costs that the companies would incur. However, the compensation should be less than the costs of purchasing and maintaining, or even renting the vehicles dedicated for such jobs. This way, the trucking companies gain additional income while the terminals save money by using readily available resources. Furthermore, because the utilization rate of existing trucks becomes higher and no new trucks are needed, this is a more durable approach to meeting the transport need within the port.

To realize such a task allocation, terminals need to be informed of the individual schedules of the different trucking companies. This poses a hurdle because getting such information is expensive and the trucking companies may be reluctant to share their entire schedules. One way to circumvent this difficulty is to use *auctions* as a means to collect information from different parties. Auctions have become increasingly popular for allocating resources among individual players in many application domains, such as in spectrum auctions (Cramton 2002), health care (Smits and Janssen 2008), industrial procurement (Gallien and Wein 2005a, Bichler et al. 2006) and logistics (Sheffi 2004, Ball et al. 2006). In the auction for our trucking task

allocation case, we assume that all terminals together act as an auctioneer and they announce a set of available jobs. Different trucking companies can bid for those jobs, depending on their idle trucks at specific times. Given the bids of different companies, the terminals then decide on a best allocation of jobs to companies.[2] Because there are ITT movements every day that need to be executed, this task allocation activity would be held daily. Some studies have shown that greedily minimizing cost does not fare well with repeated auctions. Participants could experience starvation in the long run, which will reduce their incentive to continue participating in the allocation activity (Gopinathan and Li 2011). Furthermore, repeated auctions may affect the relationships between the auctioneer and bidders, which in turn affects the latter's way of bidding (Jap and Haruvy 2008). To prevent these adverse effects, we should not only look at optimizing the costs in the task allocation, but we should also incorporate fairness in the task allocation that results from the auctions. We do this by reassuring that all interested parties will receive some market share, therefore giving trucking companies an incentive to continue participating in the task allocation activity. As we do not know the exact utility functions of the players, the number of jobs allocated to them will be used to measure the fair distribution of the utilities of the players.

We study a "max-min fair minimum cost allocation problem" (MFMCA). The majority of existing work involving fairness uses mathematical programming models in which fairness is incorporated in either the constraints (Meng and Yang 2002, Perugia et al. 2011) or in the objective function (Bertsimas et al. 2011b, 2012, Barnhart et al. 2012). However, we aim for a polynomial-time solution. The difficulty of our problem lies in the additional fairness criterion, which requires the developed algorithm to satisfy three criteria: allocation maximization, fairness, and cost minimization. To the best of our knowledge, no existing polynomial-time algorithm can be directly applied to solve our problem. In this chapter, we propose polynomial-time algorithms to solve MFMCA as a two-level optimization problem. First, we aim at a fairest allocation among companies while ensuring that a maximal set of tasks can be allocated for execution. We call this the "max-min fair allocation problem" (MMFA). Second, because there might be an exponential number of allocations that

[2] Auctions are used in this research as a way to collect local information from the participants. We do not consider the bidding behavior of the bidders in this chapter.

are considered max-min fair, we would like to determine which of these fair allocations has the lowest cost. The resulting allocation is max-min fair with minimum cost. To this end, we develop a polynomial-time optimal method that consists of two novel algorithms: (1) to solve MMFA, we construct an algorithm, called IMaxFlow, using a progressive filling idea in a flow network (Bertsekas et al. 1987), and then (2) by using the solution obtained from MMFA, we propose another algorithm, called FairMinCost, that smartly alters the structure of the problem to solve MFMCA optimally.

The contribution of this chapter is two-fold.

1. Despite the new fairness criterion, we are able to develop an optimization method to solve the task allocation problem to optimality in polynomial-time.

2. Using computational results, we provide insights into situations in which fairness can be incorporated without giving up too much efficiency.

The rest of the chapter is organized as follows. We start with a literature review in Section 2.2, followed by a problem definition in Section 2.3. In Section 2.4, we introduce two polynomial-time algorithms to solve MMFA and MFMCA, respectively. We prove that the output of these algorithms is the optimal allocation in terms of fairness and cost minimization. In Section 2.5, using different sets of scenarios, we test the algorithm in terms of its effect on the cost and job distribution. We conclude and point out interesting directions of future work in Section 2.6.

2.2 Literature review

The idea of factoring fairness into decision making has been studied in various fields. One of the earlier and still important areas of application where fairness has been considered is that of bandwidth allocation in telecommunication networks (Jaffe 1981, Zukerman et al. 2005). In this area, continuous flows with predefined origin-destination pairs are used, leading to algorithms that increase flow over all paths simultaneously until links are saturated, or that split up bandwidth equally among competitors. Bertsekas et al. (1987) give a simple algorithm for computing max-min fair rate vectors for flow control in networks, the so-called progressive filling algorithm, which is treated as one of the standard fairness concepts within the

telecommunications or network applications (Ogryczak et al. 2005). In their problem setting, they assume that each session has an associated fixed path in the network. The algorithm starts with no flow, and then flow gets gradually increased over all paths simultaneously until a link in a path is saturated. The algorithm then continues from step to step, equally incrementing the flow in all paths that are not using saturated links, until all paths contain at least one saturated link. Tomaszewski (2005) provides a general mathematical programming formulation for solving max-min fair problems using the progressive filling algorithm. Although we cannot use these proposed solution methods directly, we are able to borrow the idea of the progressive filling algorithm when developing our method for solving MMFA.

Fairness, or equity, has also been incorporated in staff scheduling. They attempt to distribute the workload fairly and evenly among personnel, where it is a typical strategy to construct cyclic rosters (Ernst et al. 2004). The more popular measures for equity in this field are the variance and variants of the Gini index. Equity is then incorporated in mathematical models in either the objective function, e.g. minimizing the variation in workload, or through the use of constraints, which provide lower and upper bounds on the workload (Eiselt and Marianov 2008). Resource allocation is yet another field in which fairness plays an important role. An example of a very weak fairness constraint in this field is that any task will be able to use its requested resource eventually. A much stricter fairness requirement can be found in proportionate fairness (Baruah et al. 1996). With proportionate fairness, the difference in the number of resource allocations to tasks will never be more than one, ensuring that all tasks have similar access to resources. Dominant Resource Fairness is another type of fairness requirement, which is a generalization of max-min fairness for multiple resources, where it maximizes the minimum dominant share across all users (Ghodsi et al. 2011). Fairness influences the order in which resources are scheduled to tasks, as certain tasks may take precedence.

Another domain in which fairness is incorporated is the field of air traffic management. In this field, fairness is important for air traffic flow management (Lulli and Odoni 2007, Barnhart et al. 2012), flight scheduling (Kubiak 2009), and allocation of take-off and landing slots at airports (Bertsimas et al. 2011b, 2012). These studies consider a fair distribution of the utilities of all players usually expressed in monetary units or delay time. The air traffic flow management problem has been shown to be

NP-hard (Bertsimas and Patterson 1998), and therefore mathematical programming models are often used in which the fairness measurement is incorporated in the objective function with which good computational results are achieved (Bertsimas et al. 2011b, 2012, Barnhart et al. 2012). In addition, Hoffman et al. (2005) and Kim and Hansen (2015) emphasize that equity and fairness are important in the air traffic flow management program design, because equitable treatment of airlines in such programs will be less likely to encourage gaming behavior by a highly competitive industry. If one fails to consider equity, it might be detrimental to an otherwise well-designed air traffic flow management program. Ogryczak et al. (2014) provides a nice overview of the various areas of application of fairness and the most important models and methods of fair optimization.

Surprisingly, fairness has not yet been investigated widely in transportation optimization problems, although it has been treated as a psychological factor that influences acceptability of policies like road pricing (Fujii et al. 2004, Eriksson et al. 2008). In road network design fairness is also an issue, because without fairness network users might not get any benefit from the network design project, and therefore it may be difficult to rally public support and it may be easy to evoke opposition to the implementation of such a project (Meng and Yang 2002). In this application fairness is enforced through the addition of a constraint on the difference of the travel cost ratio between before and after the project. There has also been some work in vehicle routing problems, where fairness is considered in the extra-time distribution of a transportation service (Perugia et al. 2011). In order to incorporate fairness, they make use of a capping function, which enforces an upper bound on the extra-time. Litman (2002) gives an overview of many different transportation decisions where fairness could be incorporated. However, there is hardly any literature on incorporating fairness in task allocation problems in transportation.

We use the number of tasks allocated to a player as our measure of fairness. Thus, fairness is a property inherent in the allocation itself. We introduce a novel solution method because in our problem we try to assign tasks to players without any information on the players' utilities. This is in contrast to Bertsimas et al. (2011a, 2012), who assume that one knows the utilities of players, such that efficiency and fairness can be expressed as a function of the utilities. In addition, we define our fairness measurement in terms of the allocation itself rather than in terms of

some characteristic of the consequence of the allocation. Examples of the latter are tardiness and delay time, which are often used in air traffic flow management (Lulli and Odoni 2007, Bertsimas et al. 2011b, Barnhart et al. 2012). We will show that our fairness measurement simplifies the optimization problem and that we are able to develop polynomial-time algorithms to find an optimal fair allocation, which is highly desirable in practice.

2.3 Problem definition

We assume that the set of available tasks (or jobs) to be distributed is known in advance by the central planner. For instance, in our motivating example, the terminals know a day in advance which container vessels will arrive and how many containers they will need to handle. The terminals are thus able to construct a schedule for their vehicles and cranes a day ahead, and this schedule reveals the necessary inter-terminal transport movements. These inter-terminal transport movements are the jobs to be auctioned. We assume a set of time periods $\mathcal{T}$, which consists of T time periods. The set of jobs, denoted by $\mathcal{J}$ consisting of a total of J jobs, comes with an earliest available time and a latest completion time for each job. We assume that jobs are independent. Each job can therefore be executed individually regardless of the execution of other jobs. We define for each job $j_i \in \mathcal{J}$ its possible starting time as a mapping: $\mathcal{J} \times \mathcal{T} \mapsto \{0, 1\}$. When it is clear from the context, we abuse the notation and use j_i^t to denote that job j_i is available at time period $t \in \mathcal{T}$.

Once the set of jobs $\mathcal{J}$ together with their possible starting times has been made available, a set of companies $\mathcal{K}$, consisting of K companies, may bid on individual jobs. We do not consider combinatorial bids in this chapter. In addition to the selection of jobs that a company $k \in \mathcal{K}$ wishes to perform, the company also needs to provide their available capacity n_k^t in time period t in which it is able to perform the jobs. We assume that each job takes up one unit of capacity and can be completed within one time unit. Furthermore, company k needs to provide its desired compensation (or cost), $c(j_i, k)$, for the bid job $j_i \in \mathcal{J}$. A bid, B_k, from a company k is thus a tuple: $\langle \boldsymbol{c}_k, \boldsymbol{n}_k \rangle$, where $\boldsymbol{c}_k$ is a set of costs $c(j_i^t, k)$, which denote the compensation

of performing job j_i at time t, and $\boldsymbol{n}_k$ is a set of capacities n_k^t, which specify the capacity of company k at time period t.

The focus of this research is on the design of task allocation algorithms, and not on the auction design. Therefore, to illustrate our approach, we adopt a simple sealed-bid first-price auction format, where terminals can announce their available tasks and each company can submit their bids via, for example, a bidding website. In a sealed-bid first-price auction all bidders submit their sealed bids simultaneously so that no bidder knows the bids of other participants and the winning bidder pays the price they submitted. Once all bids from the bidding companies $\mathcal{K}$ have been collected, which can be enforced by a time limit, the auctioneer then decides which companies get to execute which jobs, that is, the auctioneer determines a task allocation $\pi : \mathcal{J} \times \mathcal{T} \times \mathcal{K} \mapsto \{0, 1\}$. An allocation is *feasible* if (1) each job is allocated to at most one time slot and to at most one company, i.e., for each $j \in \mathcal{J}$, $\sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \pi(j, t, k) \leq 1$; and (2) the number of jobs needed to be executed at time t does not exceed the capacity at time t of the company to which those jobs are assigned, i.e., for each k and t, $\sum_{j\in\mathcal{J}} \pi(j, t, k) \leq n_k^t$. The companies then receive the corresponding compensations specified in their bids for executing the assigned jobs. The focus of this chapter is on determining an optimal allocation π of jobs to bidders. Following our motivating example, there are three ordered objectives for a fair job allocation: (1) the number of allocated jobs in π is maximized, (2) the allocation is fair to the bidders, and (3) the total compensation for executing the jobs is minimized.

Objectives 1 and 3 are rather straightforward given the context. For the fairness objective, we use the notion of max-min fairness derived from Rawls' fairness principle (Rawls 1971). The central idea of max-min fairness is that the minimum utility of all bidders will have been maximized. In this work, we use the number of allocated jobs as a measure of the bidders' utility. In this way, we do not need to worry about the companies' actual utility functions, which they are likely unwilling to share with the auctioneer and which are difficult to model.

Given a feasible allocation π, let the number of allocated jobs be

$$Z = \left|\{j_i \ : j_i \in \mathcal{J}, \pi(j_i, \cdot, \cdot) = 1\}\right|.$$

Let $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_K)$ denote the number of jobs ω_k assigned to company $k \in \mathcal{K}$ in π. We call $\boldsymbol{\omega}$ an allocation vector. Clearly, it holds that $\sum_{\omega_k \in \boldsymbol{\omega}} \omega_k = Z$. Given Z jobs, there may exist many possible allocations that distribute Z jobs to K companies. We call an allocation vector $\boldsymbol{\omega}$ *Z-feasible* if and only if $\boldsymbol{\omega}$ can lead to a feasible allocation and $\sum_{\omega_k \in \boldsymbol{\omega}} \omega_k = Z$.

The max-min fairness principle entails that given a total of Z jobs, the number of jobs for any company cannot be increased by at the same time decreasing the number of jobs of the other companies that have the same number of jobs or less. More formally, let $\boldsymbol{\omega}$ be a Z-feasible vector, and σ be a sorting operator in which the components of $\boldsymbol{\omega}$ are sorted in nondecreasing order: $\sigma(\boldsymbol{\omega})_i \leq \sigma(\boldsymbol{\omega})_j$ if $\omega_i \leq \omega_j$. Let $\boldsymbol{\phi} = \sigma(\boldsymbol{\omega})$. We want to maximize the lexicographical minimum in all Z-feasible allocation vectors $\boldsymbol{\phi}$. Intuitively speaking, we want to have an allocation that distributes a set of jobs among the companies as evenly as possible.

Definition 1 (Max-min fairness)**.** *Given Z jobs to be distributed, we say a Z-feasible sorted allocation vector $\boldsymbol{\phi}$ is* lexicographically greater *than another Z-feasible sorted vector $\boldsymbol{\phi}'$ if there exists a smallest index j ($1 \leq j \leq K$) such that $\boldsymbol{\phi}_j > \boldsymbol{\phi}'_j$, and for index i, $1 \leq i < j$, it holds that $\boldsymbol{\phi}_i = \boldsymbol{\phi}'_i$. An allocation vector is* max-min fair *with regard to Z jobs if it is lexicographically greater than any other Z-feasible vector.*

We now use the following example to illustrate the three objectives of the job allocation problem.

Example 1. *Suppose we have* 5 *jobs to be auctioned. The jobs can be done in the following time periods:* $(j_1 : j_1^1); (j_2 : j_2^2, j_2^4); (j_3 : j_3^2, j_3^3); (j_4 : j_4^3, j_4^4); (j_5 : j_5^5)$. *Three companies submit their bids, as shown in Table 2.1. The first row in the table shows that company k_1 bids on job j_1 that is to be executed during time period 1, for a compensation of* 20.

In this example, all 5 *jobs can be feasibly assigned. There are five feasible allocations: π_1 assigns j_1^1, j_2^2 to k_2 and j_3^3, j_4^4, j_5^5 to k_3; π_2 assigns j_1^1 to k_1 and $j_2^2, j_3^3, j_4^4, j_5^5$ to k_3; π_3 assigns j_3^2 to k_2 and $j_1^1, j_2^2, j_4^4, j_5^5$ to k_3; π_4 assigns j_1^1 to k_1, j_2^2 to k_2 and j_3^3, j_4^4, j_5^5 to k_3; and π_5 assigns j_1^1 to k_1, j_3^2 to k_2 and j_2^2, j_4^4, j_5^5 to k_3. The allocation vectors of these five assignments are $\boldsymbol{\phi}_1 = (0, 2, 3)$, $\boldsymbol{\phi}_2 = \boldsymbol{\phi}_3 = (0, 1, 4)$, and $\boldsymbol{\phi}_4 = \boldsymbol{\phi}_5 = (1, 1, 3)$, respectively. In this example, we have two max-min fair al-*

Time points	1	2	3	4	5
company k_1	$j_1 : 20$				
company k_2	$j_1 : 30$	$j_2 : 40$, $j_3 : 25$			
company k_3	$j_1 : 10$	$j_2 : 20$, $j_3 : 20$	$j_3 : 25$, $j_4 : 25$	$j_2 : 30$, $j_4 : 20$	$j_5 : 20$

Table 2.1: The bids of three companies include desired jobs in each time period and their associated costs. The capacity of all companies is assumed to be 1 for each time period.

locations: π_4 and π_5, because their allocation vectors $\boldsymbol{\phi}_4$ and $\boldsymbol{\phi}_5$, respectively, are lexicographically greater than any other vectors derived from π_1, π_2, and π_3.

Concerning the third objective of the allocation, we notice that π_4 has a total compensation of 125, while π_5 has a total compensation of 105. Thus, in this example, the optimal allocation that satisfies all three objectives is π_5 as it has the optimal max-min fairness with the least compensation.■

We now formally define the optimization problem that we study in this chapter.

Definition 2 (Max-min fair minimum cost allocation (MFMCA) problem)**.** *Given a set of available jobs $\mathcal{J}$ with their possible starting times, suppose a set of valid bids $\mathcal{B} = \{B_1, \ldots, B_K\}$ is submitted by K bidders. Each bid $B_k = \langle \boldsymbol{c}_k, \boldsymbol{n}_k \rangle$ specifies for each bid job j_i its starting time and the desired compensation $c(j_i^t, k)$, together with the company's capacity n_k^t for each time period $t \in \mathcal{T}$. The objective of the max-min fair minimum cost allocation problem is to find the optimal feasible allocation $\pi_{\boldsymbol{\phi}_f} : \mathcal{J} \times \mathcal{T} \times \mathcal{K} \mapsto \{0, 1\}$, such that the number of allocated jobs $Z = |\{j_i : j_i \in \mathcal{J}, \pi(j_i, \cdot, \cdot) = 1\}|$ is maximum, and the allocation leads to a max-min fairness vector $\boldsymbol{\phi}_f$ with regard to Z jobs, with the least total compensation $\sum_{j \in \mathcal{J}, k \in \mathcal{K}, t \in \mathcal{T}, \pi_{\boldsymbol{\phi}_f}(j,t,k)=1} c(j_i^t, k)$.*

We treat MFMCA as a two-level optimization problem. First, we determine what allocation is deemed max-min fair, and second, we determine which of the possibly many max-min fair allocations has the lowest cost.

Definition 3 (Max-min fair allocation (MMFA) problem)**.** *The objective of the max-min fair allocation problem is to find the optimal max-min fairness vector $\boldsymbol{\phi}_f$ that indicates the maximum number of jobs that can be assigned feasibly and that leads to a max-min fair allocation among all bidders.*

Given the output of the first-level optimization problem (MMFA), i.e., a max-min fairness vector, we search for the allocation that gives the desired fair allocation and that has the lowest total compensation.

2.4 Polynomial-time optimal algorithm for MFMCA

In this section, we introduce a two-stage network flow based polynomial-time algorithm to solve the proposed MFMCA problem. In the first stage, we propose an iterative maximum flow algorithm, called IMaxFlow, to enforce a fairest job distribution over companies while ensuring that the maximal number of jobs can be allocated. The output of the IMaxFlow algorithm, i.e., the optimal max-min fairness vector $\boldsymbol{\phi}_f$, is then used as input to the FairMinCost algorithm to construct a new flow network. By any standard minimum-cost maximum-flow algorithm on this constructed flow network, we prove that we obtain the optimal solution to MFMCA. In the next section we present the proposed two-stage algorithm, starting with the iterative maximum flow (IMaxFlow) algorithm.

2.4.1 IMaxFlow algorithm for solving MMFA

Given an instance of the MMFA problem, we can construct a network flow, and then apply the proposed iterative maximum flow algorithm to obtain the optimal max-min fairness vector.

Suppose the set of available jobs is $\mathcal{J}$. We want to build a flow network to push $\mathcal{J}$ from source node a to sink node b. The flow network is a directed graph $G = (V, A)$ with capacities $C_{u,v}$ for each $(u, v) \in A$. The flow network can be constructed from any problem instance of MMFA by adding the following node layers and arcs from a to b: (1) First, we create a node layer for the jobs $\mathcal{J}$. Each job $j_i \in \mathcal{J}$ of this job layer is connected with source node a. Because each job only needs to be executed once, the capacity of these arcs is 1. (2) As each job has certain time periods in which it can be executed, we construct another node layer next to the job layer with job-time nodes j_i^t for each available time period t for each job j_i. The job-time nodes

are connected to their corresponding job nodes in the job layer with the arcs having a capacity equal to 1, because a job can only be executed at most once in a certain time period. (3) From the bids of the companies we know which companies bid on which jobs at which time periods with a certain cost. Therefore, from these bids we can construct yet another node layer with company-time nodes that indicate the time periods t in which each company k is available, denoted by k^t. These nodes are connected to the corresponding job-time nodes where the company made a bid at that particular time period. These arcs each have a capacity of 1. However, unlike previously created arcs, these arcs have costs associated with them equal to the corresponding compensations indicated in the bids. These costs do not play a role in solving MMFA, as its objective is not related to the cost. (4) Once we have constructed this company-time layer, we can construct another node layer consisting of company nodes. Each node in this company layer corresponds to a company $k \in \mathcal{K}$. The company-time nodes in the company-time layer will then be connected to their respective companies in the company layer to aggregate the former. These arcs have a capacity n_k^t equal to the capacity that a company k has indicated as being available in that particular time period t. Finally, we connect all nodes in the company layer with sink node b. For each company $k \in \mathcal{K}$ the edge between its node and the sink has a capacity $N_k = \sum_{t \in \mathcal{T}} n_k^t$, which is the total capacity over all time slots. An example of the resulting flow network is illustrated in Figure 2.1.

Given the constructed flow network G, the value of a flow $f = f(a, b)$ is the total flow that can be pushed from the nodes in the company layer to the sink node b, i.e., $f = \sum_{v \in \{k_1, \ldots, k_K\}} f(v, b)$. Hence, it is clear that the solution to the problem of finding the maximum flow given the translated flow network problem is equivalent to finding the maximum number of jobs that can be allocated to the companies in MMFA. Therefore, given an instance of the MMFA problem, if we run a standard maximum flow algorithm on the constructed flow network G, we will obtain a solution that tells us the maximum number of jobs that can be allocated.

However, the objective of the MMFA problem is also to find the optimal max-min fair solution. Therefore, to solve this maximum flow problem with an additional fairness property, we introduce an iterative maximum flow algorithm that applies the maximum flow algorithm, such as Edmonds-Karp (Edmonds and Karp 1972), in a greedy fashion. In this way, the flow assigned to each company is increased

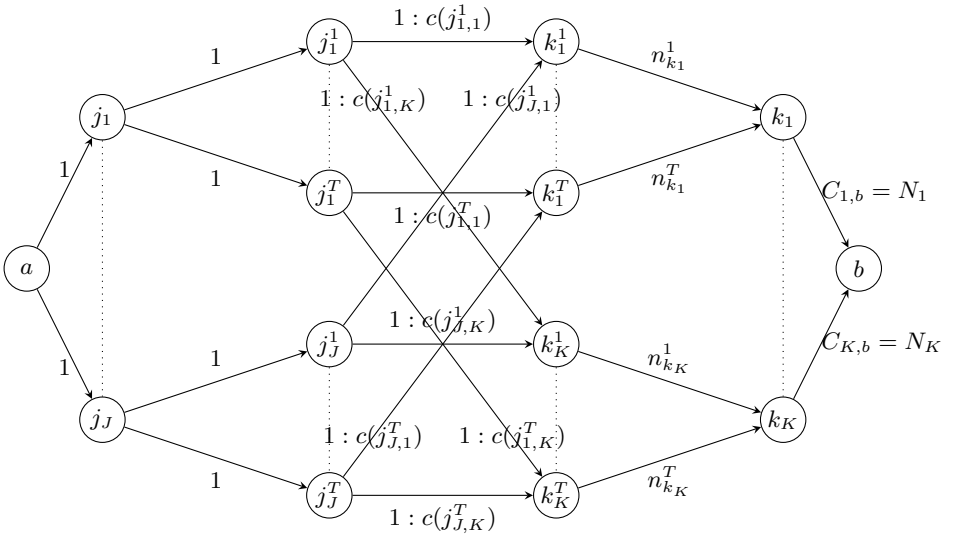


Figure 2.1: A constructed flow network for solving MMFA, where $j_1, \ldots, j_J$ represent a set of available jobs, $j^1_1, \ldots, j^T_J$ are job-time nodes, $k^1_1, \ldots, k^T_K$ are company-time nodes, and $k_1, \ldots, k_K$ represent a set of companies.

step by step until no more flows can be assigned. This idea is similar to the so-called progressive filling algorithm (Bertsekas et al. 1987). Our proposed iterative algorithm IMaxFlow works as follows.

Initiation We construct a set $\mathcal{Q}$ that contains all companies and we set the capacity for all companies to 0, which means that in G, the capacity on the arcs connecting the nodes $k_1, \ldots, k_K \in \mathcal{K}$ in the company layer and the sink node b is set to 0, i.e., $C_{k,b} = 0$ for all $k \in \mathcal{K}$ in Figure 2.1.

Iterations In the first iteration I_1, we arbitrarily pick a company $k_q \in \mathcal{Q}$ and then increase its capacity by 1, i.e., $C_{k_q,b} = C_{k_q,b} + 1 = 1$. We then run a standard maximum flow algorithm, which returns a maximum flow $f^{I_1}_{k_q}$ given the restricted capacities. We check whether k_q receives a flow, i.e., whether $f^{I_1}(k_q, b) = 1$ is true. If $f^{I_1}(k_q, b) = 0$, then we can conclude that company k_q will not be allocated any

job even if we would further increase its capacity. In this case, we fix the capacity $C^f_{k_q,b}$ of the edge between the sink and company k_q to $C^f_{k_q,b} = 1 - 1 = 0$, and remove company k_q from set $\mathcal{Q}$. If $f^{I_1}(k_q, b) = 1$, then we know that company k_q can handle a flow of 1, so we can let $C_{k_q,b} = 1$ and continue. We then choose another company in $\mathcal{Q}$ and repeat the above-mentioned process until we have done the same for all companies in $\mathcal{Q}$. Recall that all companies can get at most one job in this iteration because their capacity is set to 1.

We then start the next iteration I_2. We arbitrarily pick a company $k_q \in \mathcal{Q}$ and check whether it has reached its total capacity. If so, we fix its capacity $C^f_{k_q,b} = C_{k_q,b}$ and remove k_q from $\mathcal{Q}$. Otherwise, we increase its capacity to $C_{k_q,b} = C_{k_q,b} + 1 = 2$. We again run the maximum flow algorithm on G with the updated capacity and obtain a maximum flow $f^{I_2}_{k_q}$. If the maximum flow $f^{I_2}_{k_q}$ is the same as the maximum flow obtained in the previous step (for the first company in iteration I_2, this is the flow at the end of the previous iteration, f^{I_1}), we can conclude that increasing the capacity $C_{k_q,b}$ for company k_q does not result in a larger flow. Therefore, we fix the capacity $C^f_{k_q,b}$ of the edge between the sink and company k_q to $C^f_{k_q,b} = 2 - 1 = 1$, and remove company k_q from $\mathcal{Q}$. We repeat this for all other companies $k_q \in \mathcal{Q}$. For the subsequent companies in the same iteration, we compare the flow obtained after running the maximum flow algorithm on G with the maximum flow obtained in the previous step, which is $f^{I_2}_{k_{q-1}}$. If the maximum flow $f^{I_2}_{k_q}$ is larger than the maximum flow obtained in the previous step, then we can let $C_{k_q,b} = 2$ and continue.

In this way, during iteration I_i we fix a company k_q's capacity to $C^f_{k_q,b} = i - 1$ in G, either when the company does not receive more flow than in the previous step $I_{i,k_{q-1}}$ (or I_{i-1} if k_q is first in I_i), or when the company reaches its maximal total capacity, i.e., $C^f_{k_q,b} = i - 1 = N_{k_q}$. In each iteration we always add one more capacity to the company-sink edges whose capacities have not been fixed.

Termination We iterate this process until $\mathcal{Q}$ is empty, that is, when the flow no longer increases with the addition of more capacities to the companies, or when the capacities of all the companies have reached their limits. It also follows that the capacities of all the company-sink arcs are fixed to some values.

We return the maximum flow f found upon termination as the maximum number of jobs that can be allocated, and the fixed capacities $C^f_{k,b}$. The fixed capacities $C^f_{k,b}$

— equivalent to the number of flows on the company-sink edges, $f(k_1, b)$, $f(k_2, b)$, ..., $f(k_K, b)$ — specifies the number of jobs ω_{k_1}, ω_{k_2}, ..., ω_{k_K} assigned to companies $k_1, \ldots, k_K$. The fixed capacities $C^f_{k,b}$ also comprise the max-min fairness vector $\boldsymbol{\phi}_f = \sigma(\boldsymbol{\omega})$.

This iterative maximum flow algorithm is described in Algorithm 1. Note that this adaptation is independent of the maximum flow algorithm used and is therefore suitable to be used in combination with any existing maximum flow algorithm.

Algorithm 1 IMaxFlow algorithm for solving MMFA

Input: $G = (V, A)$ a constructed flow network for an instance of MMFA, where a, b are the source and sink node, respectively. The capacity of a company-sink edge is denoted as $C_{k,b}$ for $k \in \mathcal{K}$. N_k denotes the maximum capacity of company k

Output: a maximum flow f and a max-min fair allocation vector $\boldsymbol{\phi}_f$

```
f_curr ← 0; f_prev ← −1
Q = K; I = 0  {I denotes the iteration number}
C^f_{k,b} ← 0, ∀ k ∈ K  {C^f_{k,b} denotes the final fixed capacity for company-sink edge e(k, b)}
C_{k,b} ← 0, ∀ k ∈ K  {update G by setting capacities of company-sink edges to 0}
while Q ≠ ∅ do
  I = I + 1  {increase the iteration number by 1}
  for each k ∈ Q do
    f_prev ← f_curr
    if C_{k,b} < N_k then
      C_{k,b} ← C_{k,b} + 1
      Call maximum flow algorithm (MF) on G, f_curr ← MF(G)
      if f_curr = f_prev then
        C_{k,b} ← C_{k,b} − 1; C^f_{k,b} ← C_{k,b}
        Q ← Q \ {k}
      end if
    else
      C^f_{k,b} ← C_{k,b}, Q ← Q \ {k}
    end if
  end for
end while
return f_curr as f, sorted (C^f_{1,b}, ..., C^f_{K,b}) as φ_f
```

We illustrate IMaxFlow with the following example.

Example 2. *Refer to the problem instance in Example 1. We can construct the accompanying flow network as shown in Figure 2.2. The IMaxFlow algorithm first sets all the capacities of the three companies — i.e., the edges $e(k, b)$ connecting to sink b — to 0. Then it increases the capacity of $e(k_1, b)$ by 1 and runs the maximum flow algorithm, which obtains $f(k_1, b) = 1$. This is repeated for each company. At the end of the first iteration we have $f^{I_1}(k_1, b) = f^{I_1}(k_2, b) = f^{I_1}(k_3, b) = 1$, and the total maximum flow is $f^{I_1} = 3$. This can be achieved by pushing a flow from j_1 to k_1, a flow from j_2 to k_2, and a flow from j_3 to k_3.*

During the second iteration, $C^f_{k_1,b}$ is fixed to 1 as k_1 has reached its highest capacity and $f^{I_2}_{k_1} = f^{I_1} = 3$. Next, the capacity of $e(k_2, b)$ is set to 2. After running a standard maximum flow algorithm, we have a maximum flow $f^{I_2}_{k_2} = 3$, because k_1 and k_2 together can be assigned two jobs (either j_1, j_2 or j_1, j_3) and k_3 receives one job because its capacity is still 1. As $f^{I_2}_{k_2} = f^{I_2}_{k_1}$, increasing k_2's capacity does not help to increase the flow but may harm the fairness because it may happen that both j_1, j_2 (or j_1, j_3) can be allocated to k_2. Hence, we fix k_2's capacity $C^f_{k_2,b}$ to 1. We then look at the case where the capacity of $e(k_3, b)$ is increased to 2. It is clear that $f^{I_2}_{k_3}$ is now 4.

Thus, we continue with iteration 3, where we only increase k_3's capacity to 3. After running IMaxFlow, we have a flow of 5, with a possible allocation of j_1 to k_1, j_3 to k_2, and j_2, j_4, j_5 to k_3.

As increasing k_3's capacity will not increase the flow any further, $C^f_{k_3,b}$ is fixed to 3, and the algorithm terminates. The maximum number of allocated jobs is 5, with a max-min fairness vector of $\boldsymbol{\phi}_f = (1, 1, 3)$, which is simply the fixed capacity of each company-sink edge sorted in nondecreasing order. ■

We now prove that IMaxFlow is correct, that is, the returned flow value f is the maximum number of jobs that can be allocated, and the returned fairness vector $\boldsymbol{\phi}_f$ is the most fair job distribution over the participating companies given f.

Theorem 1. *IMaxFlow allocates the maximum number of jobs to the companies and returns a sorted allocation vector that is max-min fair.*

Proof. We will prove by induction that given a set of companies $\mathcal{K}$, at any iteration I_i of the algorithm IMaxFlow, given the available capacities of $\mathcal{K}$, the returned flow f^{I_i}

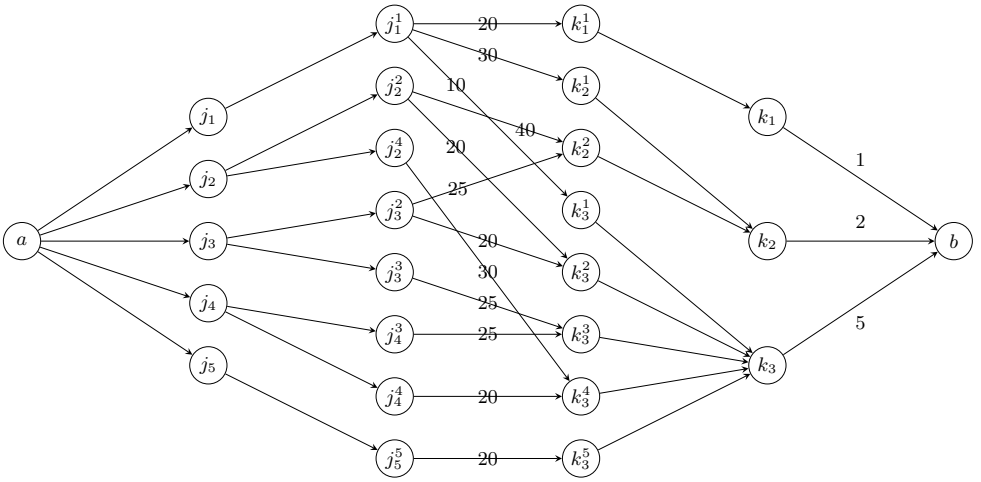


Figure 2.2: The constructed flow network given the problem instance described in Example 1. The capacities of the arcs in the flow network are 1, except for the arcs between the company nodes k_k and the sink b. The numbers on the edges between the job-time nodes j_i^t and the company-time nodes k_k^t specify the compensations of company k performing job j_i at time period t. We do not take these costs into account in MMFA.

is maximum, and the sorted allocation vector is max-min fair among all f^{I_i}-feasible vectors.

Base case: All companies start with capacity 0. In the first iteration I_1 of IMaxFlow, one company $k \in \mathcal{K}$ is arbitrarily picked and assigned a capacity of 1. Then we run the maximum flow (MF) algorithm, which determines the maximum flow of the network given the current available capacity. If this added capacity did not increase the total flow, k's capacity is fixed to 0. At the end of iteration I_1, when the last company is given a capacity of 1 and the maximum flow algorithm is run, it is clear that the returned flow f^{I_1} is maximum given the total capacity of $\mathcal{K}$. Let $\mathcal{K}_0$ be the set of companies whose capacities have been fixed to 0 during this iteration. Then

the sorted allocation vector is

$$\boldsymbol{\phi}_{f^{I_1}} = (\underbrace{0,\ldots,0}_{|\mathcal{K}_0|},\underbrace{1,\ldots,1}_{K-|\mathcal{K}_0|}).$$

It is possible that the set $\mathcal{K}_0$ is not unique. For example, a flow can be pushed either through j's node or k's node. If we pick j first to increase the capacity and to test the flow, then later increasing k's capacity to 1 will not increase the total flow and hence k's capacity will be fixed to 0, i.e., $k \in \mathcal{K}_0$. On the other hand, if we pick k earlier than j, we will have $j \in \mathcal{K}_0$. This situation however gives us the same sorted vector of two companies, which is $(0, 1)$. Thus, the first iteration of the algorithm may result in a different set $\mathcal{K}_0$, but the size of $\mathcal{K}_0$ is always the same and the total number of flows f^{I_1} that can be pushed is always maximum. Therefore, the resulting sorted allocation vector is the same for all possible f^{I_1}-feasible vectors, and it is max-min fair. Thus the statement holds for the first iteration I_1.

Induction step: Suppose the statement is true for iteration I_i, that is, after this iteration, the returned flow f^{I_i} is maximum given the total capacity added, and the sorted allocation vector $\boldsymbol{\phi}_{f^{I_i}}$ is max-min fair. Let $\boldsymbol{\phi}_{f^{I_i}}$ be

$$\boldsymbol{\phi}_{f^{I_i}} = (\underbrace{C^f_{1,b},\ldots,C^f_{m,b}}_{\text{fixed}},\underbrace{i,\ldots,i}_{K-m}).$$

In $\boldsymbol{\phi}_{f^{I_i}}$, suppose there are m company-sink edges with fixed capacities $C^f_{h,b}$, $1 \le h \le m$. We denote these companies as $\mathcal{K}_{\text{fix}}$. For the remaining unfixed $K-m$ company-sink edges, according to the algorithm, the amount of flow must be equal to their assigned capacity on iteration I_i, which is i. Hence, we have for I_i the maximum flow $f^{I_i} = \sum_{h\in\mathcal{K}_{\text{fix}}} C^f_{h,b} + i \times (K-m)$.

Now we need to show the statement stays true for iteration I_{i+1}. During this iteration, each company $j \notin \mathcal{K}_{\text{fix}}$, who does not have a fixed capacity, is assigned one more capacity to have a total capacity of $i+1$. Let $j \notin \mathcal{K}_{\text{fix}}$ be the first company to increase the capacity. After running the MF algorithm, the returned maximum flow is either f^{I_i} or $f^{I_i}+1$, corresponding to the cases that j will receive either i flow or $i+1$ with an extra capacity. If j receives i flow, it is because either its original capacity $N_j = i$ or only i flow can be pushed along the job nodes to company j's

node. At the end of iteration I_{i+1}, all companies not in $\mathcal{K}_{\text{fix}}$ have been given one more capacity and have been tested with the MF algorithm. Assume L companies not in $\mathcal{K}_{\text{fix}}$ will be assigned $i+1$ flows after iteration I_{i+1}. Then the total flow $f^{I_{i+1}} = f^{I_i} + L$ is maximum given the capacity of this iteration, as f^{I_i} is maximum at iteration I_i. The resulting sorted allocation vector is

$$\boldsymbol{\phi}_{f^{I_{i+1}}} = (\underbrace{C^f_{1,b}, \ldots, C^f_{m,b}}_{m}, \underbrace{i, \ldots, i}_{K-m-L}, \underbrace{i+1, \ldots, i+1}_{L}).$$

Similar to the reasoning for the base case of iteration I_1, these L companies could be different depending on the ordering of adding one extra capacity and testing. However, the sorted allocation vector for the companies in $\mathcal{K}_{\text{fix}}$ is always the same, i.e, $(\underbrace{i, \ldots, i}_{K-m-L}, \underbrace{i+1, \ldots, i+1}_{L})$. Together with the fact that $\boldsymbol{\phi}_{f^{I_i}}$ is max-min fair in the previous iteration I_i, we have shown that $\boldsymbol{\phi}_{f^{I_{i+1}}}$ is max-min fair among $f^{I_{i+1}}$-feasible allocation vectors.

Conclusion: By the principle of induction, it follows that the preceding statement is true for any iteration of the algorithm IMaxFlow.

Hence, it follows that after the final iteration IMaxFlow returns the maximum number of jobs to the companies and the sorted allocation vector is max-min fair. □

As a by-product of the above reasoning, we have the following lemma.

Lemma 1. *IMaxFlow returns a unique max-min fair allocation vector, given the maximum number of allocated jobs.*

Finally, we show that the proposed algorithm is a polynomial-time algorithm (see 2.A for the proof).

Theorem 2. *The IMaxFlow algorithm runs in time* $O((J^3K^3T^3) + (J^2K^4T^3))$.

2.4.2 FairMinCost algorithm

Once we know the fairness vector from IMaxFlow, we want to minimize the associated cost (compensations). Because there are many feasible max-min fair maximum flow solutions with different costs, we want to find the one with the minimum cost.

Unfortunately, we cannot apply a standard minimum-cost maximum-flow algorithm to our flow network as it may violate the max-min fairness condition while looking for the minimum cost. The obtained fairness vector tells us in what quantities the jobs will be distributed in the fairest allocation. However, we do not know *which* company would be assigned *which* number of jobs such that the total cost is smallest.

If we know the exact number of jobs all companies would get, MFMCA is easily solvable using a minimum-cost maximum-flow algorithm. This is obvious because we can set the capacities of the arcs from the company nodes to the sink to be equal to the number of jobs of the respective companies. Since we know from MMFA that the flow is maximal and feasible, and that the capacities sum up to this maximum flow, we know that all jobs will be assigned. This boils down to a simple minimum-cost maximum-flow problem that can be solved using any of the existing algorithms.

However, if the exact number of jobs that all companies will get is not known, then the capacity for each company can be any of the capacities in the fairness vector. This leaves us with many ways to construct the flow network because it is assumed that the capacity of the arcs in a minimum-cost maximum-flow problem are known. We can deal with this problem in several ways.

One way to find the minimum cost among all possible max-min fair allocations is to simply enumerate all possible max-min fair allocations and solve a minimum-cost maximum-flow problem for each of them, and then to finally choose the allocation that has minimum cost. However, this method would be computationally inefficient, because it can be viewed as a multiset permutation with $\binom{p}{r_1,r_2,\ldots,r_p} = \frac{p!}{r_1!r_2!\ldots r_p!}$ possibilities, where $p = \left|\boldsymbol{\phi}_f^U\right|$, in which $\boldsymbol{\phi}_f^U$ denotes the vector of unique capacities in $\boldsymbol{\phi}_f$, and r_i denotes how often capacity i appears in $\boldsymbol{\phi}_f$, $r_i = \sum_{k\in\mathcal{K}} \left|\boldsymbol{\phi}_f(k) = i\right|$. For each possibility, we would need to run a minimum-cost maximum-flow algorithm. The resulting running time would be exponential.

Instead, in this chapter we propose an algorithm that makes variable capacities on arcs in the flow network possible. Given the fairness vector $\boldsymbol{\phi}_f = (\phi_1, \ldots, \phi_K)$, we introduce a solution method that runs in polynomial-time. To this end, we adjust the network flow model such that the fair job distribution $(\phi_1, \ldots, \phi_K)$ will be intact at the same time that cost minimization takes place. The challenge is to somehow

enforce the capacities of the fairness vector obtained from IMaxFlow in the final allocation.

For ease of explanation, we denote an instance of the original MFMCA problem as $P = \langle \mathcal{J}, \mathcal{K}, \mathcal{T}, \mathcal{B} \rangle$. We now introduce a new problem $P' = \langle \mathcal{J}', \mathcal{K}', \mathcal{T}', \mathcal{B}' \rangle$ adapted from P. The key construction of P' given P is that we will update the original set of jobs $\mathcal{J}$ to $\mathcal{J}' = \mathcal{J} \cup \mathcal{J}^d$, where $\mathcal{J}^d$ is a set of dummy jobs. Each dummy job provides a flow of 1 and has a cost of 0. These dummy jobs will be performed during dummy time periods $\mathcal{T}^d$, thus, $\mathcal{T}' = \mathcal{T} \cup \mathcal{T}^d$. The set of companies $\mathcal{K}'$ in P' stays the same as in the original problem P, i.e., $\mathcal{K}' = \mathcal{K}$, however, they have capacities at dummy periods $\mathcal{T}^d$ for performing dummy jobs $\mathcal{J}^d$. The number of dummy jobs and dummy periods will be determined by the fairness vector $\boldsymbol{\phi}_f = (\phi_1, \ldots, \phi_K)$ returned by the IMaxFlow algorithm on the instance of P. We construct a flow network G' for problem P' based on the constructed flow network G for an instance of problem P by adding the dummy jobs and dummy times in G. In addition, we update the capacities of all companies to ϕ_K. After completing G', we claim that if we run a standard minimum-cost maximum-flow algorithm on G', the solution is a fair minimum cost job allocation for the original problem P. We denote this procedure as the FairMinCost algorithm.

Construction of G'. We first show how we can construct a flow network G' for problem P' based on the constructed flow network G for an instance of problem P. Given G, we will add a number of so-called horizontal dummy layers (DL for short). We have a dummy layer for each increment of 1 from the lowest number of jobs, ϕ_1, to the highest, ϕ_K. Hence, the total number of DL is equal to the difference between the lowest and the highest number of jobs in the fairness vector $\boldsymbol{\phi}_f$, i.e., there are $\phi_K - \phi_1$ dummy layers: $DL_1, DL_2, \ldots, DL_{\phi_K - \phi_1}$. Each layer is meant to provide dummy jobs to companies such that all companies can have jobs up to a specified number. We associate each dummy layer to the specified number in $\boldsymbol{\phi}_f$, that is, DL_1 is associated with number $\phi_1 + 1$, DL_2 with $\phi_1 + 2$, and $DL_{\phi_K - \phi_1}$ with $\phi_1 + (\phi_K - \phi_1) = \phi_K$. In each dummy layer DL_l, we create a set of dummy job nodes $\mathcal{J}_l^d$ in the job layer of the network equal to the number of companies that have a lower capacity than $\phi_1 + l$ in the fairness vector $\boldsymbol{\phi}_f$. These dummy jobs $d_{l,i} \in \mathcal{J}_l^d$ are connected to source node a. We assume that each dummy layer DL_l has its own

unique dummy time t'_l and all dummy jobs from DL_l need to be executed at time t'_l. Thereafter, we create dummy job-time nodes $d_{l,i}^{t'_l}$ in the job-time layer for each dummy job $d_{l,i} \in \mathcal{J}_l^d$, and connect them to their corresponding dummy job nodes $d_{l,i}$. We assume that every company in $\mathcal{K}'$ is capable of performing every dummy job in $\mathcal{J}^d$, but that for each dummy time t'_l the capacity of every company is 1. Thus, for each dummy layer DL_l, we create dummy time-company nodes $k^{t'_l}$ in the time-company layer for each company $k \in \mathcal{K}'$ and connect them to all dummy job-time nodes in that particular dummy layer DL_l. Finally, we connect the dummy time-company nodes k^{t_l} to the company nodes k in the company layer of the flow network G. All added arcs have a cost of 0 and a capacity of 1. Finally, we change the capacities of the arcs from the company nodes k to the sink b in G to the largest number according to the fairness vector, i.e., ϕ_K.

By creating nodes for each company per dummy layer, we are making sure that each company can be assigned at most one dummy job in each dummy layer. Therefore, each company is able to get any of the capacities in the fairness vector and it is not predetermined which companies are assigned which capacity. This is exactly the flexibility we desire.

Example 3. *In Example 1, we obtained a fairness index of $\boldsymbol{\phi}_f = (1, 1, 3)$, and we showed the constructed flow network G in Figure 2.2. We now show how to add dummy layers to G for this instance in order to obtain G'. Figure 2.3 shows the final construction.*

Given $\boldsymbol{\phi}_f = (1, 1, 3)$, we have to create $(3 - 1) = 2$ dummy layers. In the first dummy layer DL_1 we create two dummy jobs $d_{1,1}$ and $d_{1,2}$ for the companies that have capacity 1 in order for each of them to reach a capacity of 2. We then connect these two dummy jobs to the same dummy time t'_1. Thereafter, we create three dummy company-time nodes that are connected to the two dummy job-time nodes and to the three company nodes. The capacity of each arc is 1. This means that every company is able to do any of the dummy jobs during dummy time t'_1 but that only one dummy job from the same dummy layer can be assigned to the same company due to capacity constraints. Subsequently, we use a similar construction for the second dummy layer DL_2. In this layer we again need to create two dummy jobs because there are two capacities smaller than 3 in $\boldsymbol{\phi}_f$. The cost of all added edges is 0. After creating

DL_2, we change the capacities of the arcs between the companies to the sink node from $(1, 2, 5)$ *to* $(3, 3, 3)$. ■

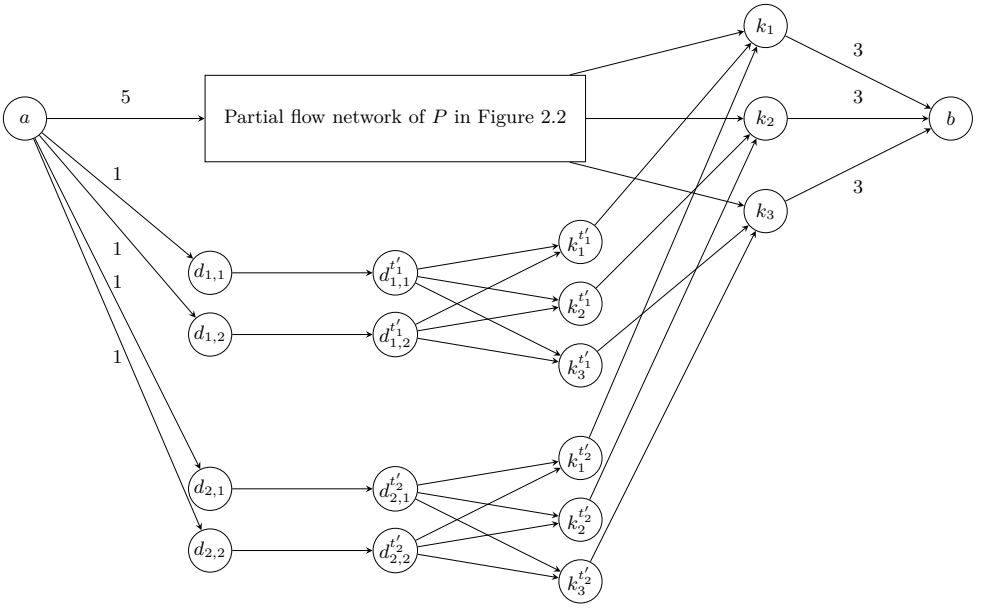


Figure 2.3: Minimum-cost maximum-flow network with dummy jobs for problem P' for the problem instance described in Example 1. All arcs in the added dummy layers have a cost of 0 and a capacity of 1.

Finding minimum-cost maximum-flow. Given the constructed network G', all edges have a cost $\texttt{cost}(e(u', v'))$ of 0, except the edges between the original job-time nodes and the original company-time nodes. We then run any existing (polynomial-time) minimum-cost maximum-flow algorithm on G' that is constructed for problem P'. The solution is a flow f' satisfying $f' = \arg\min_{u',v' \in V'} \texttt{cost}(e(u', v'))f'(u', v')$, and $f'(k)$ is the number of allocated jobs to company $k \in \mathcal{K}$. Let π'_k denote the set of jobs assigned to k. After removing the dummy jobs in π'_k, i.e., $\pi_k = \pi'_k \setminus \{d \mid d \in \mathcal{J}^d\}$, we obtain π_k that is a set of real jobs in $\mathcal{J}$ assigned to company k. Then the allocation

vector $(\pi_{k_1}, \ldots, \pi_{k_K})$ represents an optimal max-min fair allocation with the least costs, which is the solution to the original problem P.

To summarize, given an instance P of the problem MFMCA, we use the IMaxFlow algorithm to obtain a max-min fair allocation vector $\boldsymbol{\phi}_f$. Algorithm FairMinCost then constructs an instance P', built upon the flow network of P, by adding a set of dummy nodes and arcs determined by $\boldsymbol{\phi}_f$. We then run an existing polynomial-time minimum-cost maximum-flow algorithm on the flow network of P'. In the resulting flow, we remove the dummy jobs and dummy flows. Our running example demonstrates how the algorithm works.

Example 4. *We run an existing minimum-cost maximum-flow algorithm on the flow network for problem P' (see Figure 2.3) constructed in Example 3. The job allocation π' for problem P' is: k_1 is assigned $\{j_1^1, d_{1,1}, d_{2,1}\}$, k_2 is assigned $\{j_3^2, d_{1,2}, d_{2,2}\}$, and k_3 is assigned $\{j_2^2, j_4^4, j_5^5\}$. All dummy jobs are assigned, and the allocation vector is $(3, 3, 3)$. The total cost is 105. We now remove all dummy jobs from π' in order to obtain the solution π to the original problem P. We then have: j_1^1 to k_1, j_3^2 to k_2 and j_2^2, j_4^4, j_5^5 to k_3. The fairness vector of π is $\boldsymbol{\phi}_f = (1, 1, 3)$, which is max-min fair, obtained from the algorithm IMaxFlow as illustrated in Example 2. The total cost of π is 105 as dummy jobs have no cost. This is the same as the optimal max-min fair allocation in Example 1.* ■

We now claim FairMinCost is correct, i.e., the final flow returned by the FairMinCost algorithm gives us an optimal solution to P with a max-min fairness vector $\boldsymbol{\phi}_f$ and has the lowest cost given $\boldsymbol{\phi}_f$.

Theorem 3. *The FairMinCost algorithm returns an optimal solution of instance P for MFMCA.*

The proof is given by the following two lemmas.

Lemma 2. *The fairness vector obtained from solving problem P' by the FairMinCost algorithm is the max-min fairness vector $\boldsymbol{\phi}_f$ returned as an optimal solution to MMFA.*

Proof. Let $\boldsymbol{\phi}_f = (\phi_1, \phi_2, \ldots, \phi_K)$ be the nondecreasingly ordered capacities for the companies in the fairness vector of the optimal solution of MMFA. Hence in an

optimal solution to P, the allocation vector is also $\boldsymbol{\phi}_f$. In problem P', the capacities are set to $\boldsymbol{\phi}' = (\phi_K, \phi_K, \ldots, \phi_K)$, as we add $\boldsymbol{\phi}^d = (\phi_1^d, \phi_2^d, \ldots, \phi_K^d) = (\phi_K - \phi_1, \phi_K - \phi_2, \ldots, \phi_K - \phi_K)$ capacity for the dummy jobs.

We show that in the allocation of the optimal solution of problem P', each company $k \in \mathcal{K}$ will be assigned exactly ϕ_k^d dummy jobs by any minimum-cost maximum-flow algorithm. We first note that all dummy jobs will be allocated because there is sufficient capacity added to the network to account for the dummy jobs and they have a cost of zero.

We will show that any company $i \in \mathcal{K}$ cannot get assigned more than ϕ_i^d dummy jobs. Let $i = 1$ be the first company in the sorted allocation vector, i.e., it has the most dummy capacity. For company 1, its number of allocated dummy jobs cannot be more than ϕ_1^d, simply because there are in total $\phi_K - \phi_1 = \phi_1^d$ dummy layers in G' and any company can only get at most one job from each dummy layer. Take an arbitrary company i, $2 \leq i \leq K$, WLOG, suppose company i is assigned $\phi_i^d + 1$ dummy jobs. This extra one dummy job has to come from another company j ($j < i$) who has more dummy capacity than i. Hence, j, $j < i$, should receive $\phi_j^d - 1$ dummy jobs. If j still gets ϕ_j^d dummy jobs, it blocks the possibility for i to receive one extra dummy job as there will not be a sufficient number of dummy jobs in the dummy layers to support this allocation, due to the construction of dummy jobs in dummy layers.

Now, if $\phi_j^d,\ j < i$, will decrease, then there are two possibilities. First, if $0 \leq \phi_j^d - \phi_i^d \leq 1$, then $\boldsymbol{\phi}^d$ will not change as it will retain its order. Second, if $\phi_j^d - \phi_i^d \geq 2$, then this will result in a more even distribution of dummy jobs over companies. However, since $\boldsymbol{\phi}_f = \boldsymbol{\phi}' - \boldsymbol{\phi}^d$, this will result in a more even, or in other words, fairer distribution of jobs in $\boldsymbol{\phi}_f$. This contradicts our claim that $\boldsymbol{\phi}_f$ is max-min fair. The same reasoning holds if we decrease ϕ_i^d, for $i = 1, \ldots, K-1$ due to symmetry. If we decrease ϕ_i^d, our only option is to increase a $\phi_j^d,\ j < i$, which, as we have seen before, cannot occur due to insufficiently available dummy jobs.

□

Lemma 3. *The optimal solution for problem P in terms of cost is the same as the optimal solution for problem P'.*

Proof. Assume that the cost of the allocation of jobs in P' is different than in P. If the allocation in P' has a lower cost than the allocation in P, then because the dummy jobs have a cost of zero and we have added sufficient capacity, we can remove the dummy jobs and obtain an allocation for P that has a lower cost than the optimal allocation in P. This contradicts the assumption that the allocation in P is optimal. Now if the allocation in P' has higher costs than the allocation in P, then we can add dummy jobs to the optimal allocation in P and increase the capacities accordingly so that we obtain problem P'. We will then have an allocation for P' that has a lower cost than the optimal allocation previously found in P'. This contradicts the assumption that the allocation in P' is optimal. Therefore, the cost of the optimal allocation of jobs in P is the same as the cost of the optimal allocation in P'. □

We now show the running time complexity of the proposed FairMinCost algorithm. After constructing the updated graph G', we use a standard solution method for minimum-cost maximum-flow problems, namely the cycle-canceling algorithm. Given a feasible flow, the cycle-canceling method tries to find a negative cycle in the residual graph whose residual capacity it increases, so that the negative cycle disappears and the resulting solution is a solution with lower costs. Instead of choosing an arbitrary negative cycle, the cycle with the minimum mean cost is chosen, which makes the problem strongly polynomial-time solvable (Goldberg and Tarjan 1989). In order to find minimum mean-cost cycles, we use Karp's algorithm (Karp 1978). This will give us our desired solution in which we have an allocation that is max-min fair and has minimum cost among all possible max-min fair allocations.

The runtime complexity of the FairMinCost algorithm is given below (see 2.B for the proof).

Theorem 4. *The FairMinCost algorithm runs in time* $O(J^3K^3(K+T)^3(JK+JT+KT)^2\log(JK+JT+KT))$.

In conclusion, we can optimally solve MFMCA in polynomial time using first the IMaxFlow algorithm and then the FairMinCost algorithm.

2.5 Computational results

In this section we investigate the performance of the algorithms through numerical experiments. We are interested in the following two performance measurements:

1. The effect of fairness on the cost, the so-called *price of fairness* (POF) (Bertsimas et al. 2011a). POF is defined as the relative increase in the total cost (TC) under the fair solution, compared to the minimum cost (MC) solution; that is,

$$POF = \frac{TC(\text{MFMCA}) - TC(\text{MC})}{TC(\text{MC})}.$$

2. The effect of fairness on the job distribution. The job distribution depicts the number of jobs assigned to each company.

Therefore, we generate test cases with various parameters and compute both the minimum cost solution (MC) using the standard minimum mean-cost cycle-canceling algorithm, and the fair minimum-cost solution using the two-stage algorithm: first IMaxFlow, and then FairMinCost. Next to the costs, we take a look at the allocations in both cases and see how fairness influences the allocation. All algorithms are coded in Java with the support of the JGraphT library (JGraphT 2014). We run the experiments on the Lisa Compute Cluster of SURFsara (SURFsara 2014).

2.5.1 Test instances

We derive test instances from our motivating example. In order to make the experiments representative of the situation at the Rotterdam port, we need representative values for the different parameters. First of all, we define one time unit as one hour, just as in the port. Although some tasks require more time than others, a task from one end of the port to the other does not take more than one hour due to the layout of the port. We choose a time window of 10 time periods, $t_1, \ldots, t_{10}$, corresponding to a typical working day. We then set the number of jobs to 250 (Duinkerken et al. 2006), and the number of companies to 50, based on the members of the "VZV" (Verenigde Zeecontainer Vervoerders), the Dutch alliance of sea-container carriers, which represents the different carriers in meetings with the terminals, the port, and other entities.

The jobs have a latest completion time. This is set to 3 time periods after the earliest time the job becomes available. This means that if a job becomes available at the beginning of t_1, it can be started at t_2 and t_3 as well but not at t_4, as its latest completion time was at the start of t_4. Jobs are distributed over the 10 time periods but not uniformly. Since there are two peak hours throughout the day (Duinkerken et al. 2006), we configure jobs to have a 25% chance of starting at t_2 and another 25% chance of starting at t_6. If a job would not specifically start at a peak hour, it has an equal chance to start at any time period from t_1 to t_8. This ensures that each job has a time window of 3 in which it can be executed. This also ensures that the number of jobs available in the first and last two time periods are smoothed out.

Scenarios. In our experiments we assign each company a certain probability to bid on each job at an available time period. We distinguish three different scenarios. The first scenario is when all companies are not very eager to bid on jobs or they do not have many trucks available. In this *low competition* scenario there is a 25% chance of a company bidding on a job, for all companies, for all jobs. The second scenario is the exact opposite: companies are actually eager to bid on the jobs or they have many trucks available. In this *high competition* scenario there is a 75% chance of bidding on a job. In the third scenario we combine the first two scenarios by splitting up the companies into two groups of equal size, where the companies in the first group have low competition, and the companies in the second group have high competition. This case is more representative of reality, as there will be large companies that have plenty of trucks available, and there will also be smaller companies that only have a few trucks available. We call this the *mixed competition* scenario.

The number of trucks all companies will have available at a certain time period, the *capacity*, will be drawn uniformly random between 0% and 5% (*5% capacity*) or between 0% and 10% (*10% capacity*) of the total number of jobs they bid on in that particular time period, rounded to the nearest integer. Furthermore, we ensure that companies will have at least one truck available when they have bid on at least one job. Note that due to the dependency on the number of jobs they bid on in a time period, low competition companies will have fewer trucks available at a certain time period because they bid on fewer jobs, whereas high competition companies will have more trucks available because they bid on more jobs.

Now that we know how companies will bid on jobs, the question remains how much they will bid. We will have two cases here. The first case is when all companies have their bid drawn from the same distribution. We choose a uniform distribution that ranges from 30 to 60. We choose this range because the hourly wage of a truck driver plus the fuel costs for the largest distance within the port amounts to roughly 30 euros. Because companies would like to make some profit with these extra jobs, we let the bids range up to 60. We call this cost scenario the *homogeneous costs* case and this can be applied to all three aforementioned bidding scenarios. The second case is when some companies decide to bid relatively low while others decide to bid relatively high. In the cases of low and high competition, half of the companies will have their bids drawn from a uniform distribution that ranges from 30 to 50 and the other half from 40 to 60. When there is mixed competition, the low competition companies will bid high, from 40 to 60, whereas the high competition companies will bid low, from 30 to 50. The thought behind this is that low competition companies value their trucks more than the high competition companies do. Low competition companies only have a few trucks available and thus can only bid on a few jobs, whereas the high competition companies have more trucks available and will bid on more jobs. Therefore, the high competition companies will have to compete with many others for the same jobs, and so they will offer lower prices. We call this cost scenario the *heterogeneous costs* case.

To summarize, we have six scenarios in total, each with 50 companies, 250 jobs, and a time window of 3 time periods for a job:

1. Low competition, homogeneous costs (low/hom).
2. Low competition, heterogeneous costs (low/het).
3. High competition, homogeneous costs (high/hom).
4. High competition, heterogeneous costs (high/het).
5. Mixed competition, homogeneous costs (mix/hom).
6. Mixed competition, heterogeneous costs (mix/het).

Out of these six scenarios, we believe the last scenario with heterogeneous companies and heterogeneous costs to be the most interesting, because it comes closest

to the real situation at the port. We also expect this scenario to yield a relatively bad performance in terms of price of fairness, because in the minimum cost solution most jobs will be allocated to the companies with high competition and low costs. However, because we want to enforce fairness, we need to also allocate jobs to the companies with low competition and high costs, which may increase the total cost substantially.

For each scenario, we run 100 experiments with both the minimum mean-cost cycle-canceling algorithm for a minimum-cost solution and the fair two-stage algorithm for a fair solution. We record the job allocations in both solutions and the difference in total cost between the minimum-cost solution and the fair solution.

In the end we will investigate the effect of the amount of jobs and companies on the solutions. One can imagine that the price of fairness will differ depending on the number of jobs that needs to be allocated. To test this, we run experiments with 50 to 500 jobs in increments of 50, while maintaining all other parameters. In the same vein, the price of fairness may be dependent on the number of companies present. When there are only a few companies present the allocation may lack flexibility, whereas having many companies might drive up the costs because more companies have to be allocated a number of jobs. Therefore, we run experiments with 25 to 100 companies in increments of 25, while keeping the other parameters the same as in the base case.

2.5.2 Results: price of fairness

We first present the results of the fair allocations compared to the minimum cost allocations for the experiments with all six scenarios. In Figures 2.4 and 2.5, the average price of fairness over 100 experiments per scenario, with *5% capacity* and *10% capacity*, respectively, are shown in a boxplot. Tables 2.2 and 2.3 show the accompanying statistics, i.e., the mean, standard deviation, and the minimum and maximum.

It is clear that costs play an important role in the differences in the total cost between the minimum-cost and fair solutions. In the scenario with homogeneous costs (i.e., low/hom, high/hom, mix/hom), the price of fairness ranges from 0 to 2.42. However, in the scenarios with heterogeneous costs (i.e., low/het, high/het,

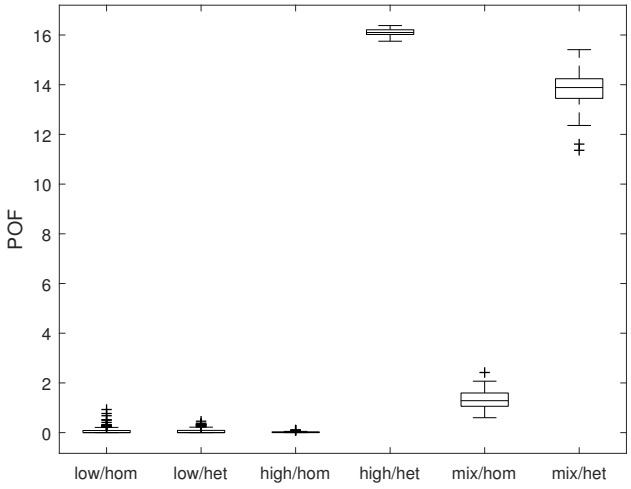


Figure 2.4: Boxplot of the price of fairness with 5% capacity, with the different scenarios and the price of fairness on the horizontal and vertical axes, respectively.

		low/hom	low/het	high/hom	high/het	mix/hom	mix/het
Min-cost	mean	6477.97	7362.67	7515.21	7539.31	7559.42	7537.85
	std	355.28	424.07	4.06	8.41	8.20	8.11
	min	5585	6236	7506	7522	7535	7250
	max	7335	8417	7526	7560	7579	7564
Fair	mean	6483.29	7367.40	7516.45	8752.90	7658.93	8578.70
	std	358.25	425.22	4.52	4.12	28.23	51.025
	min	5582	6245	7506	8732	7602	8403
	max	7385	8417	7529	8760	7743	8695
POF	mean	0.08	0.06	0.02	16.10	1.32	13.81
	std	0.17	0.11	0.02	0.13	0.37	0.70
	min	0.00	0.00	0.00	15.75	0.60	11.36
	max	0.93	0.46	0.11	16.38	2.42	15.41

Table 2.2: Statistics of minimum cost and fair cost and POF over 100 experiments with 5% capacity.

mix/het) the price of fairness ranges from 0 to a staggering 17.27. This is as expected, because if the costs are similar for all jobs for all companies, it will be relatively easy to reallocate jobs to a different company with similar costs. Once costs vary more among companies there will be an increase in costs because jobs that were allocated to relatively cheap companies are forced to be reallocated to more expensive companies.

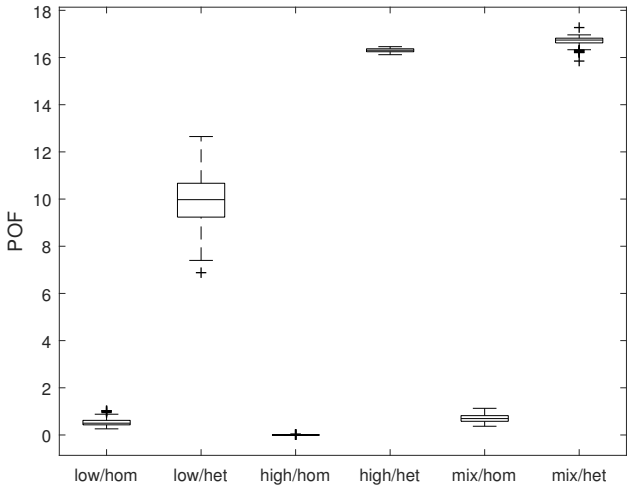


Figure 2.5: Boxplot of the price of fairness with 10% capacity, with the different scenarios and the price of fairness on the horizontal and vertical axes, respectively.

		low/hom	low/het	high/hom	high/het	mix/hom	mix/het
Min-cost	mean	7708.83	8069.41	7509.81	7524.81	7535.81	7524.44
	std	22.24	90.82	3.45	5.26	6.70	5.80
	min	7661	7857	7503	7514	7518	7511
	max	7774	8294	7522	7538	7557	7542
Fair	mean	7750.36	8872.97	7509.90	8751.89	7589.32	8780.72
	std	25.92	17.68	3.48	1.39	14.12	12.84
	min	7704	8833	7503	8750	7561	8733
	max	7816	8934	7522	8756	7622	8815
POF	mean	0.54	9.97	0.00	16.31	0.71	16.70
	std	0.16	1.19	0.00	0.08	0.17	0.20
	min	0.26	6.88	0.00	16.12	0.37	15.85
	max	1.03	12.65	0.03	16.46	1.13	17.27

Table 2.3: Statistics of minimum cost and fair cost and POF over 100 experiments with 10% capacity.

The seeming discrepancy in the low/het scenario between the 5% and 10% capacity cases can be explained by the structure of the low/het scenario. Because there are only low competition companies in this scenario, all companies will bid on only a few jobs. When we set the capacity for each time period to only 5% of those bids, it happens frequently that the capacity is set to an extremely low number, i.e., 0 or 1. This

means that there is not much leeway in the fair solution for jobs to be reallocated. The fair solution is often similar to the minimum-cost solution as there are only a few other possible allocations. This can also be observed in Table 2.2 where the means of the costs of the minimum-cost and the fair solutions in the low/het scenario are similar but are extremely high compared to those in the low/hom scenario.

By increasing the capacity to 10%, we allow more room for jobs to be reallocated. In Table 2.3 we can see that the difference in average cost between the low/het and low/hom scenarios in the minimum-cost solution is significantly smaller than in the 5% capacity case. As there is more room for reallocation, this eventually results in a higher price of fairness.

We can see that competition also has an influence on the price of fairness. Both high and mixed competitions result in a higher price of fairness than low competition. We can see this in the low/het, high/het, and mix/het scenarios in the 10% capacity case (average price of fairness: 9.97, 16.31, and 16.70 respectively). At first this may seem surprising. One would imagine that having more possibilities for allocation will result in both the minimum-cost and fair solutions to be closer to each other compared to when there are limited possibilities. However, this discrepancy can be explained by looking at the minimum cost and fair cost of the solutions (see Table 2.3). We can see that the cost for the minimum-cost solutions in the low/het scenario is on average higher (8069.41) than that of the high/het (7524.81) and mix/het (7524.44) cases. This is due to the limited possibilities if there are companies bidding only on a few jobs. However, the average cost in the fair solutions in the low/het scenario (8872.97) is similar to that of the high/het (8751.89) and mix/het (8780.72) scenarios. This results in a smaller difference between the minimum-cost and fair solutions in the low/het scenario compared to the high/het and mix/het scenarios.

The mix/het scenario, where there is a mix of low- and high-competition companies, seems to have a price of fairness similar to or lower than the high/het scenario (13.81 against 16.10 in the 5% capacity case, and 16.70 against 16.31 in the 10% capacity case). This is somewhat surprising at first. Due to the presence of high-competition companies, it is clear that the minimum-cost solutions would be similar to the solutions in the scenario with high competition because the more expensive low-competition companies are being ignored. However, one would expect that the fair solutions will have higher costs because the more expensive low-competition

companies also need to be allocated jobs. For this we have to keep in mind that the degree of fairness is not equal among the scenarios. It appears that due to the presence of low-competition companies, which have fewer bids and therefore fewer allocation possibilities, they get fewer jobs allocated to them even in the fair allocation. This in turn means that the other, high-competition companies get allocated more jobs that are cheaper. In the end, this results in lower costs overall. This effect can be seen clearly in the 5% capacity case. When we increase the possible capacity to 10%, low-competition companies get assigned more jobs, almost as many as the high-competition companies. This results in higher costs.

Summary. All things considered, we can see that the price of fairness is fairly low when the costs are homogeneous among companies. Jobs can be easily reallocated to make a more fair allocation while keeping the total cost similar because the individual costs of a job for each company are similar. In this case, fairness can be easily enforced without increasing the costs too much or at all. When costs are heterogeneous however, we have to pay a higher price for fairness. This is as expected because we would also need to allocate jobs to companies with high costs, whereas we would only opt for companies with low costs in the minimum-cost solution. In the case of low competition, allocations tend to have slightly higher costs because there is a limited availability of jobs to be allocated to companies. This holds true for both the minimum-cost and fair allocations. In the cases of high and mixed competitions, the costs of the minimum-cost solutions are similar in the cases with homogeneous costs. This is as expected because only the companies with the lowest costs get chosen while the ones with high costs are ignored. However, when we enforce fairness, jobs will be forcibly reallocated to companies with high costs, which might increase the total cost. The price of fairness is the highest in the high/het and mix/het scenarios. Depending on the actual discrepancy between the different costs, one might opt out of the idea of enforcing fairness when the price of fairness becomes too high.

2.5.3 Results: job distribution

Figures 2.6 and 2.7 show job distributions of both the minimum-cost and the fair solutions at one instance for each scenario. We choose to show the job allocations

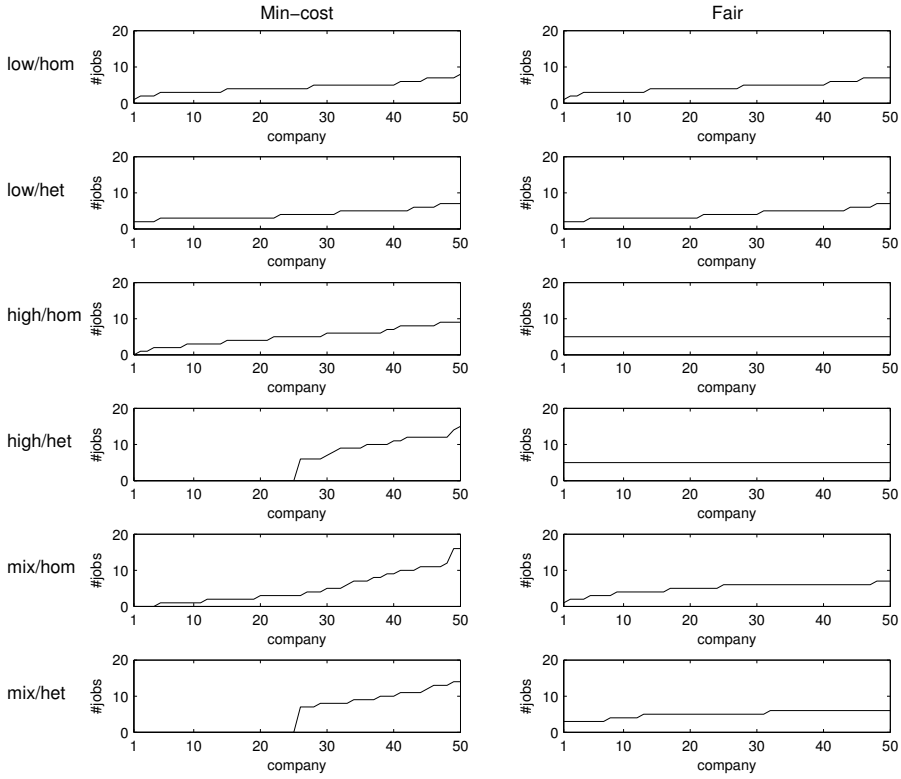


Figure 2.6: Job distributions in the minimum-cost and fair solutions for each scenario with 5% capacity, sorted in ascending order by number of assigned jobs, with companies on the horizontal axis and the number of assigned jobs on the vertical axis.

of the instances in which the highest POF was found because there were many cases with a POF of 0% in some scenarios. The allocations are sorted in nondecreasing order so that they represent the fairness vector. We can see that in all cases the fair job distribution is smoother than the minimum-cost distribution, which is exactly what we desire.

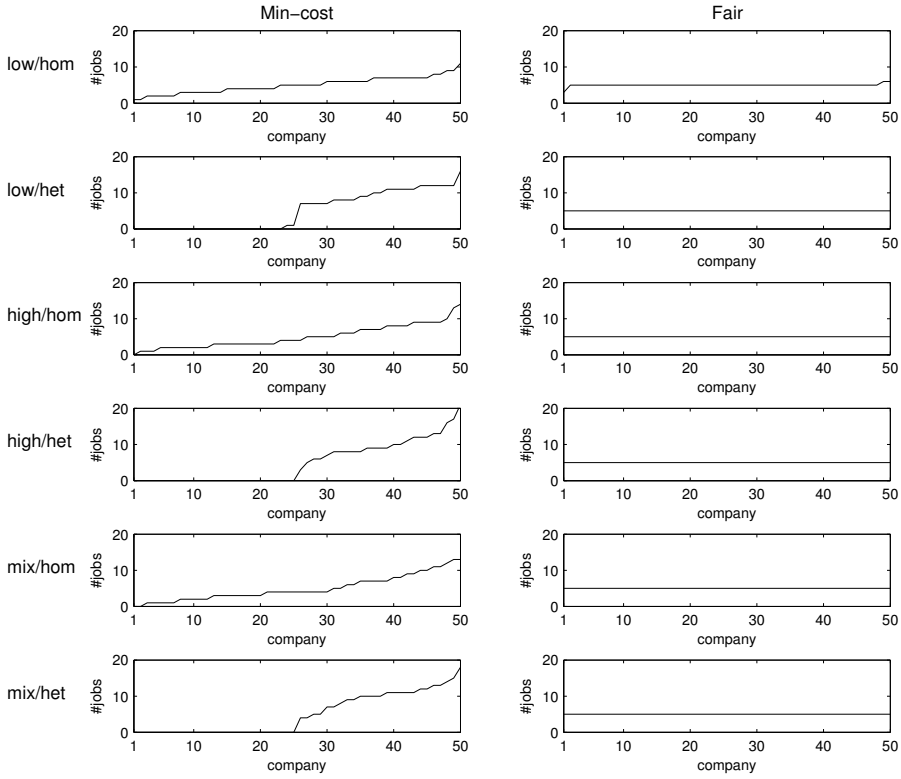


Figure 2.7: Job distributions in the minimum-cost and fair solutions for each scenario with 10% capacity, sorted in ascending order by number of assigned jobs, with companies on the horizontal axis and the number of assigned jobs on the vertical axis.

For the low/hom and low/het scenarios in the 5% capacity case, we can see that the job distributions in the minimum-cost and fair solutions do not differ much, with the exception of a few companies getting one job more, or less. This further supports our claim that in these scenarios it is often the case that the fair solution is similar to the minimum-cost solution because there exists only a few feasible allocations.

In the high-competition scenarios (high/hom and high/het) for both the 5% and 10% capacity cases, we can see that the fair allocation assigns each company the same number of jobs. This is possible because all companies have many bids, which results in much leeway while reallocating. In the low- and mixed-competition scenarios we can see that due to the inclusion of low-competition companies that do not bid on many jobs, it can still happen that certain companies get assigned fewer jobs than others. This is due to capacity restrictions and to the fact that it is simply not feasible for our proposed algorithm to assign more jobs to those companies. This effect can therefore be seen to be more prominent when there is a lower capacity (5%).

In the homogeneous cost scenarios (low/hom, high/hom and mix/hom) of the 10% capacity case, we can see that in the minimum-cost solution there are a few companies that do not get assigned any jobs or are assigned only a few jobs. This is primarily due to their higher individual costs compared to other companies. However, in the fair solution all companies are allocated roughly the same number of jobs. A few companies will receive fewer jobs than others, but this is due to capacity restrictions as explained above. Given that there are no or just minimal differences between the costs of the minimum-cost and the fair solutions in the homogeneous cost scenarios, the price of fairness when costs are homogeneous is minimal. This is as expected. Reallocating jobs does not come at a significant price because the cost for a job is similar among all companies. A similar effect can be seen in the homogeneous cost cases in the 5% capacity case. However, not all companies get the same number of jobs due to capacity restrictions being more prominent.

With heterogeneous costs, the first 25 companies in the minimum-cost solution have been allocated only a few jobs or even none at all because of the higher costs these companies have (except for the 5% capacity low/het scenario, which is due to capacity restrictions). However, in the fair solution, these companies do get a significant number of jobs, although this comes with a hefty POF, which can get up to as much as 17.27.

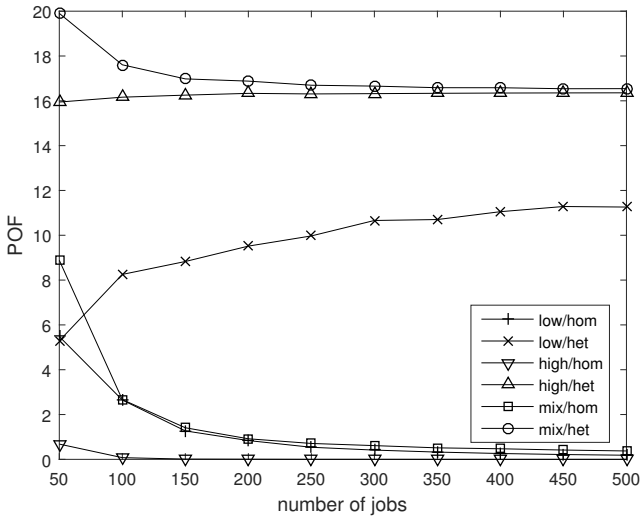


Figure 2.8: Line charts of average POF over various numbers of jobs with 10% capacity, with the number of available jobs in an experiment and the average price of fairness over 100 experiments on the horizontal and vertical axes, respectively.

2.5.4 Results: varying number of jobs

We investigate the effect of the number of jobs on the price of fairness. We vary the number of jobs from 50 to 500 while keeping the same number of companies of 50. For each scenario and number of jobs we run 100 experiments and take the average price of fairness over these experiments. The results are displayed in Figure 2.8.

We can see that for scenarios with homogeneous costs the price of fairness can be rather high when the number of jobs is low, as much as 8.88 in the mix/hom scenario. This can be accredited to the lack of flexibility in allocation when there is a small number of jobs. The distribution from which the costs of jobs are drawn may be the same for all jobs, but there is still variation in the costs. With a small number of jobs this variation plays a larger role when reassigning jobs from the minimum-cost solution to a fair solution, especially when low-competition companies that bid on only a few jobs are present. Even if low-competition companies had a high bid on a particular job, if it is only one of the few jobs they bid on, the job needs to be

allocated to these same bidders in the fair solution. This raises the total cost in the fair solution. We notice this effect particularly in the mix/hom scenario. A job that was assigned to a high-competition company in the minimum-cost solution, where the lowest cost was chosen, can suddenly be assigned to a low-competition company that only submitted a few bids.

As the number of jobs increases, the number of bids also increases, providing more flexibility for reallocation when a fair solution needs to be constructed. There will be a bigger chance of having a bid for the same job from another company with similar cost as the company in the minimum-cost solution. The average price of fairness decreases when the number of jobs increases and eventually the price of fairness seems to stabilize close to zero.

For scenarios with heterogeneous costs the effects are more complicated. Due to heterogeneous costs, jobs that were allocated to cheap companies (ones that bid between 30 and 50) in the minimum-cost solution need to be reallocated to expensive companies (ones that bid between 40 and 60) in the fair solution. This means an average increase of 10 in the cost per reallocated job. When the number of jobs increases, the share of reallocated jobs increases as well, thus increasing the total cost. This effect can be seen in particular in the low/het scenario, where the average price of fairness increases as the number of jobs increases. In the high/het scenario this effect is not as prominent, because there is high competition and thus there are many bids to choose from. Then there is a substantial chance that there exists a bid from another company that is similar in cost.

For the mix/het scenario, we have to keep in mind that high-competition companies have bids between 30 and 50, whereas low-competition companies have bids between 40 and 60. This means that when a job from a high-competition company in the minimum-cost solution needs to be reassigned to a low-competition company in the fair solution (something that also happens in the mix/hom scenario) the cost will increase by 10 on average. This results in a less steep decline in the average price of fairness compared to the mix/hom scenario when gradually going from 50 jobs to 200 jobs. The increase in costs resulting from reallocating jobs to companies with completely different costs is especially noticeable when there are fewer jobs. This increase in costs due to reallocation weighs more in the mix/het scenario than the effect of the average increase of 10 when switching from a high-competition company

to a low-competition one. The average price of fairness is therefore declining in the mix/het scenario in contrast to the low/het and high/het scenarios where the price of fairness increases with the increase in the number of jobs.

For all scenarios with heterogeneous costs it seems that the average price of fairness eventually stabilizes as the number of jobs increases. This again can be accredited to the flexibility that the increasing number of jobs, and therefore bids, introduces for reallocation when a fair solution needs to be constructed. Reallocations become more efficient and will eventually weigh up against the increase in costs due to the increased amount of reallocations.

2.5.5 Results: varying number of companies

In order to investigate the effect of the number of companies on the price of fairness we vary the number of companies from 25 to 100 in increments of 25, while now fixing the number of jobs to 250. For each scenario and number of companies we run 100 experiments and again take the average price of fairness. The results are shown in Figure 2.9.

We notice that for the scenarios with homogeneous costs the price of fairness is rather low, ranging from 0.00 to 1.48 in the mix/hom scenario. The addition of more companies does not have much effect, as the costs are similar among all companies. This creates more flexibility for the fair allocation. Only when the number of companies is low in the mix/hom scenario, there is a slightly higher price of fairness due to lack of flexibility to allocate the jobs to companies with similar costs in the fair allocation.

When we look at the scenarios with heterogeneous costs, we can see that for the high/het and mix/het scenarios the price of fairness seems to decrease as more companies participate, which is again due to added flexibility as the same number of jobs can be distributed among more companies with many more bids. The seemingly odd occurrence of a slightly lower price of fairness at 25 companies compared to the case with 50 companies can be explained by the lack of competition. When we take a look at the actual costs of the minimum cost allocations, we can see that it is slightly higher in the case of 25 companies than it is when there are 50 companies. Because of the limited number of companies, there is not much competition between the bids.

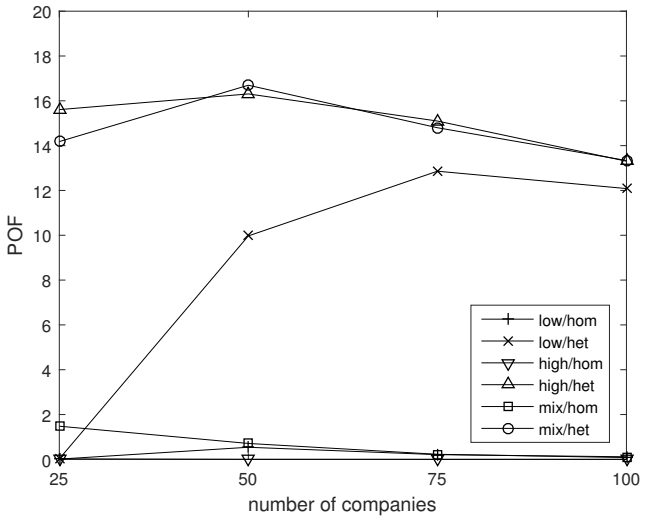


Figure 2.9: Line charts of average POF over various numbers of companies with 10% capacity, with the number of companies in an experiment and the average price of fairness over 100 experiments on the horizontal and vertical axes, respectively.

However, the costs for the fair allocations with 25 companies is similar to those in the cases with more companies, yielding a lower price of fairness.

The results of the low/het scenario stand out the most. It seems that the minimum cost and the fair allocations have similar costs when there are 25 companies, having an average price of fairness of 0.02. The price of fairness then increases substantially to 9.97 and 12.86 as the number of companies increases to 50 and 75 companies, respectively. It decreases again to 12.08 when the number of companies is further increased to 100. The average price of fairness of 0.02 with 25 companies can be easily explained when we take a look at the allocations. It seems that due to the small number of bids with only 25 companies and low competition, the fair allocation is often exactly the same as the minimum cost allocation. There is simply no other allocation possible. As the number of companies increases, the number of bids also increases, adding more leeway for the fair allocation. With 50 companies the number of bids seems to be sufficient in order to distribute the jobs to companies evenly.

However, the number of bids is still relatively low, resulting in the costs of the minimum cost allocations to be much higher than in the case of 75 or 100 companies. At the same time, the costs of the fair allocations gradually decrease as the number of companies increases. The decrease in costs of the minimum cost allocations is much heftier than the decrease in costs of the fair allocations, which results in the increase in price of fairness. Going from 75 to 100 companies the number of bids increases again, lowering the costs for the minimum cost allocations slightly, while the costs for the fair allocations decrease more with the added bids. This finally results in a slight drop in the price of fairness.

2.6 Conclusions and discussion

In task or job allocation problems there are many ways to assign jobs to all interested parties. The most common way is to minimize the costs of such allocation by only considering the cheapest companies. In this chapter, instead of just focusing on costs, we take into account the job distribution over companies. We try to allocate jobs to all participating parties as fairly as possible in terms of the number of allocated jobs. This additional criterion is particularly relevant in our motivating example, an inter-terminal transport problem (ITT) in the port of Rotterdam, where we want to use the trucks already present at the port to execute inter-terminal transport jobs. Because such a job allocation will be repeated daily, it is crucial to give those companies incentives to be involved in this activity by assigning them jobs based not only on their costs but also on ensuring some market share, i.e., allocated jobs.

To meet the new fairness criteria in task allocation, we developed a polynomial-time optimal method consisting of two novel algorithms: IMaxFlow, which uses a progressive filling idea, and FairMinCost, which smartly alters the structure of the problem. The output of these two algorithms is a max-min fair task allocation with the least total cost. In the experiments we looked at several scenarios for both the jobs that are being auctioned, and the companies who are bidding on the jobs. From the results of the experiments we find that in the situation where the costs among companies are similar, implementing fair allocations comes with almost no extra cost for the task owner. When the prices are highly volatile however, the auctioneer may

need to pay more for the fairness. When there are relatively few jobs, the price of fairness will usually be relatively high due to the lack of flexibility in reallocating jobs. As the number of jobs increases, the price of fairness will stabilize due to the flexibility granted by the increase in the number of bids. Similarly, the more companies are participating, the more bids there will be, resulting in more flexibility for reallocation and a lower price of fairness. However, the price of fairness can also be low when there are only few companies. This is then mainly due to lack of flexibility for reallocation, so that the fair allocation is similar to the minimum cost allocation. This means that the number of participants should be sufficient in order to have the desired flexibility needed for reallocation. The auctioneer should contemplate whether the fairness in the allocation is worth the extra cost. It is necessary to investigate specific cases regarding the price of fairness.

We made certain assumptions in this work because we had a real case of the ITT problem in the port of Rotterdam in mind. Some of these assumptions can be relaxed to some extent. For instance, we assumed that each task can be completed within one time unit. If the tasks have different durations, we can normalize them by using a time unit large enough to encompass the task with the longest duration. We may lose some efficiency by doing this, but it makes the problem solvable using our proposed algorithms. Furthermore, we assume that the tasks are independent. One way to tackle interdependent tasks is to make them available in subsequent time periods, i.e., make sure one task has been executed before the next one is made available for execution.

The focus of this research is on designing efficient algorithms for finding fairest task allocations. Auctions are used in this research as a way to collect local information from the participants. This information is then used as input in the task allocation problem. We do not consider the bidding behavior of the bidders in this chapter. However, bidders may choose to misreport their inputs in an attempt to affect the allocation in their favor. In order to incentivize the bidders to bid truthfully, the mechanism design aspect of the auction needs to be studied as future research (Van Der Krogt et al. 2008). In addition, we have only looked at one quantification of fairness in this research, in which we only consider the number of tasks in an allocation. As we have seen in the literature, there are many definitions of fairness and many different quantifications of fairness (Ogryczak et al. 2008, González-Pachón

and Romero 2016). Furthermore, our implementation of fairness is useful for giving companies equal market share, presumably resulting in more satisfied participants, which can be useful for long-term relationships between companies and auctioneer. Some companies might not be satisfied with this fair allocation, as they would rather maximize their own market share. However, the use of a fair allocation is exactly to counter certain companies gaining too big of a market share, suppressing other smaller participants. The idea behind the fair allocation is to not maximize the welfare of one or a few participants, but to maximize the overall welfare of all participants, including the auctioneer. The auctioneer can decide which quantification of fairness to use that best suits their goals. When considering game theoretical properties of the auction mechanism, another quantification of fairness might prove to be better in attaining the desired properties. Investigating different quantifications of fairness with mechanism design will be an interesting future direction.

In many real-world cases, the bids from one company can be combinatorial, that is, the cost of receiving two jobs is strictly smaller than the total cost of executing the same two jobs separately. If this combinatorial property exists between jobs, the task allocation problem becomes NP-hard (Cramton et al. 2006). For example in the ITT problem, if one job is to transport some goods from location A to B, and another job is to ship some goods from B to C, it seems that giving both jobs to one company leads to smaller costs than letting two companies execute the two jobs separately. After consulting with a port manager, we find out that the margin of these two instances is so narrow that we gladly ignored the combinatorial property. The advantage of this is that we now have a polynomial-time algorithm to compute the optimal allocation in terms of fairness and cost. For the cases where the tasks have a high degree of complementarities our proposed algorithms cannot be directly applied. Adapting our algorithms for solving such cases is open for further research.

Even though our work has been inspired by the situation in the port of Rotterdam, the described setting is not unique to this application. This work can be applied to many other task allocation problems in which the centralized planner wants to enforce some kind of fairness among the agents. Due to the rise of the so-called sharing economy (Goldman and Gorham 2006, Belk 2014), collaborative consumption in transport, for example car-sharing, has gained interest in the past years. The main concern in this area is on where to station the shared-use vehicles

(Fan et al. 2008, Shaheen et al. 2010, Kek et al. 2009). However, online platforms for collaborative consumption in transport have recently been upcoming. In these platforms participants are free to join or leave as they please. One might think of taxi services that are operated by civilians. Another application would be in airport slot allocation. In this area, although not as dynamic as in the cases of the port and taxi services, it is important to allocate slots to airlines both efficiently and fair, as such to motivate new entrants (Castelli et al. 2011, Condorelli 2007). It would be interesting to see whether our methods can be used in those applications.

Appendix

2.A Proof of Theorem 2

Proof. IMaxFlow starts with capacity 0 for all company-sink edges and it adds only 1 more capacity at each iteration. Thus, IMaxFlow takes at most $\max_{k\in\mathcal{K}}(N_k) < J$ iterations, and in each iteration there are at most K steps. In each step, a maximum flow algorithm is called. In our case, this is the Edmonds-Karp algorithm, which runs in time $O(|V|\,|A|^2)$. The flow network G consists of at most 2 (source and sink) $+J + JT + TK + K$ nodes and at most $J + JT + JTK + TK + K$ arcs. This results in a running time of $O(JK(JT + TK)(JKT)^2) = O((J^4K^3T^3) + (J^3K^4T^3))$. □

2.B Proof of Theorem 4

Proof. The Goldberg-Tarjan algorithm is known to terminate after $O(|V|\,|A|^2 \log(|V|))$ iterations, with Karp's algorithm having a running time of $O(|V|\,|A|)$. This results in a $O(|V|^2\,|A|^3 \log(|V|))$ algorithm for solving the second stage of the MFMCA problem. In G' there are $\phi_K - \phi_1 < J$ dummy layers. In each dummy layer the number of dummy jobs is upper bounded by K. The number of vertices in each dummy layer is then at most $3K$. This results in the number of vertices being upper bounded by $(JT+TK)$ for the original problem P and by JK for the dummy part of problem P', for a total of $JT + TK + JK$. The number of arcs $|A|$ is upper bounded by JTK for

P and by JK^2 for the dummy part of problem P', for a total of $JTK + JK^2$. Hence, FairMinCost runs in time $O(J^3K^3(K+T)^3(JK+JT+KT)^2 \log(JK+JT+KT))$. $\square$

Chapter 3

Participation behavior and social welfare in repeated task allocations[1]

3.1 Introduction

Task allocation problems have focused on achieving one-shot optimality, which typically aims at finding the minimum cost allocations (Weerdt et al. 2012). In Chapter 2 a fair task allocation problem has been studied, where we propose a fair task allocation algorithm that assigns companies jobs based not only on their costs but also tries to allocate jobs to all participants as fairly as possible. We have demonstrated the benefit of factoring fairness into task allocation. In the experiments, among the majority of test instances, fairness comes with a very small price in terms of cost. The motivation of developing fair task allocation algorithms was inspired by an actual transportation situation in the port of Rotterdam in the Netherlands, where many small inter-terminal tasks need to be assigned to companies who have trucks that are already present in the port. Those trucks that come from the hinterland to

[1]This chapter is based on Ye and Zhang (2016), Ye et al. (2017b).

drop or pick up containers often have spare time in between tasks. Hence, terminals could take advantage of these idle trucks by providing them with jobs that they can perform within the port while waiting for their next scheduled job. This benefits both the truck operators and the port operator, in addition to using readily available resources, which increases the utilization rate of existing trucks. Furthermore, this means that less or even no new trucks are needed to perform the jobs, which results in being a more durable approach to meeting the transport need within the port. Because such allocations will be repeated daily and any truck company that is present in the port is free to participate, it is crucial to encourage those companies to participate in this activity.

We hypothesize in Chapter 2 that due to psychological factors, using an allocation algorithm with fairness as a main criterion will encourage companies' participation in the repeated task allocation game. More participants ensure more supplies in the system, which will eventually lead to a higher social welfare. The objective of this chapter is to test this hypothesis. We study a repeated task allocation problem. Besides the inter-terminal transportation example of the port of Rotterdam, another example of repeated task allocation would be private taxi services, in which any party is free to take up a job, different from traditional taxi services. An incoming job may be proposed to several drivers in the neighbourhood, but eventually only one of them will be assigned the job. In these settings, participants share their idle resources, and therefore, it is important to ensure some portion of the market share to the players to encourage their participation.

We consider agents to be not completely rational. Hence, the assumption that an agent will participate in the game as long as its expected utility is non-negative may not hold anymore. Instead, we take into account psychological factors of agents that may curb the decision to participate in two ways. First, we model agents' participation based on prospect theory (Tversky and Kahneman 1992). Second, we propose a model of agent optimism based on a fuzzy connective. In fuzzy decision theory there is a large number of connectives, i.e., aggregation operators, which can be used for modeling different types of decision making behavior. The generalized averaging operators are especially interesting, since they can be used to model human decision making behavior (Kaymak and van Nauta Lemke 1998, Kaymak 2017). Instead of using prospect theory to incorporate human decision making behavior, we will now

use the generalized averaging operators to incorporate an agent's utility and previous experiences into its current participation decision. Each agent will have its own optimism level, which is based on its relative allocation compared to the other agents. This optimism level will transform an agent's utility in order to determine its participation probability that will influence the participation decision in a given round. In turn, we also look at the effect of agents' participation on the social welfare, which is measured by the allocation quality.

The outline of this chapter is as follows. In Section 3.2 we present a brief overview of existing literature on repeated games, prospect theory, and fuzzy decision theory. Section 3.3 introduces the problem setting and in Section 3.4 we show how we use prospect theory and the generalized averaging operator to model agents' decisions on participating in each round of the games (Section 3.4). This participation probability is derived based on the previous allocation outcomes, and particularly, on an agent's perception on its received proportion in comparison to other agents. In Section 3.5 we conduct simulation experiments to investigate how the allocation influences agents' decision to participate by using two task allocation algorithms, of which one only looks at optimality in terms of costs, and the other looks at optimality in terms of primarily fairness and secondarily costs (Chapter 2). We also look at the effect of agents' participation on the social welfare in each round, and its effect on the overall social welfare, where the social welfare is measured by the allocation quality. Finally, Section 3.6 contains the concluding remarks.

3.2 Literature review

Repeated problems have been studied in the fields of game theory (Benoit and Krishna 1985, Mailath and Samuelson 2006) and auctions (List and Shogren 1999, Rothkopf 1999). In these studies, agents are assumed to be rational, i.e., as long as their expected utility is non-negative, they do not opt out of the game. Therefore, with an individual rational mechanism, the participating parties are fixed (e.g., Nisan and Ronen (1999)). In our work, we discard the assumption of rationality, and model agents' different participation behaviors by linking their optimism or pessimism motives with the outcomes of the previous games. We do this using both

prospect theory, and a fuzzy set connective from fuzzy decision theory. Prospect theory has been widely studied in behavioral economics (Camerer 2004, Lahdelma and Salminen 2009). It is a behavioral model that shows how people handle decisions that involve risk and uncertainty. However, this behavioral theory is rarely used in task or resource allocation problems. Fuzzy sets have been used to model human decisions, and several fuzzy connectives for this purpose have been proposed based on experimental work (Kovalerchuk and Taliansky 1992, Thole et al. 1979, Zimmermann and Zysno 1980). The generalized means are of particular interest, as they easily allow for modeling a wide range of the degree of compensation (van Nauta Lemke et al. 1983, Dyckhoff and Pedrycz 1984).

3.3 Problem definition

First, we revisit the task allocation setting that we have studied in Chapter 2. We assume that the set of available jobs to be distributed among agents is known in advance by the central planner. We assume a set of time periods $\mathcal{T}$, consisting of T time periods. The set of jobs, denoted by $\mathcal{J}$ consisting of a total of J jobs, comes with an earliest available time and a latest completion time for each job. We assume that jobs are independent. We define for each job $j_i \in \mathcal{J}$ its possible starting time as a mapping: $\mathcal{J} \times \mathcal{T} \mapsto \{0, 1\}$. When it is clear from the context, we abuse the notation and use j_i^t to denote that job j_i is available at time period $t \in \mathcal{T}$. Once the set of jobs $\mathcal{J}$ together with their possible starting times has been made available, a set of companies $\mathcal{K}$, consisting of K companies, may bid on individual jobs. In addition to the selection of jobs that a company $k \in \mathcal{K}$ wishes to perform, the company also needs to provide their available capacity n_k^t in time period t in which it is able to perform the jobs. We assume that each job takes up one unit of capacity and can be completed within one time unit. Furthermore, the company k needs to provide its desired compensation (or cost), $c(j_i, k)$, for the bid job $j_i \in \mathcal{J}$. A bid, B_k, from a company k is thus a tuple: $\langle \boldsymbol{c}_k, \boldsymbol{n}_k \rangle$, where $\boldsymbol{c}_k$ is a set of costs $c(j_i^t, k)$, and $\boldsymbol{n}_k$ is a set of capacities n_k^t. Once all bids from the bidding companies $\mathcal{K}$ have been collected, the auctioneer determines a task allocation $\pi : \mathcal{J} \times \mathcal{T} \times \mathcal{K} \mapsto \{0, 1\}$. If a job j_i is allocated, it will result in a fixed value V, which indicates how much a task

Table 3.1: The bids of three companies include desired jobs in each time period and their associated compensation.

	t_1	t_2	t_3	t_4	t_5
k_1	$j_1 : 20$				
k_2	$j_1 : 30$	$j_2 : 40$ $j_3 : 25$			
k_3	$j_1 : 10$	$j_2 : 20$ $j_3 : 20$	$j_3 : 25$ $j_4 : 25$	$j_2 : 30$ $j_4 : 20$	$j_5 : 20$

is worth to the auctioneer. For all unallocated jobs, their values are set to 0. The social welfare U given an allocation π is defined as the difference between the total value of allocated tasks and the total costs of performing the allocated tasks. In a typical task allocation problem, the objective is to maximize the social welfare by choosing an optimal allocation. Note that for this one-shot task allocation problem, we can use any existing min-cost max-flow algorithm to find the optimal allocation.

In Chapter 2 a fair algorithm is developed to ensure a max-min fair allocation to agents. The max-min fairness principle means that given a total of Z jobs, the number of jobs for any agent cannot be increased by at the same time decreasing the number of jobs of other agents that have the same number of jobs or less. Intuitively speaking, we want to have an allocation that distributes the set of jobs among the agents as evenly as possible. To meet the fairness criterion a polynomial-time fair method is developed consisting of two novel algorithms: IMaxFlow, which computes a max-min fair vector, i.e., the most even distribution over agents given all bids, and FairMinCost, which finds the minimum cost allocation that satisfies the given max-min fair vector. The output of these two algorithms is a max-min fair task allocation with the least total cost. We now use the following example to illustrate the algorithms used in Chapter 2 to obtain the minimum-cost and fair allocations.

Example 5. *Suppose we have* 5 *jobs, all having a value of* $V = 100$. *The jobs can be done in certain time periods and three companies submit their bids, as shown in Table 3.1. Any min-cost max-flow algorithm results in* $\pi_{mincost}$ *assigning* j_3^2 *to* k_2 *and* $j_1^1, j_2^2, j_4^4, j_5^5$ *to* k_3, *with a total compensation of* 95, *and social welfare of* 405. *For the fair allocation, using the algorithms in Chapter 2 we obtain the max-min fair allocation vector* $\boldsymbol{\phi} = (1, 1, 3)$ *using IMaxFlow. Thereafter, using FairMinCost, we*

obtain the fair allocation π_{fair}*, which assigns* j_1^1 *to* k_1*,* j_3^2 *to* k_2 *and* j_2^2, j_4^4, j_5^5 *to* k_3*, with a total compensation of* 105*, and social welfare of* 395. ■

In this chapter, we will extend the one-shot task allocation problem in Chapter 2 to a multi-round task allocation problem. We introduce round $r \in \mathcal{R}$. We then have tasks $j_{i,r} \; \forall \; j_{i,r} \in \mathcal{J}_r$, with $\mathcal{J}_r$ the set of jobs in round r, $\mathcal{J}_r \subseteq \mathcal{J}$. In the repeated task allocation game with R rounds, the objective is to maximize the social welfare over all rounds, that is, to maximize $\sum_{r=1}^{R} U_r = \sum_{r=1}^{R} \sum_{j_{i,r} \in J_r, \pi_r(j_i,\cdot,\cdot)=1} |j_{i,r}| \cdot V - c(j_{i,r}, k)$, where π_r is an allocation in round r. In addition, we assume agents decide for themselves whether they would like to participate in bidding in a certain round or not. This participation decision will be modelled as a participation probability p_{kr}, which is dependent on earlier allocation outcomes. The repeated task allocation in a round r goes as follows. After the auctioneer announces the available tasks, agent k decides on whether to participate or not by computing its participation probability p_{kr}, and if participating, it submits the bid $b_{r,k}$ based on its observed capacities and compensations. As we do not study agents' bidding strategies in this chapter, we simply assume agents submit their bids based on their true values. The auctioneer then decides on the task allocation π_r and the payments $c(j_{i,r}^t, k)$. Finally, agent k observes all participants' bids and the outcome π_r.

3.4 Modeling participation behavior

We consider two different ways of modeling an agent's participation probability p_{kr} to show the difference in impact an agent's participation decision can have on the allocation, dependent on how the agent behaves. First, we use prospect theory, and second, we use a generalized averaging operator from fuzzy decision theory.

3.4.1 Prospect theory

Prospect theory (Tversky and Kahneman 1992) demonstrates that people view their expected utility not in absolute terms, what one would expect from expected utility theory, but rather relative to a reference point. In addition, it indicates that people are loss-averse, where they would be more willing to take risks in order to avoid a loss, and they would avoid taking risks if it concerns a gain. There is a distinction

between two phases in the choice process: (1) the editing phase, where a simpler representation of the outcomes of alternatives is obtained, as they are coded as gains or losses relative to a reference point; and (2) the evaluation phase, where the edited prospects are evaluated and the prospect of highest value is chosen. The overall value of an edited prospect is expressed by a weighting function and a value function. In Tversky and Kahneman (1992), the authors propose the form of the value function given by

$$v(x) = \begin{cases} x^{\alpha} & \text{if } x \geq 0 \\ -\lambda(-x^{\beta}) & \text{if } x < 0 \end{cases} \tag{3.1}$$

where $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ are coefficients determining the concavity and convexity for gains and losses, respectively, and $\lambda > 1$ is the loss-aversion coefficient.

The participation probability p_{kr} of agent k in round r is dependent on their experience in previous rounds. To model the participation decision using prospect theory, we use the average proportion in the previous round over all companies as the reference point in the editing phase, and a positive (or negative) difference between a company's proportion in the previous round and the average proportion as a gain (or loss). The intuition is that if an agent feels being treated worse in comparison with others, it might be more uncertain and will care less about participating again, because the time and effort put in the preparation when participating can then be seen as a loss. More formally, denote z_{rjtk} and x_{rjtk} as the binary variables that indicate whether in round r agent k has participated in bidding on job $j_{i,r}$ or is assigned $j_{i,r}$ in the allocation π_r, respectively. For ease of notation we use round r directly in the subscript of the variables. Denote $\mathcal{K}_r^+ \subseteq \mathcal{K}$ as the subset of agents where $k^+ \in \mathcal{K}_r^+$ has $\sum_{jt} z_{rjtk^+} > 0$ in round r. For every agent k that has participated in the round prior to round r, $k^+ \in \mathcal{K}_{r-1}^+$, we can use the proportion of number of jobs won in round $r-1$ over the number of bids submitted in round $r-1$, ψ_{kr}, as a measurement of the possible gain and loss of an agent $k \in \mathcal{K}$ in round r, see Eq. (3.2). We set the reference point to be the average proportion over all companies $k' \in \mathcal{K}$, and we normalize the difference between the proportion of agent k and the average proportion, as in Eq. (3.3). For the evaluation phase, we use the value function in Eq. (3.4) to obtain the probability of agent k bidding in round r. In

Eq. (3.5) the prospect probability $\hat{p}(\psi'_{kr})$ is scaled into the probability interval $[0, 1]$.

$$\psi_{kr} = \frac{\sum_{j,t} x_{(r-1)jtk}}{\sum_{j,t} z_{(r-1)jtk}} \tag{3.2}$$

$$\psi'_{kr} = \frac{\psi_{kr} - \frac{1}{K}\sum_{k' \in \mathcal{K}} \psi_{k'r}}{\max_{k^+ \in \mathcal{K}^+_{r-1}} (|\psi_{k^+r} - \frac{1}{K}\sum_{k' \in \mathcal{K}} \psi_{k'r}|)} \tag{3.3}$$

$$\hat{p}(\psi'_{kr}) = \begin{cases} {\psi'_{kr}}^{\alpha} & \text{if } \psi'_{kr} \geq 0 \\ -\lambda(-{\psi'_{kr}}^{\beta}) & \text{if } \psi'_{kr} < 0 \end{cases} \tag{3.4}$$

$$p'(\psi'_{kr}) = \max(p_{LB}, \frac{\hat{p}(\psi'_{kr})}{2} + 0.5) \tag{3.5}$$

When an agent's proportion is equal to the average proportion, we assume that every agent is indifferent on participating, therefore, $p'(\psi'_{kr}) = 0.5$. When an agent is allocated more tasks than others on average, then the participation probability increases, as the agent will feel more certain in participating. However, when an agent is allocated less tasks than others on average, the participation probability decreases. In order for the probability to not become negative we set a lower bound p_{LB}, so that agents will still be able to come back and participate in later rounds although with a very small probability. If an agent has not participated in bidding in the previous round, we set $p'(\psi'_{kr}) = p_{LB}$. In order to take into account the experience from previous rounds into the participation probability, we will apply simple exponential smoothing. Thus, the participation probability for agent k in round r is calculated as

$$p_{kr} = \gamma p'(\psi'_{kr}) + (1 - \gamma) p(\psi'_{k(r-1)}), \tag{3.6}$$

where γ is the smoothing factor. In the first round, $r = 1$, we assume that all agents will participate in bidding.

Example 6. *In Example 5 we have seen three agents who bid on five jobs. Consider this as round 1 with participation probabilities* $p_{k_i 1} = 1.00$ *for all three agents. We choose* $\alpha = \beta = 0.88$ *and* $\lambda = 2$*, as they are commonly adopted in prospect theory, and* $p_{LB} = 0.01$ *and* $\gamma = 0.5$*. In round 2, we obtain* $\psi_{k_i 2} = (\frac{1}{1}, \frac{1}{3}, \frac{3}{8})$*, for agent* k_1*,* k_2 *and* k_3*, respectively. The reference point becomes* 0.57*, and the correspond-*

ing differences are $(0.43, -0.24, -0.19)$. *The normalized proportions become* $\psi'_{k_i 2} = (1, -0.55, -0.45)$. *Using (3.4) and (3.5), we obtain* $p'(\psi'_{k_i 2}) = (1.00, 0.01, 0.01)$. *Eventually, the participation probability can be obtained through (3.6), resulting in* $p_{k_i 2} = (1.00, 0.505, 0.505)$. *Agents* k_1 *and* k_3 *decide to participate in this round, bidding on 3 and 5 jobs, and are assigned 1 and 2 jobs, respectively. Using this information, we can repeat the calculations for round 3, which result in* $p(\psi'_{k_i 3}) = (0.81, 0.01, 1.00)$. *Agent* k_2 *will be very unlikely to participate in the next round of allocation.* ■

3.4.2 Fuzzy decision theory

We make use of the generalized averaging operators from fuzzy decision theory in order to aggregate the utilities of previous m rounds, taking into account an agent's personal optimism/pessimism, in a similar fashion as in Lovric et al. (2009). The generalized averaging operators are given by

$$p_{kr}(s_{kr}) = \left\{ \frac{1}{\min(m, r-1)} \sum_{q=\max(1,r-m)}^{r-1} \mu_{kq}^{s_{kr}} \right\}^{1/s_{kr}}, \tag{3.7}$$

for $s_{kr} \in \mathbb{R} \setminus \{0\}$, and

$$p_{kr}(0) = \left\{ \prod_{q=\max(1,r-m)}^{r-1} \mu_{kq} \right\}^{1/\min(m,r-1)}, \tag{3.8}$$

where p_{kr} is the participation probability for agent k in round r, and s_{kr} is a parameter indicating the optimism level of agent k in round r. A positive s_{kr} indicates that the decision making behavior is optimistic, whereas a negative s_{kr} indicates that the decision making behavior is pessimistic. The higher the s_{kr}, the closer to the maximum value of the sample $p_{kr}(s_{kr})$ will be. Whereas the lower the s_{kr}, the closer to the minimum of the sample it will be. If $s_{kr} \to \infty$ ($s_{kr} \to -\infty$), $p_{kr}(s_{kr})$ will be the maximum (minimum) of the sample, and when $s_{kr} = 1$ ($s_{kr} = -1$) we obtain the arithmetic (harmonic) mean. In the special case that $s \to 0$, we will use (3.8),

which returns the geometric mean. The decision function (3.7) holds the following properties:

- $p_{kr}(s_{kr})$ is continuous in parameter s_{kr};
- $p_{kr}(s_{kr})$ is monotonic and nondecreasing in s_{kr};
- $p_{kr}(s_{kr})$ is increasing in μ_{kr};
- $p_{kr}(s_{kr}) \in [0, 1]$ if $\mu_{kr} \in [0, 1]$.

We let the optimism parameter s_{kr} be dependent on the outcome of the previous round. Let us first denote z_{rjtk} and x_{rjtk} as the binary variables that indicate whether in round r agent k has participated in bidding on job $j_{i,r}$ or is assigned $j_{i,r}$ in the allocation π_r, respectively. For ease of notation we use round r directly in the subscript of the variables. Denote $\mathcal{K}_r^+ \subseteq \mathcal{K}$ as the subset of agents where $k^+ \in \mathcal{K}_r^+$ has $\sum_{jt} z_{rjtk^+} > 0$ in round r. For every agent k that has participated in the round prior to round r, $k^+ \in \mathcal{K}_{r-1}^+$, we can use the proportion of number of jobs won in round $r-1$ over the number of bids submitted in round $r-1$, ψ_{kr}, as a measurement of the optimism or pessimism of an agent $k \in \mathcal{K}$ in round r.

$$\psi_{kr} = \frac{\sum_{j,t} x_{(r-1)jtk}}{\sum_{j,t} z_{(r-1)jtk}} \tag{3.9}$$

This proportion does not indicate how well the agent has performed compared to its peers. Therefore, we determine the average proportion over all agents $k' \in \mathcal{K}$, and the difference of the agent's proportion with this average. As these differences may be very small, as every agent might have bid on many tasks but was allocated only a few, we normalize the differences to the largest absolute difference.

$$\psi'_{kr} = \frac{\psi_{kr} - \frac{1}{K}\sum_{k' \in \mathcal{K}} \psi_{k'r}}{\max_{k^+ \in \mathcal{K}_{r-1}^+}(|\psi_{k^+r} - \frac{1}{K}\sum_{k' \in \mathcal{K}} \psi_{k'r}|)} \tag{3.10}$$

Since $\psi'_{kr} \in [-1, 1]$, we want our optimism parameter s_{kr} to have a wider range, as otherwise we would be obtaining values in between the harmonic and arithmetic

mean. Hence we multiply ψ'_{kr} by a constant c_s to obtain $s_{kr} \in [-c_s, c_s]$,

$$s_{kr} = c_s \psi'_{kr}. \tag{3.11}$$

The membership values μ_{kr} are the utilities of the agents, which is the total compensation an agent k receives for the allocated tasks in round r. Since we would like the participation probabilities p_{kr} to lie in the interval $[0, 1]$, we need to have $\mu_{kr} \in [0, 1]$. Therefore, we take the normalized compensation, where an agent's compensation is divided by the maximum compensation of all agents in that round,

$$\mu_{kr} = \frac{U_{kr}}{\max_{k \in \mathcal{K}} U_{kr}}, \tag{3.12}$$

where U_{kr} denotes the utility, or social welfare, of agent k in round r.

Note that the calculation of ψ'_{kr} is based on agent k having participated in the previous round. However, if this is not the case, the proportion and therefore also the participation probability p_{kr} will be undefined. Since we would like to give agents the opportunity to enter and participate again in later rounds, we assign a very small participation probability p_{low}.

We assume that all agents will participate in bidding in the first round, $p_{k1} = 1, \forall\, k \in \mathcal{K}$. A participation probability of $p_{kr} = 0.5$ means that agent k is indifferent on whether to participate or not. When p_{kr} is higher, agent k is more keen on participating, whereas when p_{kr} is lower, the agent will be more likely to refrain from participating. The optimism parameter s_{kr} plays a large role in swinging this participation probability up or down.

The following example illustrates how the participation probability develops throughout the rounds using the fuzzy connective.

Example 7. *In Example 5 we have seen three agents who bid on five jobs. Consider this as round 1 with participation probabilities* $p(\psi'_{k_i 1}) = 1.00$ *for all three agents. As parameters we choose* $c_s = 5$ *and* $p_{low} = 0.01$*. In round 2, we obtain* $\psi_{k_i 2} = (\frac{1}{1}, \frac{1}{3}, \frac{3}{8})$*, for agent* k_1*,* k_2 *and* k_3*, respectively. We can now calculate the average proportion as* $\frac{1}{3}(\frac{1}{1}+\frac{1}{3}+\frac{3}{8}) \approx 0.57$*, and the corresponding differences, which are* $(0.43, -0.24, -0.19)$*. The normalized proportions become* $\psi'_{k_i 2} = (1, -0.55, -0.45)$ *using* (3.10)*. Therefore, our optimism parameters become* $s_{k2} = (5, -2.75, -2.25)$*. The membership values*

are $\mu_{k2} = \frac{(80,75,240)}{240} = (0.33, 0.31, 1)$. *Eventually, the participation probability can be obtained through (3.7), resulting in* $p_{k2} = (0.33, 0.31, 1)$. *Agents* k_1 *and* k_3 *decide to participate in this round, bidding on 3 and 5 jobs, and are assigned 1 and 2 jobs, respectively. Using this information, and the information that the obtained compensations are* $U_{k3} = (75, 0, 160)$*, we can repeat the calculations for round 3. The resulting participation probabilities are* $p_{k3} = (0.37, 0.01, 1.00)$. *Agent* k_2 *will be very unlikely to participate in this round.* ■

3.5 Experiments

We are interested in how social welfare develops over multiple rounds and how two different allocation algorithms, a min-cost max-flow algorithm and a fair algorithm, influence it. Therefore, we conduct a simulation study. We use a first-price sealed-bid auction where the allocation π_r and the number of jobs bid on from all agents will be made available after each round. This means that all bidders have information on the allocations and the number of bids from all previous rounds in order to determine their participation probability. Note that only the number of jobs bid on will be made available after a round, not the actual bids themselves, so that agents cannot adjust their own bids accordingly. We will use the same test instances as described in Chapter 2. We use $T = 10$ time periods per round. We assume the value of each task to be the same, $V = 100$. The tasks have a latest completion time, which we set to 3 time periods after the earliest time the task become available. In this scenario, which is similar to the situation in the port of Rotterdam, there are two peak hours in a day. Therefore, tasks have a 25% chance of starting at t_2 and another 25% chance of starting at t_6. If a task does not start at a peak hour, it has an equal chance to start at any time from t_1 to t_8. A bidder k has a predetermined set of tasks it is interested in. In the simulation, each task in each time period has a chance of either 0.25 or 0.75 to be selected for this set, which are indicated as the *low* and *high competition* case, respectively. For each task in this set the bidder has a bid. We will distinguish between two bid cases. The first bid case is where every agent draws their bid cost from an uniform distribution in the interval $[30, 60]$, based on the hourly wage of a driver and the fuel costs with an additional profit, which we

call the *homogeneous cost* case. In the second bid case a predetermined half of the agents will draw their bid cost from an uniform distribution in the interval $[30, 50]$, and the other half of the agents will draw theirs from the interval $[40, 60]$. The idea behind this case is that the former half of the agents who draw their bids from a lower interval represents the agents that are large and can make use of economies of scale to bid low. The latter half, on the other hand, are small agents that bid relatively high, because they cannot make use of economies of scale and have to provide a service for minimally that cost. We call this the *heterogeneous cost* case. Combining scenarios with low/high competition and homogeneous/heterogeneous cost, we construct four scenarios: *L/hom*, *L/het*, *H/hom* and *H/het*.

It depends on the participation probability, p_{kr}, whether a bidder will actually participate in bidding in a particular round r. For each round r, a uniform random number is generated for bidder k, which will be compared to its participation probability. If they decide to bid in that round, i.e., the uniformly distributed random number is lower than the value for the participation probability, they will submit the bids for all the tasks they are interested in in that round. The lower bound on the participation probability is set to $p_{LB} = 0.01$. This is to ensure that an agent who does not participate in the auction for a few rounds will still be able to participate in latter rounds and will not be excluded from the remainder of the auction due to having zero or even negative participation probability. In the first round no agent has experience with the auction yet. Therefore, all agents will have a participation probability equal to 1.00 and thus all agents will submit their bids. We set the parameters for the prospect function to $\alpha = \beta = 0.88$, $\lambda = 2$ (Tversky and Kahneman 1992), and for the smoothing factor we use $\gamma = 0.5$. For the generalized averaging operator, we set $c_s = 5$. We conduct 20 experiments for each of the four scenarios, with 50 rounds, 50 agents and 250 jobs.

3.5.1 Prospect theory

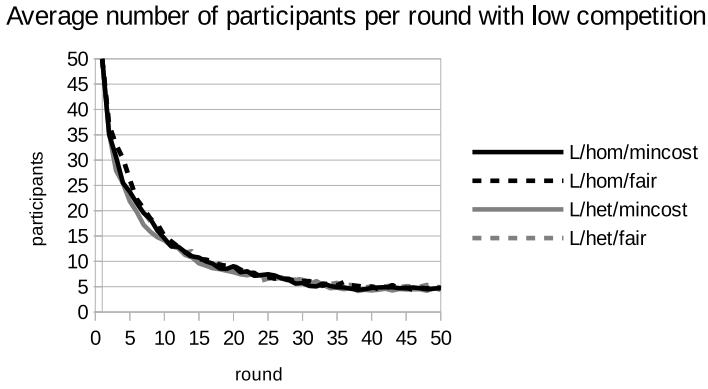
Figure 3.1 shows the average number of participants per round over the 20 experiments for 50 rounds. In the low competition case the average number of participants does not differ much between the different cost cases and allocation algorithms over the rounds. This is due to a limited number of bids, resulting in similar allocations

regardless of allocation algorithm. For the high competition case, however, the average number of participants when using the fair algorithm is substantially higher than when using the minimum-cost algorithm as the rounds progress. This is due to the jobs being allocated more evenly among agents, which results in higher participation probabilities over all rounds. The average social welfare over the 20 experiments is not very different between the different cases with low competition, as seen in Figure 3.2. In the high competition case, social welfare is substantially higher for the fair algorithm as rounds progress. This is due to the larger number of participants still present, which enables more bids for the algorithm to choose from. For the *H/het* scenario, the fair allocation obtains a lower social welfare in the earlier rounds compared to the minimum-cost allocation, but eventually surpasses it, due to the larger number of participants still present.

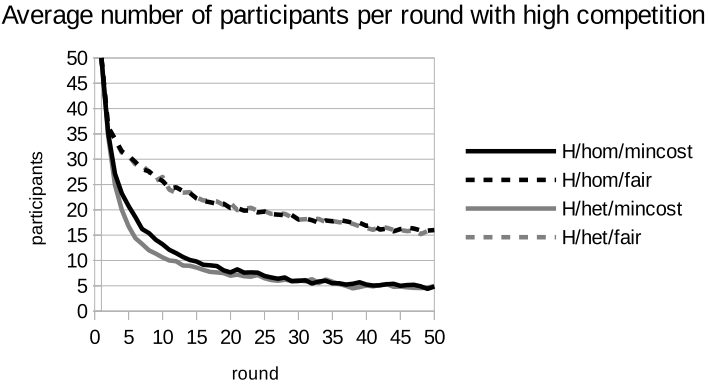
3.5.2 Fuzzy decision theory

In Section 3.5.1 we have seen that the low competition case yielded similar results using the minimum cost and the fair allocation, due to the lack of leeway in the allocation. Therefore, we only consider the *high competition* case in the experiments using the generalized averaging operator. We show the results of the first 25 rounds, as the results stabilize in the rounds thereafter.

Figure 3.3 shows the average number of participants per round over the 20 experiments. We already observe a sharp decline in the number of participants in the second round when making use of the minimum-cost algorithm, with both homogeneous and heterogeneous costs. Even though the aggregation operator only takes into account the first round, in which all agents participate, many agents are deterred from participating again in the second round due to their optimism parameter s being too low, as they obtained fewer tasks than their peers. The number of participants keeps declining and goes below an average of 5 already after round $r = 5$. When making use of the fair allocation algorithm, we can see that the decline in the number of participants is not nearly as sharp as with the minimum-cost algorithm. Even at round $r = 10$ there are still on average 15 participants, and only slowly declines after that. The average number of participants manages to stay above an average of 7 even after round $r = 25$. This is mainly due to the nature of the fair



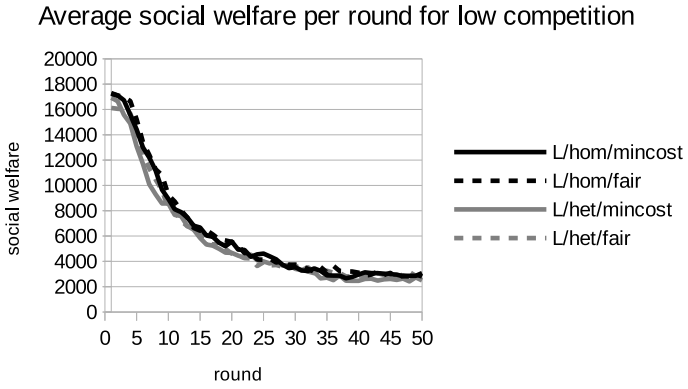
(a) Low competition



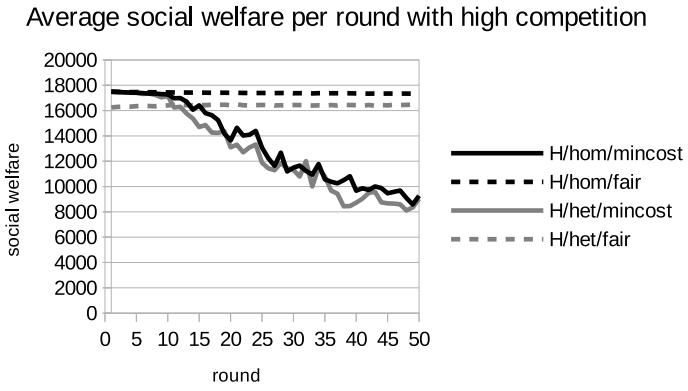
(b) High competition

Figure 3.1: Average number of participants per round over 20 experiments over 50 rounds using prospect theory.

allocation algorithm, which assigns tasks to more agents. This results in a higher optimism level among more agents, as they feel they have obtained a fair share of the allocation. This in turn also yields a higher social welfare on average among the



(a) Low competition



(b) High competition

Figure 3.2: Average social welfare per round over 20 experiments over 50 rounds using prospect theory.

agents in each round, which, together with the higher optimism level, leads to higher participation probabilities.

Consequently, a larger number of participants in each round means a higher social welfare in each round, as depicted in Figure 3.4. As the minimum-cost algorithm

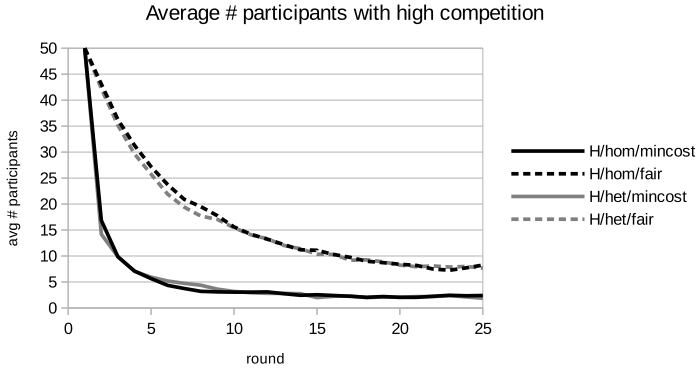


Figure 3.3: Average number of participants per round over 20 experiments over 25 rounds using a fuzzy connective.

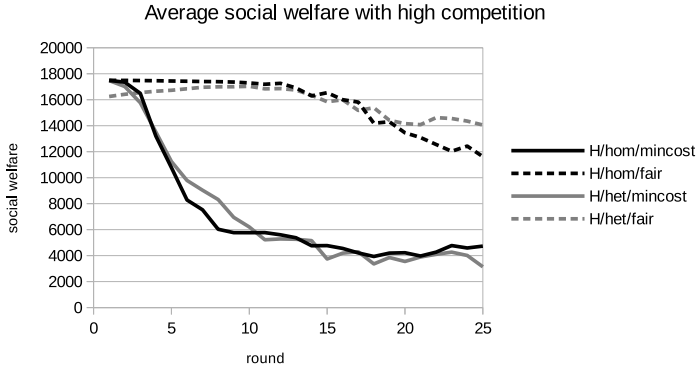


Figure 3.4: Average social welfare per round over 20 experiments over 25 rounds using a fuzzy connective.

yields a rapid declining social welfare over the rounds, tied to the drop in participants, the fair allocation algorithm yields a rather steady social welfare over the first 12 rounds. After round $r = 13$ the social welfare starts to waver and decline slowly, due to the decline in participants as well. However, even in round $r = 25$, the social welfare is approximately triple the social welfare obtained with the minimum-cost

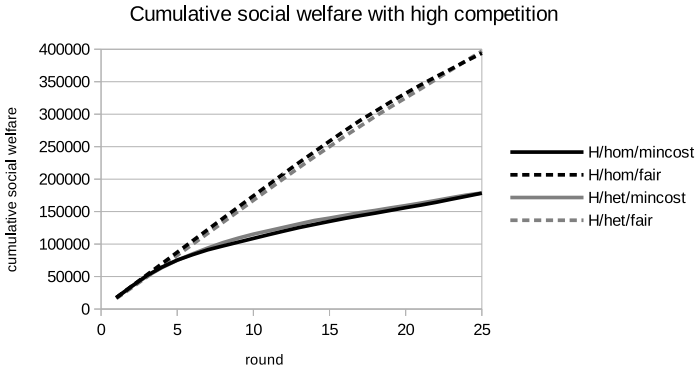


Figure 3.5: Cumulative average social welfare over 20 experiments with high competition over 25 rounds using a fuzzy connective.

algorithm. Figure 3.5 shows the cumulative social welfare over the rounds. We can see that the total social welfare obtained by the fair allocation quickly surpasses that of the minimum-cost allocation in only a few rounds. In round $r = 25$ the fair allocation obtained a total social welfare of approximately twice the amount of the minimum-cost algorithm.

Comparing these results to Section 3.5.1 using prospect theory, we observe that using generalized averaging operators results in a faster decline in participants and social welfare than when using prospect theory. In Figure 3.3 using the generalized averaging operator the number of participants at round 15 is approximately 2 using the minimum-cost allocation and 10 using the fair allocation, whereas in Figure 3.1b using prospect theory the number of participants in the respective allocations are 9 and 22. This, in turn, results in a significant drop of average social welfare in fewer rounds when using the generalized averaging operator, as seen in Figure 3.4 with an average social welfare of 6000 after 10 rounds. Using prospect theory, we can see in Figure 3.2b that the average social welfare after 10 rounds is still sitting strong at around 17000, which is close to the average social welfare of the first round. Thereafter it does drop down as rounds progress, but it does not drop as steeply as when using the generalized averaging operator. These differences can be mainly

attributed to the optimism level of agents. Fluctuations in the optimism level can cause severe changes in the participation decision, whereas with prospect theory such fluctuations only had little effect, therefore taking more time to develop noticeable effects.

3.6 Conclusion and discussion

In a repeated task or resource allocation problem with sharing nature, the participation of agents is driven not only by their expected economic gain, but also by their willingness to put effort in participating in the face of uncertainty and risk. The main contribution of this chapter is that we model agents' participation using prospect theory and generalized averaging operators from fuzzy decision theory to show the impact an agent's participation decision can have on the allocation, and how different allocation algorithms impact the number of participants and average social welfare in a repeated auction setting. Hereby we use an agent's experiences in previous rounds to determine their participation decision. The two different ways of modeling the agents' participation decision yield vastly different outcomes, which shows that the agents' behavior can have a big impact on the outcome in repeated auctions. Prospect theory has a more mellow reaction to fluctuations in the agent's optimism than the generalized averaging operator. An auctioneer needs to have a good understanding of its participants in order to model their behavior in a representative way so that meaningful results can be obtained. They can also conduct a study with a representative group of agents participating in simulated repeated auctions to figure out their behavior and whether the model using prospect theory or fuzzy decision theory, and with which parameters, is more suitable to their agents. Their behavior could also possibly be extracted from historical data on the bids submitted by their agents and their participation rate. This is left up for future research. In addition, we demonstrate that fair allocations result in more participants than minimum-cost allocations throughout the rounds when there are plenty of resources in the system, which eventually results in a higher social welfare. Therefore, an auctioneer needs to consider the long term consequences of their chosen allocation

algorithm in repeated auction settings. Giving up social welfare in the short term can result in a higher social welfare in the long term.

Allocation algorithms that take into account the participants' behaviors are especially of importance in settings like the sharing economy, which is upcoming in the past years, in which anyone is free to enter and leave as they wish. In these settings, participants share their idle resources. Therefore, it is important to encourage their participation and to yield an overall higher social welfare, which can be accomplished by ensuring some portion of the market share to the players. As our future work, it will be interesting to investigate how to design algorithms with agents' participation behaviors, which maximize social welfare in the long run.

Chapter 4

Auction optimization using regression trees and linear models as integer programs[1]

4.1 Introduction

One of the main challenges of mathematical optimization is to construct a mathematical model describing the properties of a system. When the structure of a system cannot be fully determined from the knowledge at hand, machine learning and data mining techniques have been used in optimization instead of this knowledge. They have, for example, been used in order to obtain decision values (Gabel and Riedmiller 2008), fitness functions (Huyet 2006), or model parameters (Li and Ólafsson 2005). Models that have been learned from data are frequently used in a black-box manner, e.g., using only the predictions of learned models but not their internal structure. It is also possible to use these models in a white-box manner, for instance in order to determine search space cuts and parameter bounds. Neural networks have in this way been used to model unknown relations in constraint programming (Bartolini

[1] This chapter is based on Verwer et al. (2017).

et al. 2011). In this chapter, we develop such a white-box optimization method for regression models in integer linear programming, that is, we map these entire models to sets of variables and constraints and solve them using an off the shelf solver. This white-box method together with a proposed black-box method provides a solution to an optimization problem of key interest to the artificial intelligence and operations research communities: auction design. We briefly introduce this problem domain before going into the details of our methods.

4.1.1 Sequential auction design

Auctions are becoming increasingly popular for allocating resources or items in business-to-business and business-to-customer markets. Often sequential auctions (Bernhardt and Scoones 1994) are adopted in practice, where items are sold consecutively to bidders. Sequential auctions are in particular desirable when the number of items for sale is large (e.g., flower auctions (Heck and Ribbers 1997)), or when the buyers enter and leave the auction dynamically (e.g., online auctions (Pinker et al. 2010)). In a sequential auction, an auctioneer may tune several auction parameters to influence the outcome of an auction, such as reserve prices for items and in which order to sell them. In other words, (s)he can design auctions for the purpose of achieving some predefined goal. In this chapter, we solve one specific auction design problem, namely, deciding the optimal ordering of items to sell in a sequential auction in order to maximize the expected revenue (OOSA in short). We assume bidders in such auctions are budget constrained. This is a highly relevant problem in today's auctions since bidders almost always have limited budget, as seen for instance in industrial procurement (Gallien and Wein 2005b). Previous research has shown that with the presence of budget constraints, the revenue collected by the auctioneer is heavily dependent on the ordering of items to sell (Elmaghraby 2003, Grether and Plott 2009, Subramaniam and Venkatesh 2009). This holds already for a toy problem with 2 items. Let us use a simple example to illustrate the importance of ordering in such cases.

Example 8. *Two agents A_1 and A_2 take part in a sequential auction of items. For sale are items r_1 and r_2. Suppose the items are sold by means of first-price,*

English auction[2]*. Assume the reserve prices, which are the lowest prices at which the auctioneer is willing to sell the times, for both items are 1. The amount that agent* A_1 *and agent* A_2 *are willing to pay for two items are:* $\nu_1(r_1) = 10$, $\nu_1(r_2) = 15$, $\nu_2(r_1) = 12$, $\nu_2(r_2) = 10$. *Furthermore, the budgets of* A_1 *and* A_2 *are* 15 *and* 25 *respectively.*

We assume a simple bidding strategy in this example. The agents bid myopically on each item, that is, their highest bid on one item is the lower value between the amount that they are willing to pay and their remaining budget. The auctioneer's goal is to maximize the total sold price of the items. Consider one situation where the auctioneer sells first r_2 *and then* r_1. A_1 *will get* r_2 *when she just over-bids* A_2 *with 11, and then when* r_1 *is auctioned,* A_1 *bids maximally 4 due to her budget limit, and* A_2 *will win the item with the price of* 5. *The total revenue is* 16. *However, if the selling sequence is* (r_1, r_2), A_2 *will win* r_1 *with the bid* 11, *and then* A_2 *will win* r_2 *with price 11. The collected revenue is* 22 *in this case.* □

Most of the current approaches to the ordering problem in sequential auctions assume a very restricted market environment. They either study the problem of ordering two items, see Subramaniam and Venkatesh (2009), Pitchik (2009), or a market with homogeneous bidders (Elkind and Fatima 2007). To the best of our knowledge, we are the first to consider how to order items for realistic auction settings with many heterogeneous bidders competing for many different items. This problem is highly complex—a good design on ordering needs to take care of many uncertainties in the system. For instance, in order to evaluate the revenue given an ordering, the optimization algorithm needs to know the bidders' budgets and preferences on items, which are usually private and unshared. Furthermore, the large variety of possible bidding strategies that bidders may use in auctions are unknown. This auction design problem is a typical example where the mathematical optimization model cannot be fully determined, and hence, machine learning and data mining techniques can come into play. This is exactly what our approach builds upon.

[2]The English auction that we consider is the one where the starting price is the reserve price, and bidders bid openly against each other. Each subsequent bid should be higher than the previous bid, and the item is sold to the highest bidder at a price equal to her bid.

4.1.2 Learning models for white-box and black-box optimization

Nowadays more and more auctions utilize information technology, which makes it possible to automatically store detailed information about previous auctions along with their selling sequences and the selling price per auctioned item. Our approach to solving the problem of optimal ordering for sequential auctions starts with the historical auction data. We define and compute several relevant features and then use them to learn regression trees and linear regression models for the expected revenue. Given the models, we propose two approaches to find the optimal ordering for a new set of items: (1) a best-first search that uses the models as a black-box to evaluate different orderings of the items; and (2) a novel white-box optimization method that translates the models and the set of items into a mixed-integer program (MIP) and runs this in an ILP-solver (CPLEX). Figure4.1 displays the general framework of our approaches using these two optimization methods.

Just like the traditional black-box optimization approach (see, e.g. Jones et al. (1998), Shan and Wang (2010)), our best-first search is ignorant of the internal structure of the models and only calls it to perform function evaluations, i.e., predicting the revenue of an ordering of the items. Optimization is possible by means of a search procedure that uses heuristics to produce new orderings depending on previously evaluated ones. Our best-first search makes use of dynamic programming cuts inspired by sequential decision making in order to reduce the search space.

One of the main contributions of this chapter is the realization that learned regression models can be evaluated efficiently inside modern mathematical optimization solvers. This evaluation includes the computation of feature values (the input to machine learning), the evaluation of these features using a learned model (the output from machine learning), and a possible feedback from such evaluations to new features. In this chapter, we efficiently translate all of these steps for two types of learned models (regression trees and linear regression models) into mixed-integer constraints. The resulting mixed-integer program can then be evaluated in any modern integer linear programming (ILP) solver.

In this way, modern exact solvers can be used instead of a heuristic search. These solvers use (amongst others) advanced branch-and-bound methods to cut the search space, compute and optimize a dual solution, and can prove optimality without testing every possible solution. This is the main benefit of using the white-box method over a black-box one. The downside, however, is that when the learned model is complex, the white-box method may lead to a large mathematical model that is difficult to optimize. We compare these two approaches and investigate this trade-off by applying them to the OOSA problem.

Contributions and organization Although we use sequential auction design to illustrate our method, all of our constructions are general and can be applied to any optimization setting where unknown relations can be represented using regression models that have been learned from data. The only constraint for using the white-box method is that the feature values need to be computable using integer linear functions from intermediate solutions. Our approach can thus be applied to complex optimization settings where entire orders, schedules, or plans need to be constructed beforehand.

We list our main contributions as follows:

- We demonstrate how to apply regression methods from machine learning to OOSA.
- We give an efficient encoding of regression trees and linear regressors into MIP constraints.
- We prove OOSA with budget constrained bidders to be NP-hard, also when using these regression models.
- We provide the first method that tackles OOSA in realistic settings.
- We demonstrate experimentally that white-box methods outperform black-box methods when the models are not overly complex.

In Section 4.2, we formally introduce the problem of optimal ordering for sequential auctions (OOSA), and then we show how to learn regression models from historical auction data in Section 4.3 using standard machine learning methods. Based on

the learned models, our white-box optimization method and a black-box optimization are introduced to find the optimal ordering for OOSA in Section 4.4. Extensive experiments are presented in Section 4.5 where we compare the performance of the two proposed optimization methods using both the learned models and the auction simulator. Before we conclude, we compare and discuss more related works in Section 4.6.

4.2 Optimal ordering for sequential auctions OOSA

We assume there is a finite set of bidders (or agents). Let $R = \{r_1, \ldots, r_l\}$ denote the collection of the item types, and the quantity of each item type can be more than 1. When it is clear from the context, we will slightly abuse the notation and use $S = \{r_1, r_2 \ldots, r_1, \ldots\}$ to denote the multiset of all available items. Each bidder agent i has a valuation (or preference) for each type of item $v_i : R \to \mathbb{R}^+$. In addition, each agent has a budget b_i on purchasing items, and (s)he desires to win as many items as are being auctioned within the budget limit.

In one auction, a set of n items S with type set $R' \subseteq R$ will be auctioned sequentially using a predetermined order. We use $(s_1, s_2, \ldots, s_n)$ to denote such an ordering. For example, given types r_1 and r_2 with quantities of 1 and 2 respectively, there are three possible orderings of items: $(s_1 = r_1, s_2 = r_2, s_3 = r_2)$, $(s_1 = r_2, s_2 = r_1, s_3 = r_2)$, and $(s_1 = r_2, s_2 = r_2, s_3 = r_1)$. For each r_j that is being auctioned, agent A_i puts a bid on r_j that is the minimum between the amount she is willing to pay for r_j and the remaining budget. We point out that in the case of unconstrained budget, the maximum amount an agent is willing to pay for r_j, defined as $\nu_i(r_j)$, is equal to her valuation $v_i(r_j)$.

Each item r_j comes with a reserve price that is the lowest price at which the auctioneer is willing to sell r_j. If the received bids are all below the reserve price of r_j, r_j is not sold. Otherwise, the agent who bids highest on r_j wins r_j. The winners of items transfer some payment to the auctioneer depending on the auction rule. For example, in a first-price auction, the winner pays an amount equal to her bid, and in a second-price auction, she pays the second highest bid (or the reserve price for the

item if it is higher). The revenue of the auctioneer is the sum of the total payment on the sold items and the total reserve values of the unsold items. This sequential auction ends when all items have been auctioned, or when all agents run out of their respective budgets.

We assume that such an auction is repeated over time, and each auction sells possibly different items S. At the end of each auction, we have the following information at our disposal: (1) the ordering of auctioned items; and (2) the allocation of items to agents with their payments. The optimization problem we study is: *given a set of items and budget constrained bidders, finding an optimal ordering of items in sequential auctions such that the expected revenue is maximized.* We call the problem OOSA.

We now show that the decision version of this optimization problem is NP-hard, even if we have complete information on bidders' preferences and they are not strategic (i.e., they bid truthfully according to their preferences).

Theorem 5. *Given a set of items S, preferences $v_i : S \to \mathbb{R}^+$, and budgets b_i for every bidder i. The problem of deciding whether there exists an ordering that obtains a revenue of at least $K \in \mathbb{R}^+$ is NP-hard.*

Proof. By reduction from the well-known NP-hard partition problem (Garey and Johnson 1979): Given a set of integers $I = \{i_1, \ldots, i_n\}$, is I dividable into two sets A and B such that $\sum A = \sum B$? We need two bidders with preferences such that $v_1(r_k) = 2 \cdot i_k$ and $v_2(r_k) = 2 \cdot i_k + 1$ for $1 \leq k \leq n$. The reserve price for each item k ($1 \leq k \leq n$) is i_k. The agents' budgets are $b_1 = \frac{1}{2} \sum I$ and $b_2 = \sum I$. There are n items in S and K is $\frac{3}{2} \sum I$. We claim that I is partitionable into two sets with equal sums if and only if there exists an ordering that obtains a revenue of K (or more).

($\Rightarrow$) Given a partition of I into sets A and B, we sell all items in A first, and those in B later. In this case, agent 2 will buy all items in A with price $2 \cdot i_k$ as it is the minimal bid to win the items from agent 1. After buying all items in A, agent 2 will have spent $2 \cdot \sum_{i_k \in A} i_k$, which makes $\sum I$ in total (since $\sum_{i_k \in A} i_k = \frac{1}{2} \sum I$). This is the entire budget of agent 2. All items in B are then sold to agent 1 with the reserve price i_k. Thus agent 1 pays $\sum_{i_k \in B} i_k = \frac{1}{2} \sum I$. This makes a total revenue of $\sum I + \frac{1}{2} \sum I = \frac{3}{2} \sum I = K$.

($\Leftarrow$) Suppose we have an ordering such that agent 1 and 2 spend all of their budget (K in total). This means that agent 2 wins the first set of items A, each costing $2 \cdot i_k$ till it uses all its budget. Thus we have $2 \cdot \sum_{k \in A} i_k = \sum I$, i.e., $\sum_{k \in A} i_k = \frac{1}{2} \sum I$. Agent 1 pays i_k for the remaining items in B, $B = I \setminus A$, and it uses all its money: $\sum_{k \in B} i_k = \frac{1}{2} \sum I$. Hence, we have a partition of I where $\sum A = \sum B$.

The construction is clearly polynomial time. □

Several related works deal with this type of ordering optimization problem. For example, the authors of Subramaniam and Venkatesh (2009) investigate the optimal ordering strategy for the case where the auctioneer has two items to sell. They show that when the items are different in value, the higher valued items should be auctioned first in order to increase the seller's revenue. Pitchik (2009) points out that in the presence of budget constraints, in a sealed-bid sequential auction, if the bidder who wins the first good has a higher income than the other one, the expected revenue is maximized. These greatly simplified auction settings make it possible to derive bidders' equilibrium bidding strategies. With some assumptions on the distributions of bidders' budgets and preferences, the optimal ordering can be theoretically derived. However, as real-world auctions are much more complex and uncertain in terms of the sizes of items/bidders, agents' preferences and bidding strategies, these existing results cannot be applied. In this chapter, we instead focus on learning the overall behaviors of the group of bidders from historical auction data by machine learning techniques, as the first step of solving OOSA.

In order to apply ML techniques, we assume in every sequential auction the set of participating bidders and their characteristics (preferences, budgets, bidding strategies) to be similar. This simplifies the problem of learning a good ordering. Instead of learning the individual valuations/budget/bidding strategies of agents, we can treat the agent population as a single entity for which we need to find a single global function. Obviously, such an approach will fail if the agents are radically different in every auction. However we consider this assumption sensible in many auctions such as industrial procurement auctions where the same companies repeatedly join the auctions with similar interests, and the Dutch flower auction where there can be different bidders every day, but it seldom occurs that one day bidders are only interested in roses and the next day they only want tulips. Although the different

participants can be interested in different item types, the interests of the group of participants remain stable.

4.3 Learning predictive models for OOSA

At the end of each sequential auction, we have the following information at our disposal: (1) the ordering of auctioned items; and (2) the price of each sold item. Before we build the optimization model to solve the OOSA problem, we need to find a suitable way to model the expected revenue of given orderings of auctioned items.

An ordering can be thought of as a sequence of items. However, to the best of our knowledge none of the existing sequence models fit our auction setting, see also Section 4.6.2. In this work, we view the prediction of an auction's outcome as a regression problem. We split this problem into the subproblems of predicting the value of the auctioned items. We then sum these up to obtain the overall objective function, i.e., the expected revenue $P(S)$ given a set S ($|S| = n$) of items:

$$P(S) = \sum_{1 \leq k \leq n} G(s_k, \{s_j \mid j < k\}, \{s_l \mid k < l\}),$$

where $G(s_k, J, L)$ is a regression function that determines the expected value of s_k given that J was auctioned before and L will be auctioned afterwards. The main benefit of this representation is that modern machine learning methods can be used to learn this function G from data. In addition, since every item sold represents a single sample, every auction contains many samples that can be used for learning, further reducing the amount of required data. We study two popular regressions functions.

4.3.1 Two regression functions

In this chapter, we use regression trees (Breiman et al. 1984) and least absolute shrinkage and selection operator (LASSO) (Tibshirani 1994) as regression functions, and train them using features based on the items auctioned before and after the current item. We first briefly introduce these regressors.

Regression trees Regression trees are a form of decision trees where the predicted values are real numbers. A decision tree is one of the most popular predictive models for mapping feature values to a target value. We make use of a regular univariate decision tree. It is a tree-shaped graph with a root node, interior nodes, and leaf nodes. The root and every interior node contains a Boolean test for a specific feature value f, such as $f > 5$. Every leaf node contains an output value p. It maps the feature values to an output by performing all the tests along a path from the root to a leaf. For every test performed, if the outcome is true ($f > 5$), the path is continued along the left branch, if the outcome is false ($f \leq 5$), the path is continued along the right branch. Once a leaf is reached, it outputs the value it contains p.

A regression tree learner aims to find a tree that minimizes the mean squared error of the predicted and the actual observed values. Most regression tree learning algorithms follow a greedy strategy that splits interior nodes as long as the decrease in error is significant. A split replaces one leaf node by an interior node connected with two new leaf nodes. The interior node receives as Boolean constraint one that minimizes the mean-squared error of the resulting tree, where the leaf nodes predict the mean value of all observed data values that end up in that leaf after mapping all data samples to leaf nodes.

LASSO LASSO is a method for constructing a linear regression function

$$p(f_1, \ldots, f_m) = c_1 f_1 + c_2 f_2 + \ldots + c_m f_m + d$$

where p is the value to predict, c_i are constants, f_i are feature values, and d is the intercept. The standard approach to find such a function is to minimize the mean squared error, which is easy to compute. LASSO is a popular regularized version of this simple estimation that penalizes the absolute values of the regression coefficients $c_1, \ldots, c_m$. Formally, given a dataset of features f_i^d and target values p^d, with $1 \leq d \leq k$ where k is the number of samples, it uses convex optimization in

order to find a regression function that solves the following problem[3]:

$$\min \sum_{1 \leq d \leq k} \frac{1}{2 \cdot k} (p(f_1^d, \ldots, f_m^d) - p^d)^2 + \alpha \cdot \sum_{1 \leq i \leq m} |c_i|$$

where $0 \leq \alpha \leq 1$ is a parameter for the effect of the regularization. Intuitively, the larger α, the larger the penalty of having large coefficients c_i. Consequently, a larger value of α will drive more coefficients to zero. LASSO is a useful method when there are several correlated feature values, which could make an ordinary least squares model overfit on these values. We use LASSO regression because more zero coefficients implies we need to compute less feature values in order to evaluate the learned model, which has a positive effect on the optimization performance that we will discuss in Section 4.4.

4.3.2 Learning regression functions for predicting revenues

We first give an overview in Figure 4.1 of the connection between the regression models and the optimization models for solving OOSA. Given historical auction data, a regression tree (or a LASSO linear regression function) is learned for each item type. The regression tree (or LASSO) can be used to evaluate the values of selling different items based on the feature values that are computed on a given ordering of the items. The learned regression trees (or LASSO functions) are then used in two ways to model the optimization problem OOSA: (1) Black-box optimization. In this chapter, we use a best-first search heuristic to come up orderings of the items, and then use the learned regression trees (or LASSO) to compute the expected revenue of these orderings; (2) White-box optimization. We formulate the optimization problem of finding an optimal ordering as a mixed integer linear program (MIP), which is shown to be automatically constructed from the learned regression tress (or LASSO functions).

We now present the details of learning regression trees and LASSO functions for item types. Currently, we provide the following features for these two regression models:

[3]This is the version implemented in the scikit-learn Python package (Pedregosa et al. 2011), which we use to learn the models.

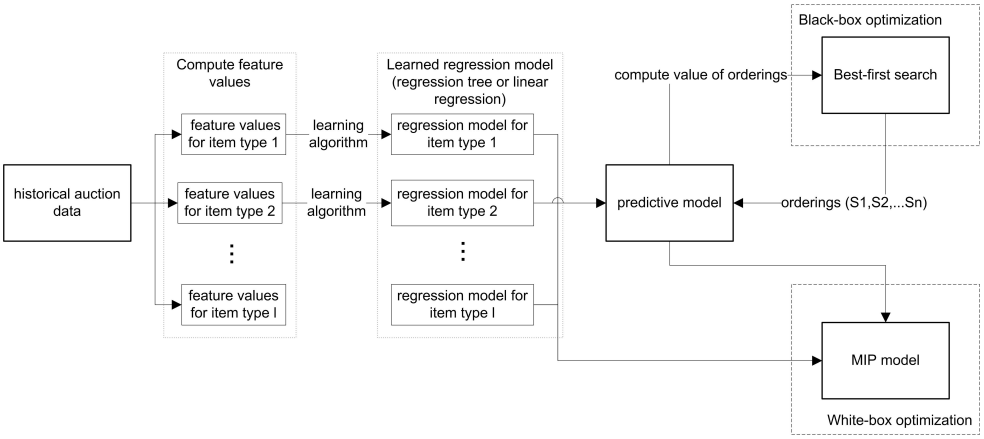


Figure 4.1: Solving OOSA using white-box optimization and black-box optimization with learned models. Black-box optimization only calls the predictive model to evaluate possible orderings. White-box optimization translates the internal structure of the predictive model to MIP constraints.

Feature 1: `sold` For every item type r, the amount of r items already auctioned.

Feature 2: `remain` For every item type r, the amount of r items still to be auctioned.

Feature 3: `diff` For every pair of item types r and r', the difference between the amount of r and r' items already auctioned.

Feature 4: `sum` For every item type r, the amount of value obtained from auctioning r items, and the overall sum.

Feature 5: `index` For every item, the index at which it was auctioned.

Other sequential features such as sliding windows and N-grams (see, e.g., Dietterich (2002)) can of course be added to the model. However, since our white-box method computes these values inside an ILP solver, the only requirement is that they can be represented using an integer linear formulation. Although the `diff` feature can be determined using the first, we add it for convenience of learning a regression

Table 4.1: The data set created from the past two auctions $\{r_2, r_1\}$ and $\{r_1, r_2\}$ in Example 9.

type	value	soldr_1	soldr_2	remainr_1	remainr_2	diffr_1r_2	sumr_1	sumr_2	sum	index
r_2	11	0	0	1	0	0	0	0	0	1
r_1	5	0	1	0	0	-1	0	11	11	2
r_1	11	0	0	0	1	0	0	0	0	1
r_2	11	1	0	0	0	1	11	0	11	2

tree, which requires many nodes to represent such values. The influence of budget constraints is only directly modeled by the fourth feature: once the amount paid for r_1 items reaches a certain (to be learned) bound, we can expect all agents that only want r_1 items to be out of budget. Although, there only exists an indirect relation between the budget constraints and the first three features, including them can be beneficial and these are easier to compute. If used by the regression model, these features thus reduce the time needed to solve the auction design problem. For similar reasons, we add the last feature. Below we give an example of how an ordering and its obtained values is transformed into a data set using these 5 types of features.

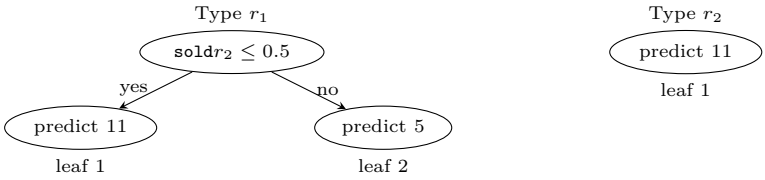


Figure 4.2: Two learned regression trees for the two item types r_1 and r_2 from Example 9. The leafs of the left (right) tree output the predicted value of the item of type r_1 (r_2), determined by the feature value soldr_2.

Example 9. *Consider the setting of Example 8. Assume two auctions have been carried out. One sold r_2 first and then r_1. The other reversed. As shown in Example 8, the first auction would obtain a revenue of 16, and the second auction would receive 22. We compute feature values from these two auctions as depicted in Table 4.1.*

Subsequently, we learn regression trees for both item types r_1 and r_2, as shown in Figure 4.2.[4]

After learning these regression trees, we can optimize the ordering for a new (unseen) multiset of items $\{r_1, r_1, r_1, r_2\}$ by trying all orderings and choosing one with maximum expected revenue: (r_1, r_1, r_1, r_2) gives $11+11+11+11 = 44$, (r_1, r_1, r_2, r_1) gives $11+11+11+5 = 38$, (r_1, r_2, r_1, r_1) gives $11+11+5+5 = 32$, and (r_2, r_1, r_1, r_1) returns $11+5+5+5 = 26$. Hence, the optimizer will choose to schedule the r_2 item after all r_1 items. □

The example showed how to evaluate different orderings of items using the learned regression trees. In general, trying all possible orderings will be impossible: for a multiset of items $S = \{r_1, \ldots, r_m\}$ of m types, there are a total of $\frac{|S|!}{\prod_{1 \leq i \leq m} |\{r_i | r_i \in S\}|!}$ unique orderings, which blows up very quickly.

4.3.3 Modeling power and trade-off

Our method of regression modeling allows the use of any regression method from machine learning for predicting unknown quantities in optimization, such as objective values and parameters. In addition, since the regression function G uses other values in a (proposed) solution as input (J, L) instead of only external parameters/data, a learned regression model represents unknown relations between the different values in a solution. The model thus answers the question "What is the value of X given that we do Y?", as opposed to "What is the value of X?" that is answered by fitting only model parameters. Answering the first question allows for many more interesting possibilities. For instance, one could use stochastic optimization with fitted parameters to produce a schedule, use regression models to predict the effect of this schedule on the parameters, and use stochastic optimization again on the newly estimated parameters. This way, one can use machine learning tools to plan further ahead. Using our white-box method, this can even be done using a single call to the optimization software (see Section 4.4).

This loop-back functionality provides a lot of power to our method, but also comes with a risk. Every time the predictive models are used there is a probability that

[4]The learned linear regression model is more straightforward. Hence we skip such an example here.

the predictions are inaccurate. When using a loop-back, these possible inaccuracies influence all future predictions that depend on it. These future predictions are thus more inaccurate and the predicted overall objective value can potentially diverge from the true value. These cascading inaccuracies are an issue, however, the added modeling power makes up for it. We make use of it in the `sum` feature, which relates the predicted value to the predictions of earlier auctioned items. This feature is very important for predicting budget constraints, and consequently is often used by the regression models to produce predictions.

4.4 White-box and black-box optimization for OOSA

Given the predictive models for the expected value per item, it is not straightforward to compute a good ordering as we already showed in Example 9. For a given ordering, we can predict the individual revenues of items using the regression model, and sum these up to obtain the revenue of the ordering. However, testing all possible orderings and choosing the one with the highest revenue will take a very long time. For instance, when we want to order 40 items of 8 types (the experimental setting in Section 4.5) with 5 of every type, we will need to test $\frac{40!}{5!^8} \approx 1.9 \cdot 10^{31}$ possible unique orderings.

In 4.A, we also provide hardness results that demonstrate there is little hope (unless $\mathsf{P} = \mathsf{NP}$) of finding an efficient (polynomial-time) algorithm that gives the optimal ordering for any regression tree or linear regression predictor. In general, we cannot do better than performing a guided search through the space formed by all possible orderings. We present two such search-based optimization methods: (1) a novel "white-box" optimization (i.e., ILP model), and (2) a "black-box" heuristic (i.e., best-first search).

4.4.1 White-box optimization: an ILP model

Given regression tree and linear regression models for the expected value per item type, we automatically formulate the problem of finding an optimal ordering as a mixed integer linear program (MIP). We discuss the encoding of a sequential auction,

feature values, objective function, and translating the learned models (regression tree and linear regression respectively) below.

Ordering an auction Given a multiset S of n items, each from a set of possible types R, we use the following free variables to encode any possible ordering of S: $x_{i,r} \in \{0, 1\}$. Item i is of type r if and only if $x_{i,r} = 1$. Thus, if x_{3,r_1} is equal to 1, it means that the third auctioned item is of type r_1. We require that at every index i at most one item type is auctioned, and that the total number of auctioned items of type r is equal to the number n_r of type r items in S.

$$\sum_{r \in R} x_{i,r} = 1 \qquad \text{for all } 1 \leq i \leq n \tag{4.1}$$

$$\sum_{1 \leq i \leq n} x_{i,r} = n_r \qquad \text{for all } r \in R \tag{4.2}$$

Any assignment of ones and zeros to the x variables that satisfies these two types of constraints corresponds to a valid ordering of the items in S. The value of such an ordering is determined by the learned regression models.

Translating feature values In order to compute the prediction of a regression model, we not only need to translate the models into ILP constraints, but also the values of the features used by these models.[5] Feature 1, 2, 3, and 5 can be computed using linear functions from the x variables:

$$\texttt{sold}_{i,r} = \sum_{j<i} x_{j,r} \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.3}$$

$$\texttt{diff}_{i,r,r'} = \texttt{sold}_{i,r} - \texttt{sold}_{i,r'} \qquad \text{for all } 1 \leq i \leq n, r, r' \in R, r \neq r' \tag{4.4}$$

$$\texttt{remain}_{i,r} = \sum_{j>i} x_{j,r} \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.5}$$

$$\texttt{index}_i = i \qquad \text{for all } 1 \leq i \leq n \tag{4.6}$$

[5] Including any transformations applied to them.

For the fourth type of feature, we use an additional variable $p_{j,r}$, which encodes the expected value of the item auctioned at index j of type r. If the item at index j is not of type r, $p_{j,r}$ is equal to 0. Since the p variables are the predictions of the regression functions, we provide their definition after defining the regression models.

$$\texttt{sum}_{i,r} = \sum_{1 \leq j \leq i} p_{j,r} \text{ for all } 1 \leq i \leq n, r \in R \tag{4.7}$$

To aid the ILP solver, we also pre-compute the minimum $m_{f,i}$ and maximum $M_{f,i}$ obtainable values of every feature f at every index i and provide these as bounds to the solver.

Constructing the objective function We aim to maximize the expected values $p_{i,r}$:

$$\max \sum_{1 \leq i \leq n} \sum_{r \in R} p_{i,r} \tag{4.8}$$

Although it is also possible to compute both the objective function and the `sum` values as very large sums over the x and model variables (described next), specifying parts of these sums as intermediate continuous p variables significantly reduces both the encoding size and the computation time.

Finally, we discuss how to encode the learned regression tree and the linear regression model as the constraints in the ILP model.

Encoding regression trees We translate the regression tree models into ILP using carefully constructed linear functions. Our encoding only requires one new set of $\{0, 1\}$ variables $z_{i,l,r}$, representing whether a leaf node l is reached for item type r at index i. The internal (decision) nodes of the trees can be represented implicitly by the constraints on these new z variables. Intuitively, we encode that a z variable has to be false when the binary test of any of its parent nodes fails. By additionally requiring that exactly one z variable is true at every index, we fully encode the learned regression trees.

Let D_r be the set of all decision nodes in the regression tree for type r. Every decision node in D_r contains a boolean constraint $f \leq c$, which is true if and only if feature f has a value less than or equal to a constant c. A key insight of our encoding is that every such boolean constraint directly influences the value of several z variables: if it is true (at index i), then all z variables representing leafs in the right subtree are false; if it is false, then all that represent leafs in the left subtree are false. In this way, we require only two constraints per boolean constraint in order to represent all possible paths to leaf nodes.

$$\texttt{fv}_{f,i} + (M_{f,i} - c) \cdot \sum_{l \in L} z_{i,l,r} \leq M_{f,i} \text{ for all } 1 \leq i \leq n, r \in R, (f \leq c) \in D_r \tag{4.9}$$

$$\texttt{fv}_{f,i} + (m_{f,i} - c) \cdot \sum_{l \in L'} z_{i,l,r} \geq m_{f,i} \text{ for all } 1 \leq i \leq n, r \in R, (f \leq c) \in D_r \tag{4.10}$$

where $\texttt{fv}_{f,i}$ is a calculation of feature value f for index i, L and L' are the leaf nodes in the left and right subtrees of the decision node with constraint $(f \leq c)$ in the regression tree for type r, and $M_{f,i}$ and $m_{f,i}$ are the maximum and minimum values of feature f at index i. For the feature calculation we simply replace $\texttt{fv}_{f,i}$ with the right-hand sides of the corresponding feature definitions.[6]

The above constraints ensure that when $z_{i,l,r}$ obtains a value of 1, all of the binary test in the parent nodes on the path to l in the tree for type r return true at index i. By construction of the regression trees, this ensures that at most one z variable is true for every type r and index i. We require however that exactly one z variable is true at every index.[7] This z has to be of the same type as the item sold at index i, denoted by the x variables:

$$\sum_{l} z_{i,l,r} = x_{i,r} \qquad \text{for all} 1 \leq i \leq n, r \in R \tag{4.11}$$

[6] We ignore the possibility that a feature f is equal to c because, since features have a limited precision, we can always replace the constants in a decision node with one that cannot be obtained by f, without changing its behavior.

[7] Counterintuitively, it can occur that the objective function (discussed below) is maximized when every z variable is false at an index i. If a small `sum` is needed to reach a high revenue prediction, it can be beneficial to auction but not sell an item.

This completes our encoding of the regression trees. The predictions of the trees at every index i are given by the z variable that is true for index i. We multiply this z variable with the constant prediction in the leaf node it represents to obtain the prediction, and store it in the p variables used to compute the `sum` feature values.

$$p_{i,r} = \sum_{l \in L_r} c_{l,r} \cdot z_{i,l,r} \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.12}$$

where $c_{l,r}$ is the constant prediction of leaf l in the tree for type r.

Complexity Our translation of regression trees is very efficient. It requires only 2 constraints per decision node (D_r, Equations 4.9 and 4.10) and 1 binary variable $z_{i,l,r}$ for every leaf node ($L + L'$, used in Equations 4.9, 4.10, and 4.11) of the tree. To encode a complete depth k tree with $2^{k+1} - 1$ nodes, this thus requires only $2 \times (2^k - 1) = 2^{k+1} - 2$ constraints for the decision nodes and 2^k binary variables for the leaf nodes. In addition, we require 1 constraint to force exactly one leaf variable to be true (the same type as $x_{i,r}$, Equation 4.11). All the other variables and constraints, used to compute the feature values and $p_{i,r}$ variables, can be computed directly without storing the result into a variable. Consequently, this adds zero variables and zero constraints to the translation.

In order to encode an entire OOSA problem, a new set of constraints representing a tree is constructed for every item type and every item index. In a problem with $|R|$ types and n items for sale, this creates $n \times |R| \times (2^{k+1} - 1)$ constraints and $n \times |R| \times (2^k)$ variables to encode a complete tree of depth k. The ordering problem itself requires $n \times |R|$ ($x_{i,r}$, Equations 4.1 and 4.2) variables and $n + |R|$ constraints. This totals to: $n \times |R| \times (2^k + 1)$ variables and $n \times |R| \times (2^{k+1} - 1) + n + |R|$ constraints. Since a complete depth k tree has $2^{k+1} - 1$ nodes, this is linear in the number of nodes of the tree.

We now discuss how to encode a linear regression model.

Encoding linear regression model Due to its linear nature, implementing linear regression in ILP is very straightforward. We can directly compute the value of the p variables using the linear predictor function:

$$p_{i,r} = \sum_{f \in \texttt{Feat}} c_{f,r} \cdot \texttt{fv}_{f,i} \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.13}$$

where $\texttt{Feat}$ is the set of all features, $\texttt{fv}_{f,i}$ is feature f's values at index i, and $c_{f,r}$ is the constant coefficient for feature f in the regression function for type r. The only somewhat difficult part is that at every index, the used regression function can change depending on the auctioned item type r. We implemented this choice using indicator functions in CPLEX.[8] This changes the above formulation as follows:

$$x_{i,r} = 1 \rightarrow p_{i,r} = \sum_{f \in \texttt{Feat}} c_{f,r} \cdot \texttt{feat}_{f,i} \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.14}$$

$$x_{i,r} = 0 \rightarrow p_{i,r} = 0 \qquad \text{for all } 1 \leq i \leq n, r \in R \tag{4.15}$$

It states that if $x_{i,r}$ is true, then the values of $p_{i,r}$ is determined using the regression function. Otherwise, its value is 0. These are the only constraints needed to fully implement a linear regression function. When using LASSO regularization, some coefficients will receive the value 0. These are removed from the encoding, making the models smaller and easier to evaluate in the solver.

Complexity The translation of regression functions is straightforward and only requires 2 constraints per item type for the indicator functions (Equations 4.14 and 4.15). Because of these indicators, we do need to encode the $n \times |R|$ $p_{i,r}$ variables. No additional constraints or variables are required. In order to encode an entire OOSA problem, we thus require only $n \times |R| \times 2$ constraints and $n \times |R| \times 2$ variables.

An example Now the two ILP models are complete and ready to solve the OOSA problem. We give the following example to illustrate how the formulation of ILP works given learned regression trees.

[8] Many other solvers have similar constructions. If not, these constraints can be implemented using a 'big-M' formulation, similar to the one we use to determine the value of the z variables in the regression tree formulation.

Example 10. *Given the learned trees in Example 9, suppose we are asked to order a new multiset of items* $\{r_1, r_2, r_2\}$*. We translate this new set, together with the learned trees into the following integer linear program with the following* $\{0,1\}$ *decision variables (for all* $1 \leq i \leq 3$*):* $x_{i,r_1}, x_{i,r_2}, z_{i,1,r_1}, z_{i,2,r_1}, z_{i,1,r_2}$*:*

$$\begin{aligned} &\max \textstyle\sum_{1\leq i\leq 3} p_{i,r_1} + p_{i,r_2} \\ \textit{where } p_{i,r_1} &= 11 z_{i,1,r_1} + 5 z_{i,2,r_1} \\ \textit{and } p_{i,r_2} &= 11 z_{i,1,r_2} \end{aligned}$$

subject to (for all $1 \leq i \leq 3$*)*

$$\begin{aligned} x_{1,r_1} + x_{2,r_1} + x_{3,r_1} &= 1 \\ x_{1,r_2} + x_{2,r_2} + x_{3,r_2} &= 2 \\ x_{i,r_1} + x_{i,r_2} &= 1 \end{aligned}$$

This denotes that exactly one x *variable is true at every index* i*,* 2 x *variables are true for item type* r_2*, and* 1 *for type* r_1*. This encodes all possible orderings. From this we compute the feature values (for all* $1 \leq i \leq 3$*):*

$$\begin{aligned} \texttt{sold}_{i,r_1} &= x_{1,r_1} + \ldots + x_{i-1,r_1} \\ \texttt{sold}_{i,r_2} &= x_{1,r_2} + \ldots + x_{i-1,r_2} \end{aligned}$$

that are used in the constraints denoting the Boolean tests in the internal nodes:

$$\begin{aligned} \texttt{sold}_{i,r_2} &+ (100 - 0.5) z_{i,1,r_1} \leq 100 \\ \texttt{sold}_{i,r_2} &+ (-0.5) z_{i,2,r_1} \geq 0 \end{aligned}$$

where $M_{\texttt{sold},i} = 100$ *and* $m_{\texttt{sold},i} = 0$*. The first two constraints encode that if* $z_{i,1,r_1} = 1$*,* $\texttt{sold}_{i,r_2} \leq 0.5$*; and if* $z_{i,2,r_1} = 1$*,* $\texttt{sold}_{i,r_2} \geq 0.5$*. Thus, if a* z *variable is true for a leaf, then all the Boolean tests of internal nodes on the path from the root to that leaf have to succeed. At last, we require that exactly one* z *variable is true at every index:*

$$\begin{aligned} z_{i,1,r_1} + z_{i,2,r_1} &= x_{i,r_1}, \\ z_{i,1,r_2} &= x_{i,r_2}. \end{aligned}$$

A satisfying assignment to the x *variables is* $x_{1,r_1}, x_{2,r_2}, x_{3,r_2}$ *set to 1, the rest to 0, corresponding to the ordering* (r_1, r_2, r_2)*. Since* $\texttt{sold}_{1,r_2} = 0$*, this leads to* $99.5 z_{1,1,r_1} \leq 100$ *and* $-0.5 z_{1,2,r_1} \geq 0$*, forcing* $z_{1,2,r_1} = 0$*. Since* $z_{1,1,r_1} + z_{1,2,r_1} =$

$x_{1,r_1} = 1$, this implies $z_{1,1,r_1} = 1$. For the next index, since $x_{2,r_2} = 1$, it forces $z_{2,1,r_2} = 1$. Similarly, we obtain $z_{3,1,r_2} = x_{3,r_2} = 1$. This results in $p_{1,r_1} = 11$, $p_{2,r_2} = 11$, $p_{3,r_2} = 11$, and an objective value of 33.

□

4.4.2 A black-box heuristic: best-first search algorithm

We also provide a black-box heuristic for solving the ordering problem, see also Verwer and Zhang (2012). The traditional method to overcome the computational blowup caused by sequential decision making is to use a dynamic programming method. Although this lessens the computational load by combining the different paths that lead to the same sets of auctioned items, the search space is still too large and waiting for a solution will take too long. Instead, we therefore employ a best-first search strategy that can be terminated anytime in order to return the best found solution so far. We show how this best-first search strategy works in Algorithm 2.

The algorithm uses a hashtable and a priority queue. The hashtable is used to exclude the possibility of visiting the same nodes twice if the obtained value is less than before (just like a dynamic programming method). These dynamic programming cuts are sensible but lose optimality as on rare occasions it could be better to sell earlier items for less, leaving more budget for the remaining ones. The priority queue provides promising candidate nodes for the best-first strategy. By computing random orderings of the remaining items, the learned models can evaluate complete orderings of all items. The best one found is stored and returned if the algorithm is terminated. Unfortunately, this does not result in an admissible heuristic for an A* search procedure. Hence, even if the algorithm pops a solution from the queue, this is not necessarily optimal. In our experience, using random orderings of the remaining items in this heuristic provides a good spread over the search space. Although some nodes can be 'unlucky' and obtain a bad ordering of the remaining items, there are always multiple ways to reach nodes in the search space and it is very unlikely that all possibilities will be 'unlucky'.

Algorithm 2 Black-box heuristic for solving OOSA: best-first search

Require: A set of items S, historical data on orderings and their values D, a maximum number of iterations m
Ensure: Returned is a good (high expected value) ordering
  Transform D into a data set
  for every item type r **do**
    Learn a regression model from D for predicting the value of item type r
  end for
  Initialize a hashtable H and a priority queue Q
  Add the empty data row to Q
  while Q is not empty and the size of H is less than m **do**
    Pop the row of features F with highest value p from Q
    if H does not contain F with a value $\geq p$ **then**
      Add F with value p to H
      Let L be the set of remaining items in F
      for every item type r of items in L **do**
        Let i_k be an item of Type r in L
        Let L' be a random ordering of $L - i_k$
        Use the models to evaluate the value p' of auctioning the ordering $i_k L'$ after F
        Create new features F' for auctioning i_k after F
        Add F' to Q with value $p + p'$
      end for
    end if
  end while
  return The highest evaluated ordering

4.4.3 Discussion: white-box or black-box optimization?

The main difference between the two abovementioned approaches (see Figure 4.1) is that the white-box method specifies the predictors entirely as constraints, which can be used to infer bounds on the predictions and cut the search space. The black-box method instead uses the predictors as oracles and is ignorant as to how the predictions are made, which are naturally more efficient to compute but cannot be used to infer search space cuts, i.e., to deduce that one ordering is better than another without testing both of them. Another key difference is that the white-box method results in a single optimization model that can be run in any modern solver, while the black-box method requires the use of executable code to produce the predictions. In the black-box setting, it is therefore much harder to use the powerful solving methods available in dedicated solvers for problems such as integer linear

optimization (ILP), satisfiability (SAT), or constraint programming (CP). Instead, general search methods can be used such as best-first search, beam-search, meta-heuristics, genetic algorithms, etc.

Both black-box and white-box approaches have their advantages. The main advantage of black-box is that its performance is for a large part independent of the complexity of the used regression model. In contrast, by explicitly modeling the regression model as constraints in a white-box, more complex regression models lead to many more constraints, which can dramatically increase in the time needed to solve it. Another advantage of black-box optimization is that it is easy to include additional cuts such as the dynamic programming cuts discussed above. Such cuts can be added as constraints in an LP formulation, but this can lead to a blow-up in runtime.

The main benefit of using the white-box approach is the use of modern exact solvers instead of a heuristic search. These solvers use (amongst others) advanced branch-and-bound methods to cut the search space, compute and optimize a dual solution, and can prove optimality without testing every possible solution. Our white-box constructions can also be easily integrated into existing (I)LP formulations that have been used in a wide range of applications in for instance Operations Research. In this way, one can combine the vast amount of expert knowledge available in these applications with the knowledge in the readily available data.

The most important downside of white-box is that an evaluation of translated models likely requires more time than running the code as a black-box, especially when the models or features are somewhat complex. In our opinion, however, the advantages of white-box optimization largely outweigh those of black-box optimization and make it a very interesting topic for research in machine learning and optimization.

4.5 Experiments

Designing an optimal ordering for sequential auctions is difficult with heterogeneous bidders, as they may value items differently, have different budget constraints, and moreover, bid rationally or irrationally with various bidding strategies. To evaluate

the performance of the proposed optimization methods, ideally, we should collect real auction data, build the optimization models, run real-world auctions with real bidders using different ordering of items produced by different methods, and then compare the resulting revenues. Since this evaluation method is not feasible for us, nor is it the main purpose of this chapter, we opted for a widely accepted evaluation approach in research community, that is, we created an auction simulator which simulates auctions with agents. We used this simulator to generate auction data sets, and to evaluate the proposed method. An overview of this process is given in Figure 4.3.

4.5.1 The simulator

Simulating auctions We simulate several sequential auction settings with the simulator: (i) first price auctions where agents bid the lower value between the amount they are willing to pay for the item, which is no higher than their valuation on the item, and their remaining budget, as in Subramaniam and Venkatesh (2009); (ii) second price sealed bid auctions, i.e. Vickrey auctions (Vickrey 1961). Agents bid truthfully on each item in each round based on their valuations, or in case of insufficient budget, they bid their remaining budget, as in Pinker et al. (2010). This is the best-response bidding strategy for myopic utility-maximizing agents who only consider the current round of the auction.[9] (iii) Vickrey auctions where agents bid smartly, i.e., they compare the utility obtained at the end of the auction when buying and not buying the item and place a bid based on the difference (see Section 4.5.4 for more details). On the last auctioned item, they bid truthfully if the budget allows.[10] Otherwise, they bid their remaining budget. Given all bids on an item, the highest bid wins. If multiple agents have the same highest bid, one of these is selected as winner uniformly at random. With these different auction settings, we intend to show our method is robust to the auction rules and bidding strategies. Below we explain how we generated agents and items for these settings and what parameters

[9] Note that in sequential Vickrey auctions with budget constrained agents, truth-telling is not an equilibrium bidding strategy (see Vetsikas (2013)).

[10] Vickrey (1961) showed that in a sequential auction with unlimited budget, it is a weakly dominant strategy for bidders to bid their true values for the last auctioned item.

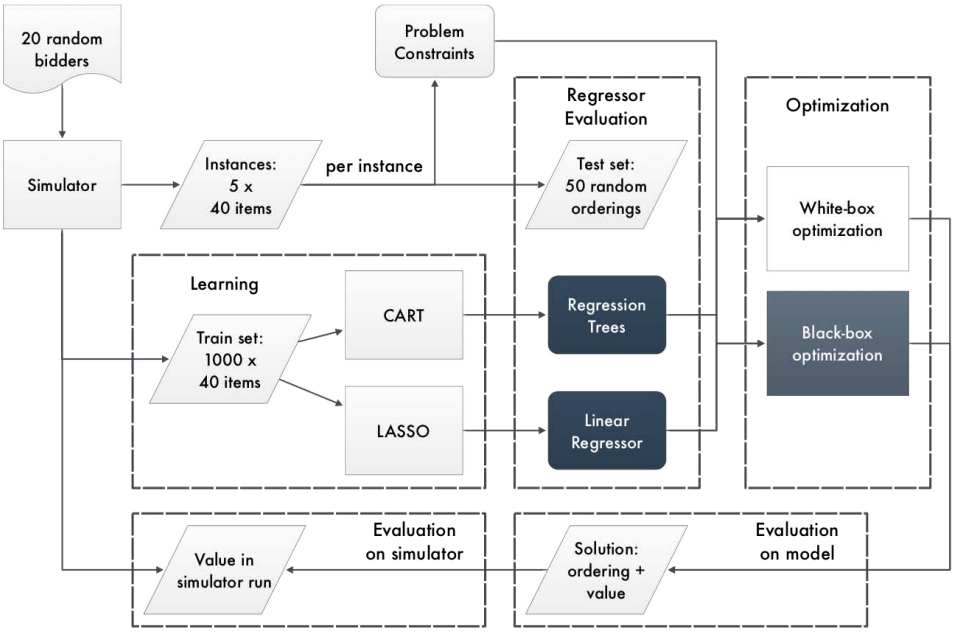


Figure 4.3: Our framework for evaluating optimization using learned models for OOSA. There is a simulator that is used in two ways: to generate historic data and to evaluate the OOSA solutions. A train set and a few unseen problem instances are generated. The train set is used to learn the regression models. Random orderings of the problem instances are used to test them. The instances together with the learned models are provided as input to the ILP-based white-box optimization and the best-first black-box optimization. The resulting orderings are evaluated using the learned models, and using simulator runs.

we used. At the end of this chapter, we will show how well our method scales when we use different parameters for the generator.

Item types We use a given set of 8 items to initialize the auction simulator. Every type r_i gets assigned a base value μ_i of $25 + (5 \cdot i)$, for $1 \leq i \leq 8$, and a reserve price $\rho_i = \frac{1}{2}(25 + (5 \cdot i))$. Every type is assigned popularity and sparsity values,

denoted by γ_i and λ_i, drawn uniformly from $[2, 10]$. The popularity value measures the degree of desirability of the item type by the agents. The sparsity is a measure for the frequency that an item type is available in one auction. In every auction, 40 items are generated using a roulette wheel drawing scheme using the sparsity values.

Bidder agents The simulator starts with 20 randomly generated bidding agents. Every such agent A_j gets assigned a budget b_j between 25 and 150 uniformly at random. They may desire 1 to 5 of the 8 item types, where popular types have a higher probability of being selected, drawn using a roulette wheel selection on the item types' popularity values. Every desired item type r_i assigned to an agent A_j is also given a value of $\nu_j(r_i) = \beta \cdot \mu_i$, where μ_i is the base value of type r_i, and β is a uniform random value between 0.5 and 2.0. The value $\nu_j(r_i)$ is used as the amount that agent A_j is willing to pay for r_i in the first-price auctions, and is used as the valuation of A_j on r_i in the second-price auctions. If this value is greater than the budget of the agent, that agent's budget is increased by another value between 25 and 150 (sampled uniformly), and adding this to its budget. This is repeated until the budget is sufficient for every item type.

Example 11. *The following is an example of eight agents $A_1, \ldots, A_8$ and four item types r_1, r_2, r_3, r_4 generated for the small-scale experiments in Section 4.5.5, where for the four item types, the reserve prices of the auctioneer are: $\rho_1 = 12.5$, $\rho_2 = 15$, $\rho_3 = 17.5$, and $\rho_4 = 20$; the sparsity values of the four item types are: $\lambda_1 = 2$, $\lambda_2 = 7$, $\lambda_3 = 2$, and $\lambda_4 = 5$; and the popularity values are $\gamma_1 = 8$, $\gamma_2 = 8$, $\gamma_3 = 6$, and $\gamma_4 = 2$. Every agent's budget is sampled between 25 and 80.*

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
budget b	*78*	*37*	*80*	*60*	*119*	*103*	*46*	*63*
$\nu(r_1)$	*34*	*24*	*20*	*41*		*38*	*24*	
$\nu(r_2)$	*59*	*30*		*21*		*58*	*25*	*42*
$\nu(r_3)$			*61*	*30*	*53*			
$\nu(r_4)$	*74*		*22*					

In the generated agents, we can clearly see that some item types are more popular than others. Item type r_4 is the least popular, being desired by only two agents, caused by the low popularity value of 2. The valuations are sampled uniformly using

the base values. Agent A_5 has a budget greater than 80, due to the budget resampling. Five examples of the generated item sequences with corresponding revenues are:

index	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
item	r_2	r_4	r_1	r_2	r_2	r_1	r_2	r_4	r_4	r_2	r_2	r_4	r_2	r_2	r_3
price	*59*	*22*	*41*	*58*	*45*	*24*	*42*	*22*	*22*	*25*	*21*	*20*	*21*	*19*	*53*
item	r_2	r_2	r_2	r_4	r_4	r_4	r_2	r_4	r_2	r_4	r_4	r_2	r_2	r_1	r_4
price	*59*	*58*	*45*	*22*	*22*	*22*	*42*	*20*	*30*	*20*	*20*	*25*	*21*	*39*	*20*
item	r_1	r_1	r_4	r_4	r_2	r_4	r_2	r_2	r_4	r_2	r_2	r_4	r_1	r_2	r_2
price	*41*	*38*	*74*	*22*	*58*	*22*	*42*	*30*	*22*	*25*	*20*	*20*	*21*	*21*	*19*
item	r_3	r_4	r_1	r_3	r_4	r_4	r_4	r_4	r_3	r_4	r_2	r_1	r_2	r_1	r_3
price	*61*	*74*	*41*	*53*	*20*	*20*	*20*	*20*	*53*	*20*	*58*	*38*	*42*	*24*	*19*
item	r_2	r_3	r_2	r_4	r_3	r_3	r_1	r_3	r_2	r_3	r_2	r_4	r_4	r_3	r_2
price	*59*	*61*	*58*	*20*	*53*	*53*	*41*	*19*	*45*	*19*	*42*	*20*	*20*	*17.5*	*30*

In the sequences, items of type r_1 or r_3 occur the least frequent, due to the low sparsity values. □

Training For a given set of 20 random bidders, the simulator generates 1000 historical auctions. The 40 items in these auctions are generated using the above scheme, and ordered randomly. These items are run in the simulator, where the agents use the above-mentioned bidding strategies to decide what value to bid, in order to determine the winners and the item selling prices. The total selling price of all items in one sequential auction is the collected revenue. For the 20 generated bidders, we first experiment the effect of different item orderings on the collected revenue by trying 100 random orderings and comparing the smallest, median, and largest collected revenue. If the difference between the largest and smallest is less than one tenth of the median revenue, 20 new bidders are generated. This process is repeated until we find a set of agents that passes this check, which typically occurs after a few iterations. By performing this check, we remove irrelevant problem instances. For the 20 agents that pass this check, we generate 1000 auctions of 40 items and simulate these auctions together with the agents. The resulting sequences of item-price pairs are then transformed to the features discussed in Section 4.3, and the resulting data set is used to train the regression trees and linear regressors.

Testing For the same set of 20 bidders, we generate 5 sets of 40 items, which are used for testing. First, the regressors are tested by comparing their predictions with the revenues generated by the simulator on 50 random orderings of each of these item sets. Second, we translate each of the item sets into constraints for both the black-box and white-box optimization solvers. The best ordering found by these solvers are compared based on their values on the regression model, and in the simulator.

4.5.2 Experimental setup

In each experiment, we generate agents and items as described above. We use an implementation of regression trees and LASSO from the scikit-learn machine learning module (Pedregosa et al. 2011) in Python to learn (and evaluate) the regressors. We learn trees of different depths of 3, 5, 8 (we call them `tree3`, `tree5`, `tree8`), and we set the minimum number of samples required to split an internal node to 10. The LASSO regressor is run with 3 different values for α: 1.0, 0.1, and 0.000001 (`lasso1`, `lasso2`, `lasso3`), the tolerance threshold to test for convergence is set to 0.0001 and we use a maximum of 100000 iterations. The resulting trees and linear models then get translated to ILP, which in turn gets solved by an ILP-solver (CPLEX (IBM 2014)). In addition, we provide the solver with an initial solution (the best of 1000 random orderings) in order to start the search, and set the focus of the solver to finding integer feasible solutions. We set a time limit on the ILP solver of 15 minutes for each instance using a single thread on an Intel core i-5 with 8 GB RAM and record the best ordering of items that the ILP solver has obtained. The last minute is spent on solution polishing (a local search procedure in CPLEX). We apply our best-first search method on the same problem instances with the same running time limit.

Evaluations There are two levels of evaluations involved in our problem. Firstly, we determine the quality of the learned regressors, as they influence the quality of the solution after optimization. For this, we tested different regression trees with different maximum depths and linear regression models with different parameters.

Next, the optimization methods are evaluated in terms of the quality of the produced ordering. The optimization methods that we compare include the proposed

white-box ILP model which finds a solution based on the abovementioned 6 regression models, the proposed black-box best-first search which evaluates a solution based on the 6 regression models, and in addition, two other simple ordering methods: (i) auctioning the most valuable item first (i.e., `mvf`), as suggested in Subramaniam and Venkatesh (2009);[11] and (ii) a random ordering strategy (i.e., `mean5000`), as seen in many real-world auctions for the purpose of fairness.

It is not feasible for us to compute the best solution given the problem size. Thus, we obtain a lower bound on the optimal solution as follows: given a set of items, we generate 5000 random orderings, and we use the *true* model (i.e., the simulator) to evaluate them and pick the one returning the highest revenue. We use the mean value of these 5000 random orderings as the output of the random ordering strategy.

We evaluate the 15 ordering methods in two ways:

- Model evaluation: we use the learned regression models to evaluate the solutions returned by the ordering methods to compute the predicted revenues.
- Actual evaluation: we run auctions with the solutions (i.e., orderings) returned by the ordering methods in our simulator to obtain the corresponding revenues. Note that such an evaluation is possible only when a simulator is available.

There are in total three sets of main experiments, as well as additional experiments, presented in this chapter.

Experiment 1 We simulate a first-price auction where agents bid the minimum value between their budget and the amount that they are willing to pay. The winner pays her winning bid.

Experiment 2 The simulator runs the Vickrey auction with myopic agents, where agents bid the minimum between their true values on each item and their remaining budget. The winner pays the higher value between the second highest bid and the reserve price of the winning item.

Experiment 3 The simulator runs Vickrey auctions with smart agents, where agents bid smartly based on their expected utility at the end of the auction (see Section 4.5.4).

[11] We simply ordered the items according to their base values. We also tested ordering them based on their mean value in the data, but this performed worse in the test.

Experiment 4 In addition, we investigate the scalability of our approach using the following experiments: (1) first, we test the small instances to show how close the orderings found by the learned models are to the actual optimum; (2) second, we generate a randomized population of bidders to test whether our method can handle the cases where the bidders in different auction vary a lot; (3) third, we use a larger set of items and item types to demonstrate how well our approach can be expected to perform in larger auctions with many more items of greater diversity.

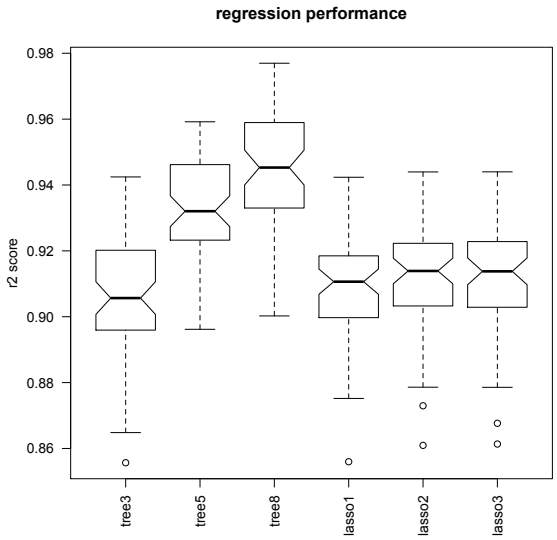


Figure 4.4: Prediction performance of different learning models. The y-axis depicts for each learning model, the obtained prediction accuracy, i.e., R^2 scores (see Equation 4.16). Each box contains 60 values from the 60 different sets of experiments.

4.5.3 Experiment 1: first-price auctions

We run 60 sets of experiments. For each set of experiment, we generate a set of agents, and generate and run new sets of items 5 times. This results in 300 models for each learning method.

Prediction accuracy We first report the performance of the learned trees and linear models in terms of prediction accuracy. Given the same set of items as used during learning, we randomly generate 50 permutations of these items as orderings and compute the predicted values of these orderings using the learned models. Thus, for every set of agents, there are $50 \times 5 \times 40 = 10000$ bids to predict. These predictions are compared with the evaluated values of the orderings by the simulator. We report the coefficients of determination, or R^2 scores, in Figure 4.4. This coefficient is a standard measure for comparing regression models and is defined as:

$$R^2 = 1 - \frac{\sum_{1 \leq d \leq k} (p^d - p(f_1^d, \ldots, f_m^d))^2}{\sum_{1 \leq d \leq k} (p^d - \texttt{mean}(p))^2}, \tag{4.16}$$

where k is the number of samples, p^d is the dth data value (i.e., the revenues returned by the simulator), $p(f_1^d, \ldots, f_m^d)$ the predicted values (i.e., the predicted revenues returned by the learning models: `tree3lp`, `tree5lp`, `tree8lp`, `lasso1`, `lasso2`, `lasso3`), and $\texttt{mean}(p)$ the mean of all data values. Each score is computed from 10000 values in Figure 4.4. A large value (close to 1.0) means the regressor is an almost perfect predictor, and smaller values indicate worse performance. Figure 4.4 shows that all regression models lead to good prediction, with the lowest R^2 score over 0.86. The learned trees with depth 8 give the best performance, followed by the trees with depth 5. Intuitively a larger tree may give a better prediction. The scores of `lasso2` and `lasso3` are very similar, and slightly better than `lasso1`. This result makes sense since LASSO with a higher regularization parameter α (i.e., `lasso1`) implies the use of less features, and hence, may have less prediction power. The tree with depth 3 shows much worse performance than the larger trees, and it is on average worse than the three linear regression models. This is confirmed by the frequencies of wins by comparing the R^2 scores in pairs in Table 4.2.

Table 4.2: The frequencies of wins of 60 runs for each method against others (row method vs column method) in terms of R^2 scores.

	tree3	tree5	tree8	lasso1	lasso2	lasso3	total wins
tree3	0	0	0	27	21	22	70
tree5	60	0	2	60	58	58	238
tree8	60	58	0	60	60	60	298
lasso1	33	0	0	0	7	9	49
lasso2	39	2	0	53	0	28	122
lasso3	38	2	0	51	32	0	123

Table 4.3: The frequencies of wins of 300 runs for each method against others (row method vs column method), evaluated in the simulator.

	tree3lp	tree3bf	tree5lp	tree5bf	tree8lp	tree8bf	lasso1lp	lasso1bf	lasso2lp	lasso2bf	lasso3lp	lasso3bf	mvf	best5000	mean5000	total wins
tree3lp	0	171	105	152	107	125	102	130	71	127	82	119	231	17	240	2068
tree3bf	125	0	83	129	81	105	71	116	61	94	67	86	226	15	230	1770
tree5lp	192	213	0	197	149	168	144	169	111	156	128	164	249	24	273	2632
tree5bf	142	163	97	0	105	122	94	134	74	118	90	115	238	16	241	2037
tree8lp	188	217	142	193	0	178	137	181	113	160	128	162	260	23	287	2668
tree8bf	170	192	130	168	121	0	112	157	91	133	106	138	251	28	269	2361
lasso1lp	193	226	150	202	160	183	0	185	113	178	136	170	263	38	269	2756
lasso1bf	168	181	127	158	113	139	100	0	88	135	91	131	233	18	251	2219
lasso2lp	224	236	184	225	182	205	174	204	0	195	159	199	265	40	286	3073
lasso2bf	171	200	141	177	134	163	115	155	94	0	107	124	243	17	273	2407
lasso3lp	214	231	169	204	167	188	155	201	126	184	0	189	257	36	281	2899
lasso3bf	176	210	131	182	134	161	124	154	89	156	96	0	245	15	276	2439
mvf	68	73	51	61	38	48	36	65	34	55	41	54	0	9	123	982
best5000	282	284	268	284	277	271	257	280	255	278	252	280	291	0	299	4158
mean5000	57	65	27	58	13	30	31	48	12	26	18	24	174	1	0	864

Performance of the ordering methods We now discuss the actual performance of the different ordering methods. After every ordering method returns its best ordering, these orderings are then evaluated in the simulator to get corresponding revenues. As we ran 300 different instances (60 sets of different agents, each with 5 sets of different items), each method has 300 such revenues. We calculate the frequencies of wins by comparing the revenues in pairs in Table 4.3. One obvious conclusion from the table is that the ordering heuristic `mvf` (i.e., most valuable item

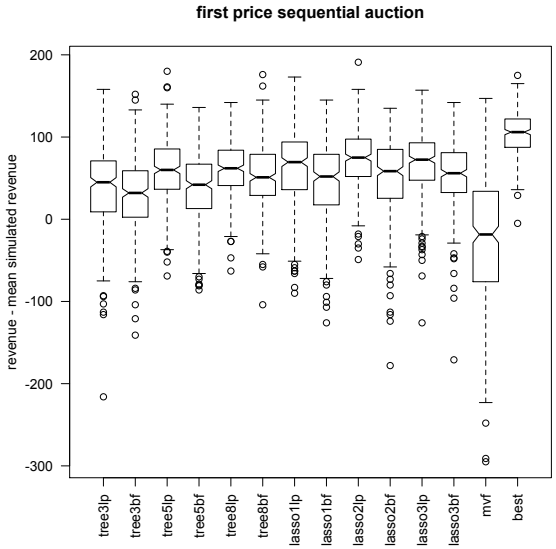


Figure 4.5: The performance of the different ordering methods evaluated by the simulator, compared to `mean5000`. The simulated auctions are first-price auctions. Each box contains 300 values.

first) performs worst, regardless of which method it compared to. In fact, this heuristic performed even worse than the random ordering strategy `mean5000` (123 wins vs. 174).[12] This result contradicts the theoretical finding that was concluded using much simpler auction settings. Another observation is that given the learned models, the developed white-box methods win over the black-box methods more than half of the time. This holds consistently for all 12 proposed methods (wins lp vs. bf: 171, 197, 178, 185, 195, 189). It shows that our new way of utilizing the internal structure of the learned models for optimization is promising.

[12]Notice that there is one instance where `mean5000` is better than `best5000`. This is due to the random scheme that we used in selecting winners who give two identical bids. Consequently, evaluating the same ordering twice in the simulator may result in two different revenues. We want to point out that this does not happen often and the effect is often negligible.

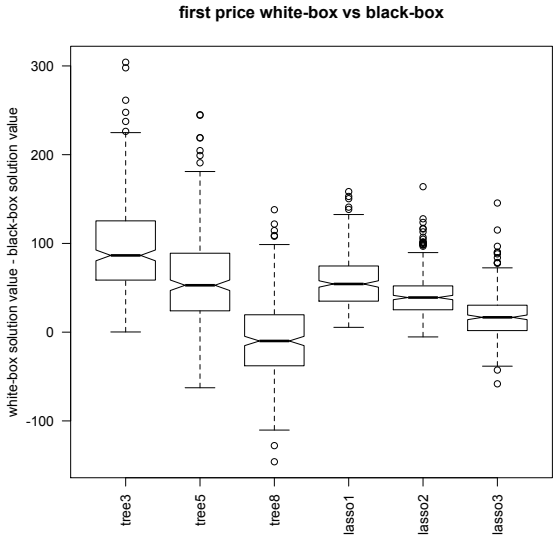


Figure 4.6: The performance difference between the LP model and the best-first search, evaluated by the predictive model. Each box contains 300 values. The simulated auctions are first-price auctions.

If we look at the results of the white-box methods built from the learned regression trees, i.e., `tree3lp`, `tree5lp`, `tree8lp`, we notice that `tree5lp` and `tree8lp` performed similarly and they are slightly better than `tree3lp`. This result is consistent with the higher R^2 scores of the larger trees. Interestingly, despite their lower R^2 scores, the linear regression LP methods return good orderings especially `lasso2lp` and `lasso3lp` which are on average better than the three regression tree LP models. These are further confirmed with Figure 4.5, which depicts the revenue differences in the simulator between the ordering methods and `mean5000`. It is more obvious from this figure that the proposed linear regression LP models are the better optimization methods than the tree LP models in practice. We believe that this is due to the cascading inaccuracies, caused by the `sum` feature that relates the predicted value to the predictions of earlier auctioned items (see discussions in Section 4.3.3). Due to

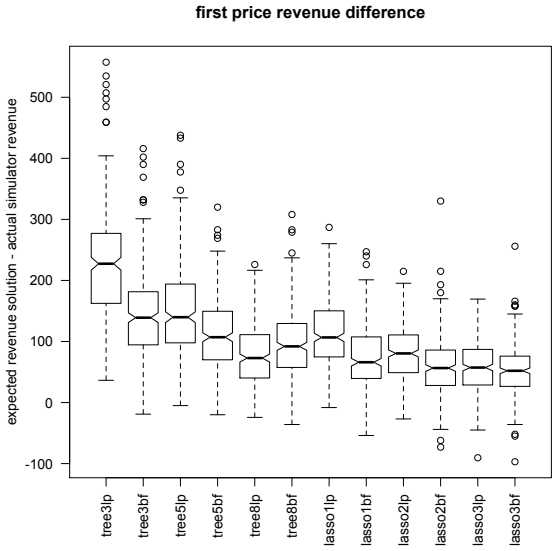


Figure 4.7: The performance difference between the values evaluated by the model and the values evaluated by the simulator. Each box contains 300 values. The simulated auctions are first-price auctions.

the crisp boundaries in regression trees, the effect of these errors on the solution evaluation is much greater than using linear regressors. We believe the effect is smaller for larger trees because they are more accurate.

Figure 4.6 shows the performance difference between the LP model and the best-first search, which are built upon the same learned regression model. The solutions are evaluated using the predictive models. The value differences between the white-box and the black-box methods are more significant on smaller trees than on bigger trees (`tree3` vs `tree5` vs `tree8`), and on linear models with higher regularization parameter than on lower regularization parameter (`lasso1` vs `lasso2` vs `lasso3`). This trend shown in the results is somehow expected. The smaller tree leads to a smaller LP model, which is easier to optimize by the solver and consequently gives a much better performance than the best-first search. Similarly, the linear regression

with a higher regularization parameter α implies less feature values to compute during white-box optimization, and therefore, its advantage over the black-box method is more obvious than `lasso2` and `lasso3` which are with smaller values of α.

We observe that all white-box LP methods are better than the black-box methods, except the LP models resulting from the trees with depth 8. The depth 8 regression trees perform better for best-first search when evaluated on the model (Figure 4.6), but better for LP when evaluated in the simulator (see Figure 4.5). The most likely reason for the strange behavior of the depth 8 trees is that the best-first search outperforms LP on harder to predict instances.[13] Intuitively, because harder-to-predict instances typically result in larger models, they are harder to optimize in CPLEX. We checked this cause by investigating whether the R^2 scores of the depth 8 regression tree are correlated with which method performing better in the model evaluation. The mean of these R^2 scores are 0.950 when the LP performs better, and 0.942 when the best-first performs better. Although this difference seems small, it is significant.

Moreover, the small difference in R^2 scores between trees with depth 5 and depth 8 also has a significant effect on the difference between their model and simulator evaluations (e.g., due to cascading errors). To demonstrate this, we report in Figure 4.7 the solution differences of the same ordering methods when being evaluated by the model and the simulator. The purpose of this comparison is to test whether the predicted outcome (using learned models) corresponds to the actual outcome (using simulator). This test is important as in general, there are no simulators available to evaluate the solutions. The figure demonstrates that the linear regression based optimization methods return more reliable solutions, i.e., their solutions evaluated on the learned linear models are closer to the solution values returned by the simulator. Note that it is logical that most values are over estimated because we the optimization tries to solve a maximization problem. The trees with depth 5 and 8

[13] Another possible cause is that the LP solver makes small rounding errors. Such errors are unavoidable because using both very small and very large coefficients and/or precision in one problem formulation can cause numerical instability. We therefore round the mean values of the leaf predictions to two digits after the decimal point in the regression tree models before translating it to LP. Unfortunately, due to the crisp decision boundaries in regression trees, these small errors can sometimes have a large effect.

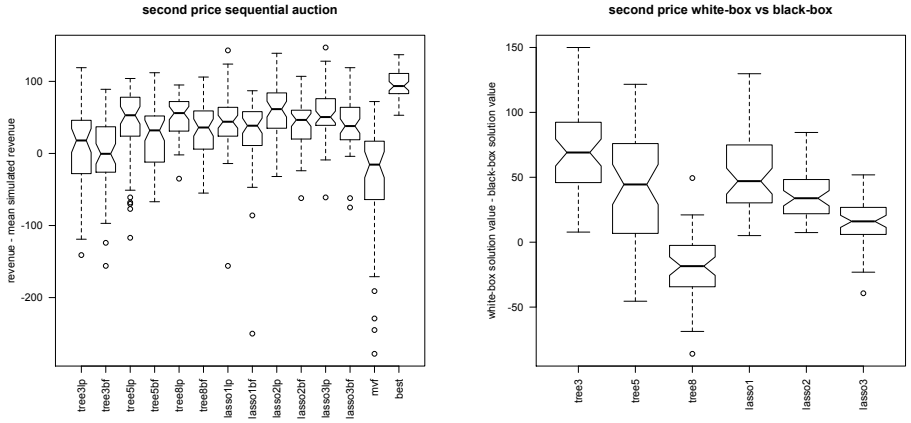


Figure 4.8: Performance of the different ordering methods with second-price auctions and truth-telling, myopic agents. The two figures show the performance evaluated by the simulator and the model respectively. Each box contains 50 values.

show a significant difference in this evaluation. The depth 3 trees end up with the highest evaluation difference, and overestimate the solution values the most.

4.5.4 Experiments 2 and 3: Vickrey auction with myopic and smart agents

In order to demonstrate that our method of auction optimization using learned models is robust to the used auction rule or the bidding strategies, we test it in a second-price auction in Experiment 2, and with smart agents in Experiment 3. We generate 10 sets of agents using the settings of the main experiments. For each set of agents we run new sets of items 5 times. Figures 4.8 and 4.9 show the same plots for these settings as Figures 4.5 and 4.6 for the setting in the first experiment.

In the second-price experiment, we test with truth-telling bidders in second-price auctions. The only differences with the setting in the first experiment are the bidding values and the payments. As Figure 4.8 shows, the results are very similar to those in the first experiments: (1) all methods outperform the naive ordering strategies from the literature, (2) the white-box outperforms the black-box methods, and (3)

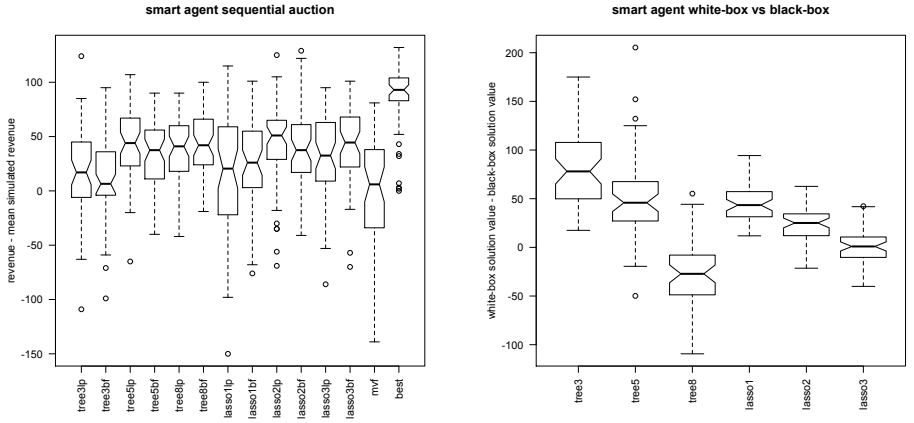


Figure 4.9: Performance of the different ordering methods with second-price auctions and smart agents. The two figures show the performance evaluated by the simulator and the model respectively. Each box contains 50 values.

the linear regression models perform best. However, the difference between the best performing methods `tree5` and `lasso2` is no longer significant.

In the third experiment, we test with smart agents that aim to maximize their final utility in a second-price auction. Specifically, when deciding what value to bid on the current item r_i, they have access to the auction simulator and use it to run the remaining items $I' \setminus r_i$. For computational reasons, when running the remaining items $I' \setminus r_i$, it is assumed that all agents bid truthfully and pay according to the second-price rule in these runs. For every bid r_i, they run the simulator twice: once in the situation where they bid the item r_i with their valuations (`run1`), and once where they do not buy the item (`run2`). They then decide what value to bid according to the following rules:

- If after `run2` the agent has a remaining budget greater than its value for the item, it bids truthfully. The intuition is that it is better to buy an item than to have budget left at the end of auction.
- Else, if the total utility after `run2` is less than after `run1`, it also bids truthfully. When it is better to buy an item, try to obtain it.

- Else, it bids its true value minus the difference in utility after `run2` and `run1`. When it is x monetary units better not to buy an item, try to obtain it for the value minus x.

Using these rules, the agents bid the highest value that they expect will give them an increase in utility.

As can be seen in Figure 4.9, also in this challenging setting, our method performs significantly better than the naive ordering rules from the literature. There is however a much larger variance in the performance of the different methods, causing the difference between black-box and white-box to be insignificant in the simulator. When evaluated on the model, however, white-box is still better than black-box in all cases except `tree8`. An interesting final observation is that, with these smart agents, the `mvf` method does seem to perform slightly better than `mean5000` (although not significantly), while in the other experiments it performed consistently worse.

4.5.5 Experiment 4: practical issues

Although optimizing the orderings in sequential auctions is a hard problem, the above experiments demonstrate that high revenues can be obtained, significantly outperforming the naive methods proposed in the literature. This holds even in the presence of smart bidders. They also show the advantage of using the white-box method for optimization, an interesting trade-off between modeling and optimization power, and that better predictors are not necessarily better optimizers. In this section, we investigate several practical issues of our approach. We start with the easier instances, which will be used to show how close the orderings found by the learned models are to the actual optimum. Afterwards, we test our method using two new settings: a randomized population of bidders and a larger set of items and item types. The randomized population mimics a feature of a type of real-world auctions: the agents bidding in an auction are not always the same as often seen in online auctions. The larger instances aim to demonstrate how well our approach can be expected to perform in large-scale auctions with many more items of greater diversity.

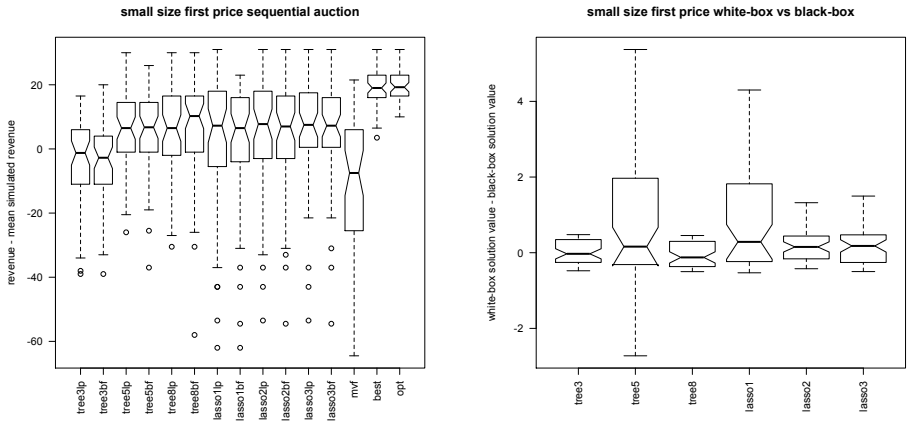


Figure 4.10: Performance of the different ordering methods and a brute-force search (opt) for 15 items from 4 types. The figures show the performance evaluated by the simulator and the model respectively. Each box contains 50 values.

Smaller auctions

The smaller problem instances are generated with: 1) a maximum budget of 80, 2) a maximum number of 3 desired items, 3) 4 item types, 4) 15 items per auction, and 5) 8 agents. All other settings are the same as those used in the main experiments. In addition, we added a brute-force method that uses the simulator as a black-box and tests all possible ordering of the items. Although a brute-force method is infeasible for 40 items, in the case of 15 items, there are only a few million possible orderings to consider. The results of 10 sets of agents, each tested with 5 sets of items, are shown in Figure 4.10. The corresponding plots for the main experiments are Figures 4.5 and 4.6.

The R^2 scores show a behavior similar to the main experiments, except that they range between 0.8 and 0.9. Outliers for the depth 3 trees can get as low as 0.5. This score decrease is likely caused by the decrease in data size (15 instead of 40 data rows per auction). The left plot if Figure 4.10 shows the results obtained using the simulator. Any difference in performance between the different methods is insignificant except for depth 3 trees, which are significantly worse at estimating the

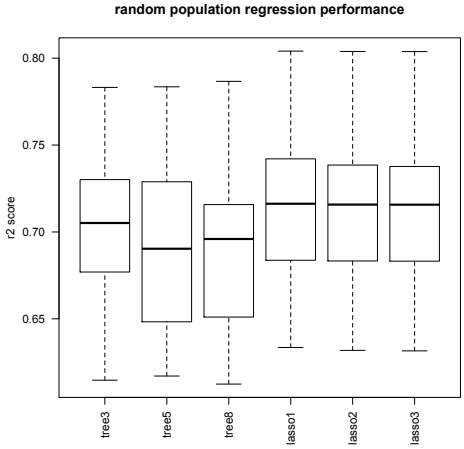
obtained revenues and therefore result in lesser quality solutions. The improvement in performance of the other methods over random is approximately 10 on average, whilst about 20 could be obtained with perfect models in theory. Frequently, the black-box and white-box methods find an optimal solution for the learned models, so this difference is entirely due to the prediction quality of the models. It is good to see that the performance of the `mean5000` method is almost indistinguishable from optimal (the last boxplot in the figure), giving confidence that this is a good approximation of the upper bound. As can be seen in the right plot, the difference between the white-box and black-box methods are negligible for these small instances.

Randomized population

In this set of experiments, we obtain different sets of agents by creating a large population of size 60, and drawing 20 of them at random for every auction. In addition, we generate a random set of 20 to 60 items for every auction, sampled uniformly at random. Other than that, we use the exact same settings as the main experiments. We show the results of 19 sets of agents (we increased the set to reduce noise in the results), each being tested with 5 sets of items, see Figure 4.12. The R^2-scores obtained in these experiments are shown in Figure 4.11.

From Figure 4.11, we make two observations. Firstly, the performance of the predictors is much worse than in the main experiments, with median values around 0.7. As one may expect, the revenue is very hard to predict in this randomized setting, because it is heavily dependent on which agents take part in the auction. Learning larger trees does not increase this score. In fact, as the wins table shows, the depth 8 trees are clearly the worst predictor. Secondly, in contrast to the main experiments, linear regression appears to be a better estimator than a regression tree, although the difference is very small.

The simulator and model results in Figure 4.10 confirm that the regression functions perform much worse in this randomized population setting. The linear regression models still perform better than random. The difference with the expected value of a random solution is so small that in cases with very uncertain populations it will probably not be worthwhile to model and optimize the problem. The model results show that also in these cases, white-box methods outperform black-box ones, with



	tree3	tree5	tree8	lasso1	lasso2	lasso3	total wins
tree3	0	9	17	6	6	6	44
tree5	10	0	15	4	5	5	39
tree8	2	4	0	0	0	0	6
lasso1	13	15	19	0	18	18	83
lasso2	13	14	19	1	0	17	64
lasso3	13	14	19	1	2	0	49

Figure 4.11: Prediction performance of different learning models on the randomized population, including the number of wins table. Each box contains 19 values from the 19 different sets of experiments.

the only exception for depth 8 regression trees where the performance of white-box and black-box is similar.

Larger problem instances

The larger problem instances are generated with the same settings as those used in the main experiments, expect for: 1) 12 item types, 2) 80 items per auction, and 3) 40 agents. In addition, because the optimization problems are harder, we used a timeout of 30 minutes instead of 15. The results of 10 sets of agents, each tested with 5 sets of items, are shown in Figures 4.13 and 4.14.

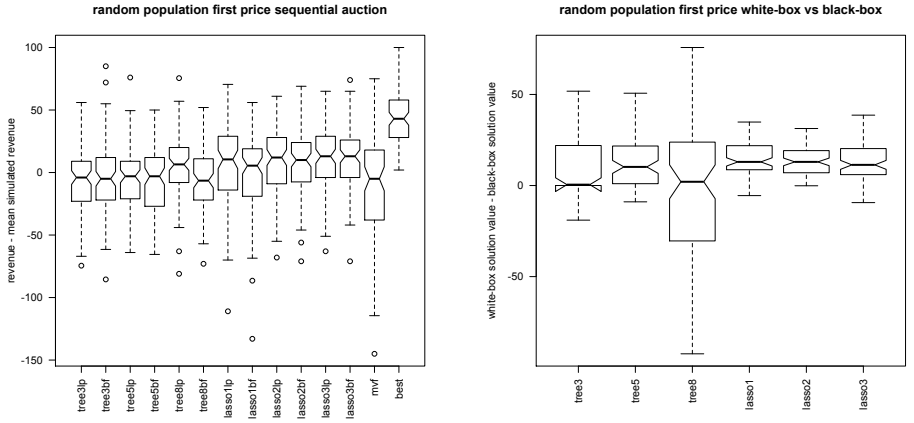


Figure 4.12: Performance of the different ordering methods for 20 to 60 items from 8 types for a random population of 20 agents selected from a pool of 60. The two figures show the performance evaluated by the simulator and the model respectively. Each box contains 95 values.

Notice that the performance for depth 8 trees is missing in Figure 4.13. The reason for this is that the CPLEX runs for these trees ran out of memory. The encoding requires $n \times |R| \times (2^k + 1)$ variables, and $n \times |R| \times (2^(k+1) - 1) + n + |R|$ constraints. If we fill in depth 8, 12 types, and 80 items, we end up with 246720 variables and 490652 constraints. This proved too much for our version of CPLEX (v12.5.1). Although the smaller trees did run, the depth 5 trees are now outperformed by the black-box approach. For depth 3 trees, it is still better to use white-box. A similar phenomenon is visible in Lasso results. In all previous results, white-box always outperforms black-box for the Lasso regressors. Suddenly, this is only the case for the smallest linear models, in those with more coefficients black-box performs better. These results match our intuition that using a white-box approach is beneficial when the models are not overly complex.

In the simulator results, we see that smaller models perform worse than larger ones, in spite of them being much harder to optimize. We believe this is due to them being worse predictors, as can be seen in the wins table in Figure 4.14. Interestingly,

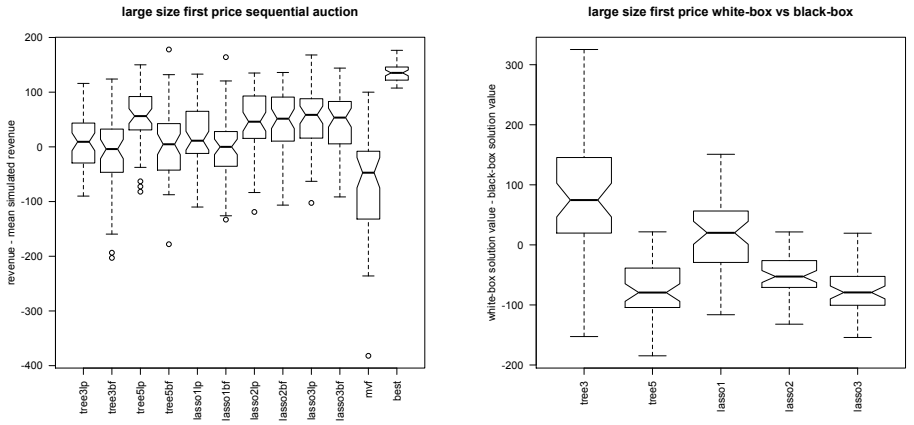
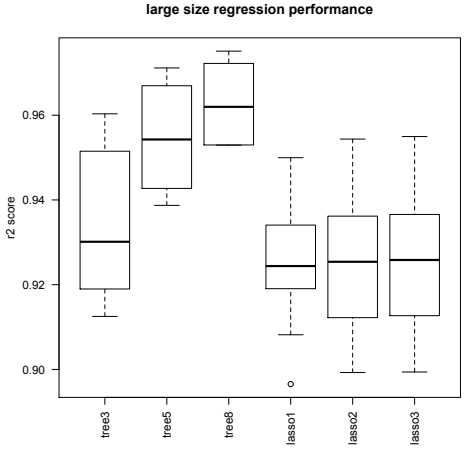


Figure 4.13: Performance of the different ordering methods for 80 items from 12 types and 40 bidders. The two figures show the performance evaluated by the simulator and the model respectively. Each box contains 50 values.

for the larger linear models, the performance difference between black-box and white-box is no longer there. For the depth 5 tree, the difference is inverted, which we believe to be due to chance. However, it is also possible that this effect is caused by over estimation. For the black-box method, the average over estimation of solutions is 237, while it is 111 for the white-box method. This can be caused by the fact that CPLEX has problems finding solutions for these large instances, making it also harder to find very specific solutions that over estimate the solution value. Investigating this behavior further is left as future work.

Figure 4.14 also shows an increase in R^2 scores: all median values are above the 0.92 mark. This is most likely cause by an increase in training data (twice as many items are sold per auction). The tree models are much better regressors for the large instances, although their optimization performance in the simulator is much worse. In fact, their performance is close to that of a random ordering. The larger linear models perform better than random, but are still far from optimal.



	tree3	tree5	tree8	lasso1	lasso2	lasso3	total wins
tree3	0	0	0	9	8	8	25
tree5	10	0	0	10	10	10	40
tree8	10	10	0	10	10	10	50
lasso1	1	0	0	0	1	1	3
lasso2	2	0	0	9	0	2	13
lasso3	2	0	0	9	8	0	19

Figure 4.14: Prediction performance of different learning models for 80 items from 12 types and 40 bidders. The y-axis depicts for each learning model, the obtained prediction accuracy, i.e., R^2 scores (see Equation 4.16). Each box contains 10 values from the 10 different sets of experiments.

4.6 Related work and discussion

We discuss related works and how our work contributes to and from several related research communities.

4.6.1 Interplay between mathematical optimization and machine learning

Many studies have investigated the interplay of data mining and machine learning with mathematical modeling techniques, see overview in e.g. Bennett and Parrado-

Hernández (2006), Meisel and Mattfeld (2010), Corne et al. (2012). Most of these investigate how to use data mining to estimate the value of parameters in decision making models or to replace decision model structure when it cannot be fully determined from the hypotheses at hand. For instance, Brijs et al. (2004) build a decision model as an integer program that maximizes product assortment of a retail store. The decision model is then refined by incorporating additional decision attributes that are the learned patterns from recorded sales data. Li and Ólafsson (2005) use a decision tree to learn dispatching rules that are then used to decide which job should be dispatched first. These dispatching rules are previously unknown, and it is assumed that it is worthwhile to capture the current practices from previous data. Gabel and Riedmiller (2008) model production scheduling problem as multi-agent reinforcement learning where each agent makes its dispatching decisions using a reinforcement learning algorithm based on a neural network function approximation.

Another line of work investigates how to use learning techniques during optimization in order to learn properties of good solutions. For instance, Defourny et al. (2012) combine the estimation of statistical models for returning a decision rule given a state with scenario tree techniques from multi-stage stochastic programming. This line of work shares similarities with the field of black-box optimization, see, e.g., Jones et al. (1998), Shan and Wang (2010), Rios and Sahinidis (2013). In black-box optimization, methods are used to approximate a function with unknown analytical form and which typically is expensive to execute. In contrast, in multi-stage stochastic programming this form is known but stochastic. An often applied technique for black-box optimization is the use of surrogate methods, see, e.g., Koziel et al. (2011). Surrogates are approximations of the black-box function that are less expensive to execute. Typical examples include linear/polynomial regression, neural networks, and other methods from machine learning. These functions are trained during optimization from (as few as possible) black-box function calls.

As learning tasks can lead to challenging optimization problems, researchers have also applied mathematical optimization methods in order to increase learning efficiency. For instance, Bennett and Mangasarian (1993) use linear programming for determining linear combination splits within two-class decision trees. Chang et al. (2012) propose a Constrained Conditional Model (CCM) framework to incorporate domain knowledge into a conditional model for structured learning, in the

form of declarative constraints. CCMs solve prediction problems. In Uney and Turkay (2006), the authors build a mixed integer program for multi-class data classification. A comprehensive overview of optimization techniques used in learning is given in Sra et al. (2012). Researchers are also interested in using mathematical optimization methods in order to find entire models and rules, see e.g., Carrizosa and Romero Morales (2013), Raedt et al. (2010), Heule and Verwer (2010).

Our approach fits in the first line of research of this interplay. The proposed best-first search method uses regression models to learn good orderings, which is then applied during search to evaluate the solutions of OOSA. Hence, similar to the works mentioned above, the models learned from data are used in a black-box fashion. This approach shares similarities with surrogate methods for black-box optimization. An important difference is that the (surrogate) models here are learned from data. Furthermore, our proposed white-box optimization method makes all the properties of the learned models visible to the optimization solver. Bartolini et al. (2011) propose a similar method by translating neural networks into constraint programming (CP) models. Their approach is simple yet effective and allows to model complex relations (such as recurrent neural networks) between any pair of decision variables based on data. This is very powerful as it allows for multiple trained neural networks to be plugged into an existing CP model that is constructed by traditional means. It is up to the system designer which relations between variables to model traditionally and which to model based on data. In contrast, we use our translation to construct (parts of) the objective function of an existing MIP model using regression functions. To the best of our knowledge, we are the first to combine regression functions that are learned from data with a MIP model in this way.

Other closely related work from the CP literature considers the problem of constraint acquisition (Bessiere et al. 2005, Bessière et al. 2007, Beldiceanu and Simonis 2012, Bessiere et al. 2013). Given a large set of possible constraints L and some training data, the goal of constraint acquisition is to compose a constraint network (a graphical representation of a CP model, see, e.g., Dechter (2003)) using the constraints in L such that it classifies all of the training data correctly, i.e., it should specify a language that includes all positive and excludes all negative training examples. Although this is a hard problem, several effective approaches have been proposed based on supervised (Bessiere et al. 2005), unsupervised (Beldiceanu and

Simonis 2012), and active (Bessière et al. 2007, Bessiere et al. 2013) learning. These approaches are based on their own new learning algorithms, some with proven performance bounds (e.g., Bessiere et al. (2013)).Once learned, the constraints are embedded into an existing CP model and fully integrated into the optimization process. From a modeling perspective, this method has both benefits and downside. On one hand, constraint acquisition allows the modeler to define his or her own dedicated constraint set L instead of only the features/variables, potentially resulting in more sensible models for the problem at hand. On the other hand, the learning problem itself is arguably less understood than those tackled by existing algorithms from machine learning, potentially resulting less quality classifiers.

In addition, in probabilistic inference, an interesting combination of machine learning and optimization has also been proposed. First, a probabilistic model is learned in the traditional way. Then, when computing the posterior distribution over some target variables given new input data, additional constraints are added in order to limit the possible assignments to the targets. While the original probabilistic inference problem can often be solved using dynamic programming methods, the additional constraints make it much harder to solve, and it is therefore translated into a MIP. This approach has successfully been applied in order to add expert knowledge to conditional random fields for semantic role labeling in Roth and Yih (2005). Later, the same principle was used to model the dependencies among textual tokens in text-based documents for entity recognition by Fersini et al. (2014). Interestingly, in this last work, the added constraints themselves were also learned from data and added as soft constraints to the MIP model.

4.6.2 Sequence models

As an auction ordering is essentially a sequence of items, our work is also related to the many machine learning approaches for sequence modeling. To the best of our knowledge none of the existing sequence models fits our auction setting. Language models such as deterministic automata (De la Higuera 2010) are too powerful since they can model every possible sequence independently and therefore require too much data to learn accurately. Short sequence models such as hidden Markov models or

N-grams (Bishop 2006) do not model the dependence on items sold a long time (more than the sliding window length) before.

Markov decision processes (MDPs) (see, e.g., Puterman (2009)) may be closest to our auction setting, as they can directly model the expected price per item and come with methods that can be used to optimize the expected total reward (revenue). However, we notice that a straightforward implementation of the auction design problem as an MDP is not possible. Let us try to model auction design as an MDP. Because of the Markov assumption, every state in this MDP has to contain all the relevant information for the auctioneer's decision on which item to auction: the set of available items, the bidders' valuation functions, budgets and strategies, and for every bidder the items (s)he already possesses. In every state q of this process, the auctioneer can choose what item i to put to auction from a multiset of available items I. The next state q' resulting from auctioning item i depends on the bidders and their valuations. These are unknown to the auctioneer, but probabilities can be used to estimate them. These probabilities $P_i(q, q')$ provide a distribution over the possible next states and the corresponding rewards $R_i(q, q')$, given i is auctioned. In every possible next state q', the set of available items is equal to $I - \{i\}$, i.e., equal to the items in q minus the sold item i. The goal of the auctioneer is to maximize the expected rewards (revenue) for a given set of items I. In every state q, (s)he thus has to take an action (choosing an item) that maximizes the sum of the expected rewards $V(q, I)$ of items in I starting in state q:

$$V(q, I) = \arg\max_{i \in I} \left(\sum_{q'} P_i(q, q') R_i(q, q') + V(q', I - \{i\}) \right).$$

In this equation, we separated I from q to highlight the major hurdle that needs to be overcome in order to represent the auction design problem as an MDP. I needs to be included in the MDP since it determines the set of available actions in every state. However, since the set of items is finite this makes the MDP acyclic and at least as large as the number of possible subsets of items from I (assuming the effect of their ordering is represented differently), i.e., at least $2^{|I|}$. In order to learn the rewards and transition probabilities, an auctioneer would therefore need an extremely large data sample.

This intuitively shows why it is difficult to represent the auction design problem as an MDP. However, with a suitable factored representation of the states and/or function approximation (Puterman 2009) of the rewards, it could be possible to represent our auction problem as an MDP. In this case, a major hurdle will be to find a representation that results in Markovian states, which is needed to apply the dynamic programming methods. Since the problem of deciding whether good auction ordering exists is NP-complete (Theorem 5), and these methods run in polynomial time, this is impossible without an exponentially large state space unless $\mathsf{P} = \mathsf{NP}$. Our method relies on solvers and search methods for NP-complete problems, making a polynomial state space possible, and therefore requiring much less data to estimate the model parameters.

4.6.3 Auction design

In the auction literature, a few existing papers investigate the impact of ordering on the performance of sequential auctions. One line of related research focuses on theoretical analysis. In the economics literature (see Elmaghraby (2003), Pitchik (2009), Subramaniam and Venkatesh (2009)), such theoretical studies were typically carried out under very restricted markets. The main research focus there is to analyze equilibrium bidding strategies of bidders who compete for (usually) two items (heterogeneous or homogeneous), and then to gain insights on the impact of ordering on the auction outcome based on derived bidding behaviors. For instance, Elmaghraby (2003) studies the influence of ordering on the efficiency of the sequential second price procurement auctions, where a buyer outsources two heterogeneous jobs to suppliers with capacity constraints. Suppliers can only win 1 job in this setting. The author shows that specific sequences lower procurement costs and identifies a class of bidders' cost functions where the efficient orderings (i.e. the auction rewards the jobs to the suppliers with the lowest total costs) and equilibrium bidding strategies exist. Pitchik (2009) points out that in the presence of budget constraints, a sealed-bid sequential auction with two bidders and two goods may have multiple symmetric equilibrium bidding functions, and the ordering of sale affects the expected revenue. If the bidder who wins the first good has a higher income than the other one, the expected revenue is maximized. Subramaniam and Venkatesh (2009) investigate the

optimal auctioning strategy of a revenue-maximizing seller, who auctions two items, which could be complements or substitutes. They show that when the items are different in value, the higher valued item (among the two) should be auctioned first in order to increase the seller's revenue. A similar revenue-maximizing strategy is proposed by Benoit and Krishna (2001) in a complete information auction setting. The authors conclude that in such a setting, when selling two items to budget constrained bidders, it is always better to sell the more valued item first. However, this strategy does not optimize the revenue anymore when more than two items are to be auctioned.

Empirical research has been conducted to test the theoretical findings in the economics community. Grether and Plott (2009) report on a field experiment that tests the ordering strategies of a seller in sequential, ascending automobile auctions. They conclude that the worst performing ordering in terms of revenue is for the seller to auction vehicles from highest to lowest values.

In the computer science literature, Elkind and Fatima (2007) study how to maximize revenue in sequential auctions with second-price sealed-bid rules, where bidders are homogeneous, i.e., all their valuations are drawn from public known uniform distributions, and they want to win only one item (but they can bid any of items). In this setting, the authors analyze the equilibrium bids, and develop an algorithm that finds an optimal agenda (i.e., ordering). Vetsikas and Jennings (2010) study a similar auction setting, but unlike Elkind and Fatima (2007), they assume the valuations are known to the bidders at the beginning of the auction. The focus of their work was to compute the equilibrium strategies for bidders. Later, Vetsikas (2013) analyzes the bidding strategies for budget constrained bidders in sequential Vickrey auctions. However, it is a challenge to compute the equilibrium strategies in practice.

We are not the first who consider learning from the previous auctions. However, the difference lies in the fact that the most existing work study how bidders learn from the past information, and update their bids. Boutilier et al. (1999) propose a learning model for bidders to update their bidding policies in sequential auctions for resources with complementarities. The bidding strategies are computed based on the estimated distribution over prices, that is modeled by dynamic programming. Goes et al. (2010) present an empirical study of real sequential online auctions. They analyze the data from an online auction retailer, and show that bidders learn and

update their willingness to pay in repeated auctions of the same item. In Pinker et al. (2010), the authors show the benefits of using earlier auction data for the management of sequential, multi-unit auctions, where the seller needs to split its entire inventory into sequential auctions of smaller lots in order to increase its profit. In their work, an auction feedback mechanism is developed based on a Bayesian model, and it is used to update the auctioneer's beliefs about the bidders' valuation distribution.

Our contribution to the auction literature lies on the fact that our approach can be applied to design optimal auctions based on historical auction data. The advantage of using machine learning and data mining methods is that they are robust to the uncertainty (or noise), and hence have high potential to be used for real-world auction design. Moreover, the approach itself is general and can be applied to many different auction optimization problems, such as finding best reserve price for items for sale, or maximizing social welfare instead of revenue. The necessary changes may include the selection of the features for learning regression models and the encoding in the white-box optimization model.

4.7 Conclusions

Mathematical optimization relies on the availability of knowledge that can be used to construct a mathematical model for the problem at hand. This knowledge is not always available. For instance, in multiagent problems, agents are autonomous and often unwilling to share their local information. Frequently, this autonomy and private information influence the outcome of the optimization, making finding an optimal solution very difficult. In this chapter, we adopt the idea of using machine learning techniques to estimate these influences for an optimization problem with many unknowns: the optimal ordering for sequential auctions (OOSA) problem.

We have demonstrated our approach by transforming historical auctions into data sets for learning regression trees and linear regression models, which subsequently are used to predict the expected value of orderings for new auctions. We proposed two types of optimization methods with learned models, a black-box best-first search approach, and a novel white-box approach that maps learned models to integer linear programs (ILP). We built an auction simulator with a set of bidder agents to simulate

an auction environment. The simulator was used for generating historical auction data, and for evaluating the orderings of items returned by our methods. We ran an extensive set of experiments with different agents and bidding strategies. Although optimizing the orderings in sequential auctions is a hard problem, our proposed methods obtained very high values, significantly outperforming the naive methods proposed in the literature. The experimental results also demonstrate the advantage of using the white-box method for optimization, which significantly outperforms the black-box approach in nearly all settings. In addition, they indicate that when the learned model becomes more complex, it potentially results in more constraints and consequently, an increase in the time needed to solve the problem in a white-box fashion. Since more complex models are (potentially) better predictors, this shows a clear trade-off between modeling and optimization power in white-box optimization. In our opinion, the benefits of the white-box approach largely outweigh the benefits of using black-box optimization.

Finally, the extended experiments demonstrate that although our encodings are efficient, the regression tree breaks down when the data becomes too noisy. An intriguing extension would therefore be to use regression forests instead of individual trees. These are known to handle noisy data much better because of the crisp boundaries in individual trees. The same experiments also show that our method does not yet scale very well with the number of items, most likely due to the increase in the number of trees that need to be evaluated. We expect that a method based on regression forests will therefore require several simplifications or optimizations in order to be feasible.

Besides an improved performance, a very big benefit of the white-box formulation is that it provides a new way of obtaining traditional mathematical models. Our method therefore has many other potential application areas, especially in problems where more and more data is being collected. Even in cases where there already exists a handcrafted optimization model, a model that is learned and translated using our method can easily be integrated into existing (I)LP formulations in order to determine part of the objective function based on data. In this way, one can combine the vast amount of expert knowledge available in these domains with the knowledge in the readily available data. We would like to investigate this combination in the future.

We chose a relatively simple auction model for ease of explanation in this chapter. However, our approach works whenever regression models are able to provide reliable predictions of the bidding values. Hence we believe it can be applied to other auction formats with more complex valuation functions (i.e. combinatorial preferences (Cramton et al. 2006)) and more complex bidding strategies. The results of our method on the larger experiments with 80 items shows that scaling the approach up to large real-world auctions will require several non-trivial simplifications. Moreover, in this chapter, we learned regression trees and linear models from simulated data in order to test the optimization performance. When applying our approach to real-world data, it is important to test whether the regressors assumptions are satisfied. If not, it may be needed to transform or filter them. We plan to discover the simplifications and test our approach with real auction data in the near future.

Our experiments highlight some interesting properties of the white-box method. Firstly, they show an improvement in performance when the number of features is reduced and/or the models are less complex. It would therefore be very interesting to investigate the effect of pruning and feature selection or reduction on the performance of our methods. Secondly, they show a tendency of the regression tree optimizer to overestimate, i.e., find orderings that have a much higher expected revenue than their revenue in practice. Intuitively, the solver abuses the crisp nature of the regression tree in order to find a solution that satisfies exactly the right constraints. Part of the problem is that, although these constraints are learned from data, and therefore uncertain, the solver treats them as exact. Fortunately, there exists a long history of methods that try to optimize in the presence of such uncertainties in the area of robust optimization, see, e.g., Ben-Tal et al. (2009). As future work, we will investigate the potential uses of these techniques for learned models.

Recently, regression tree models with linear models in the leaf nodes have also been successfully used as black-box surrogate functions (Verbeeck et al. 2013). Since it is also straightforward to translate these trees given our two encodings (replace the leaf variables by indicators for which linear function to use), it would be very interesting to investigate the possibility of a white-box alternative.

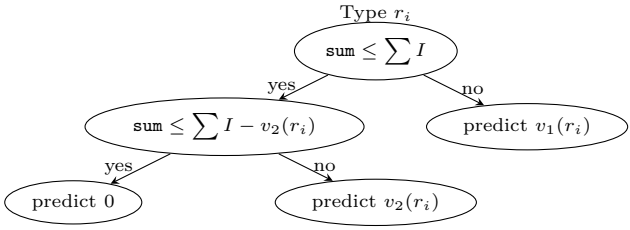


Figure 4.15: The regression tree for every item type r_i used to model a partitioning problem.

Appendix

4.A Hardness of auction design using learned predictors

We show that using predictive models instead of agents with utility functions does not reduce the complexity of the problem: it remains NP-complete for both regression trees and linear regression predictors.

Lemma 4. *Using regression trees, the problem of whether there exists an ordering that has a total predicted value of at least K is NP-complete.*

Proof. The proof follows from the fact that we can use simple regression trees to model the preferences of the two agents from Theorem 1, and evaluating an ordering using these trees can be done in polynomial time. The regression tree for every item type r_i is shown in Figure 4.15. □

Lemma 5. *Using linear regression predictors, the problem of whether there exists an ordering that has a total predicted value of at least K is NP-complete.*

Proof. We prove the lemma using a construction for computing the value of a quadratic function using only linear functions, the ordering problem, and our feature values. The maximum value of this quadratic function is then forced to coincide with the solution of a partition problem instance: Given a set of integers $I = \{v_1, \ldots, v_n\}$, is I dividable into two sets A and B such that $\sum A = \sum B$?

From the partition instance, let $k = \frac{1}{2}\sum_{1\leq i\leq n} v_i$, we construct the following items and linear regression predictors (functions $v()$):

- n items of type $x_1 \dots x_n$, with $v(x_i) = v_i - \frac{v_i \cdot \texttt{sum}(y)}{2k}$, and
- 1 item of type y, with $v(y) = 2k - \sum_{1\leq i\leq n} \texttt{sum}(x_i)$.

The objective is to maximize $\sum_{1\leq i\leq n} v(x_i) + v(y)$. The corresponding decision problem is to ask whether there exists an ordering that achieves a value of $2\frac{1}{2}k$.

($\Rightarrow$) Let A, B be a partition of I such that $\sum A = \sum B$, and let $a_1, \dots, a_{|A|}$ and $b_1, \dots, b_{|B|}$ be the corresponding items of type $x_1 \dots x_n$. The following ordering then gives a value of $\sum_{1\leq i\leq n} v_i$:

$$a_1 \ \dots \ a_{|A|} \ y \ b_1 \ \dots \ b_{|B|}$$

In this ordering, $\sum_{1\leq i\leq |A|} v(a_i) = k$ by definition of A and since $\texttt{sum}(y) = 0$ before item y is auctioned. Consequently, $\sum_{1\leq i\leq n} \texttt{sum}(x_i) = \sum_{1\leq i\leq |A|} v(a_i) = k$, giving $v(y) = 2k - k = k$. $\sum_{1\leq i\leq |A|} v(a_1) + v(y)$ thus already obtains the objective value of $2k$, and therefore $\sum_{1\leq i\leq |B|} v(b_1)$ should be equal to $\frac{1}{2}k$. By definition of B, $\sum_{1\leq i\leq |B|} v(b_i) = k - \frac{k\cdot \texttt{sum}(y)}{2k}$. Since $v(y) = k$, $\texttt{sum}(y) = k$, and thus $\sum_{1\leq i\leq |B|} v(b_i) = k - \frac{k\cdot k}{2k} = k - \frac{k}{2} = \frac{1}{2}k$, proving that the ordering obtains a value of $2\frac{1}{2}k$.

($\Leftarrow$) To prove the other direction, let us further analyze the relation between the objective function $\sum_{1\leq i\leq n} v(x_i) + v(y)$ and the auction ordering. The only term in the $v(x_i)$ predictors that depends on the ordering is $\texttt{sum}(y)$, all other terms are constants. This term is equal to zero for the x_i items auctioned before y, and equal to $v(y)$ for the items auctioned after y. Similar to the ($\Rightarrow$) part, let $a_1, \dots, a_{|A|}$ denote the x_i items before y, $b_1, \dots, b_{|B|}$ those after y, and A, B the corresponding partition of items in I. The objective value is then given by $\sum_{1\leq i\leq |A|} v(a_i) + v(y) + \sum_{1\leq i\leq |B|} v(b_i)$. We analyze these three parts in turn.

- $\sum_{1\leq i\leq |A|} v(a_i) = \sum_{v_i\in A} v_i$ since $\texttt{sum}(y) = 0$ for these items.
- $v(y) = 2k - \sum_{1\leq i\leq |A|} v(a_i) = 2k - \sum_{v_i\in A} v_i = \sum_{v_i\in B} v_i$.

- $\sum_{1 \leq i \leq |B|} v(b_i) = \sum_{v_i \in B} v_i - \sum_{v_i \in B} \frac{v_i \cdot v(y)}{2k}$, since $v(y) = \sum_{v_i \in B} v_i$, this becomes $\sum_{v_i \in B} v_i - \frac{v(y)^2}{2k}$.

Since $\sum_{v_i \in A} v_i + \sum_{v_i \in B} v_i = 2k$, the overall objective function is given by:

$$2k + v(y) - \frac{v(y)^2}{2k}$$

which is maximized when $v(y) = k$ (for $k > 0$) with value $2k + k - \frac{k^2}{2k} = 2\frac{1}{2}k$. This exact value of $v(y) = k$ is obtained when $\sum_{v_i \in B} v_i = k$. The sets A and B thus give a partition of I. □

Remarks. We proved the NP-completeness for the general case of Lemma 5. However, we do not know whether the complexity holds for more realistic valuation functions that bidders have.

Chapter 5

Decision support system for auction design using multi-objective optimization of decision trees[1]

[1]This chapter is based on a working paper.

Chapter 6

Summary and conclusions

In this dissertation we have focused on two different aspects of auctions. First, the allocation of tasks to agents at the end of the auction, and second, the design of the auction itself before it takes place. We employ techniques and methods from both the operations research and computer science fields to aid in these issues.

In Chapters 2 and 3 we study the effect of fairness and the incorporation of the agents' participation behavior in the task allocation of one-shot and multiple auctions. Our results show that fairness comes at a cost in one-shot auctions, but can actually help achieve a better outcome in repeated auctions. In addition, the way agents behave in repeated auctions by participating or withholding from participation has an important effect on the outcome of the auctions.

In Chapters 4 and 5 we use machine learning and mathematical programming models to aid in the auction design process. Our results show that historical data can be used to aid the decision maker in designing an auction that can yield better results for the auctioneer.

In Chapter 2, we study a fair task allocation problem in transportation where an optimal allocation not only has low cost but more importantly, it distributes tasks as even as possible among heterogeneous participants who have different capacities and costs to execute tasks. To tackle this fair minimum cost allocation problem we analyze and solve it in two parts using two novel polynomial-time algorithms. We show that

despite the new fairness criterion, the proposed algorithms can solve the fair minimum cost allocation problem optimally in polynomial-time. In addition, we conduct an extensive set of experiments to investigate the trade-off between cost minimization and fairness. Our experimental results demonstrate the benefit of factoring fairness into task allocation. Among the majority of test instances, fairness comes with a very small price in terms of cost.

In Chapter 3 we investigate how allocations influence agents' decision to participate. We model the agents' participation decision in two ways, using prospect theory and using a fuzzy connective. Both methods use their experiences in the auction thus far. We conduct simulations to investigate the interactions between the agents' participation behaviors and the outcomes of the task allocations in multiple rounds and the long term social welfare. We compare two task allocation algorithms, one merely focusing on costs, and the other focusing on both fairness in the allocation and costs, using the algorithms developed in Chapter 2. The simulation results demonstrate that fairness makes agents more optimistic and incentivizes agents to keep participating. This consequently leads to a higher social welfare in the long run compared to the cost-minimization algorithm.

The proposed fair task allocation algorithms can be applied to other task allocation problems where it is beneficial to spread tasks among different agents. The sharing and access economy, which have been growing rapidly in recent years, is a prime example of where this can be beneficial. These sharing and access economies thrive when there is ample supply, and thus providers need to be incentivized to keep participating. Incorporating fairness in the allocation of tasks to the providers makes the providers feel relevant and will continue providing their services. If not for this mechanism, only the providers with the lowest prices will remain, which are usually only a few who will eventually dominate the market. This effect can already be seen with accommodation rentals in large cities, where a few service providers have many accommodations that they can offer for lower prices than competitors. This makes it harder for someone new to join this economy, for whom the platform was originally intended for.

Also note that the FairMinCost algorithm can be used independently from the IMaxFlow algorithm. Therefore, a non-fair capacity vector can be used, tweaked to

the auctioneer's preferences. It can be applied to any problem in which the capacities are uncertain and variable capacities are needed.

In Chapter 4 we apply machine learning techniques to solve the optimal ordering problem in sequential auctions. We learn regression models from historical auctions and map these to integer linear programs, which we can optimize by using an existing solver. This optimization model results in good orderings with high revenues and has the advantage of being white-box. Furthermore, we show that the internal structure of regression models can be efficiently evaluated inside an ILP solver through efficient encodings of regression trees and linear regression models as ILP constraints. The experimental results show that this white-box approach significantly outperforms the black-box best-first search in nearly all settings.

Finally, we make use of this white-box approach in Chapter 5 using data obtained from an online industrial auction company. We show how to learn a classification tree by combining multiple objectives using integer programming, given the specific user's needs. The resulting classification tree is able to achieve a high precision on low-performing classes of items, which the company is particularly interested in. In addition, we compare the performance of our method with traditional classification tree algorithms, and demonstrate the flexibility of our method, as the trade-off between different learning objectives can be easily adjusted using different weights for the objectives in the integer program.

The white-box formulation demonstrates an additional way of obtaining mathematical models and can be applied in applications where historical data is available. In addition, it can aid already existing mathematical models by adding information obtained from historical data. Therefore, this method provides a way to combine expert knowledge with possible knowledge hidden in the data. Furthermore, due to the structure of mathematical models, one can easily tweak the objective function or add constraints to fit their preferences, which is difficult to do in traditional machine learning techniques. This makes the method flexible in its application and it can aid the auctioneer in the auction design process, learning from similar historical auctions.

References

Ball, M. O., G. L. Donohue, K. Hoffman. 2006. *Combinatorial Auctions*, chap. Auctions for the Safe, Efficient and Equitable Allocation of Airspace System Resources. MIT Press, 949–1015.

Barnhart, C., D. Bertsimas, C. Caramanis, D. Fearing. 2012. Equitable and efficient coordination in traffic flow management. *Transportation Science* **46**(2) 262–280.

Bartolini, Andrea, Michele Lombardi, Michela Milano, Luca Benini. 2011. Neuron constraints to model complex real-world problems. Jimmy Ho-Man Lee, ed., *Principles and Practice of Constraint Programming*, *Lecture Notes in Computer Science*, vol. 6876. Springer, 115–129.

Baruah, S.K., N.K. Cohen, C.G. Plaxton, D.A. Varvel. 1996. Proportionate progress: a notion of fairness in resource allocation. *Algorithmica* **15**(6) 600–625.

Beldiceanu, Nicolas, Helmut Simonis. 2012. A model seeker: Extracting global constraint models from positive examples. Michela Milano, ed., *Principles and Practice of Constraint Programming*, *Lecture Notes in Computer Science*, vol. 7514. Springer, 141–157.

Belk, R. 2014. You are what you can access: sharing and collaborative consumption online. *Journal of Business Research* **67**(8) 1595–1600.

Ben-Tal, A., L. El Ghaoui, A.S. Nemirovski. 2009. *Robust Optimization*. Princeton Series in Applied Mathematics, Princeton University Press.

Bennett, Kristin P., O. L. Mangasarian. 1993. Bilinear separation of two sets in n-space. *Computational Optimization and Applications* **2** 207–227.

Bennett, Kristin P., Emilio Parrado-Hernández. 2006. The interplay of optimization and machine learning research. *Journal of Machine Learning Research* **7** 1265–1281.

Benoit, J.-P., V. Krishna. 1985. Finitely repeated games. *Econometrica* **53**(4) 905–922.

Benoit, Jean-Pierre, Vijay Krishna. 2001. Multiple-object auctions with budget constrained bidders. *Review of Economic Studies* **68**(1) 155–79.

Bernhardt, Dan, David Scoones. 1994. A note on sequential auctions. *American Economic Review* **84**(3) 653–657.

Bertsekas, D. P., R. G. Gallager, P. Humblet. 1987. *Data networks*, vol. 2. Prentice-hall Englewood Cliffs, NJ.

Bertsimas, D., V. F. Farias, N. Trichakis. 2011a. The price of fairness. *Operations Research* **59**(1) 17–31.

Bertsimas, D., V. F. Farias, N. Trichakis. 2012. On the efficiency-fairness trade-off. *Management Science* **58**(12) 2234–2250.

Bertsimas, D., G. Lulli, A. Odoni. 2011b. An integer optimization approach to large-scale air traffic flow management. *Operations Research* **59**(1) 211–227.

Bertsimas, D., S.S. Patterson. 1998. The air traffic flow management problem with enroute capacities. *Operations Research* **46**(3) 406–422.

Bessiere, Christian, Remi Coletta, Emmanuel Hebrard, George Katsirelos, Nadjib Lazaar, Nina Narodytska, Claude-Guy Quimper, Toby Walsh, et al. 2013. Constraint acquisition via partial queries. *International Joint Conference on Artificial Intelligence*, vol. 13. 475–481.

Bessiere, Christian, Remi Coletta, Frédéric Koriche, Barry O'Sullivan. 2005. A sat-based version space algorithm for acquiring constraint satisfaction problems. *Machine Learning: ECML 2005*. Springer, 23–34.

Bessière, Christian, Remi Coletta, Barry O'Sullivan, Mathias Paulin. 2007. Query-driven constraint acquisition. Manuela M. Veloso, ed., *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*. 50–55.

Bichler, M., A. Davenport, G. Hohner, J. Kalagnanam. 2006. *Combinatorial Auctions*, chap. Industrial Procurement Auctions. MIT Press, 1116–1147.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Boutilier, Craig, Moises Goldszmidt, Bikash Sabata. 1999. Sequential auctions for the allocation of resources with complementarities. *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 527–534.

Breiman, L., J. Friedman, R. Olshen, C. Stone. 1984. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA.

Brijs, Tom, Gilbert Swinnen, Koen Vanhoof, Geert Wets. 2004. Building an association rules framework to improve product assortment decisions. *Data Mining and Knowledge Discovery* **8**(1) 7–23.

Camerer, Colin F. 2004. Prospect theory in the wild: evidence from the field. *Advances in behavioral economics* 148–161.

Campbell, A. M., D. Vandenbussche, W. Hermann. 2008. Routing for relief efforts. *Transportation Science* **42**(2) 127–145.

Carrizosa, Emilio, Dolores Romero Morales. 2013. Review: Supervised classification and mathematical optimization. *Computers and Operations Research* **40**(1) 150–165.

Castelli, L., P. Pellegrini, R. Pesenti. 2011. Airport slot allocation in europe: economic efficiency and fairness. *International Journal of Revenue Management* **6**(1-2) 28–44.

Chang, M., L. Ratinov, D. Roth. 2012. Structured learning with constrained conditional models. *Machine Learning* **88**(3) 399–431.

Condorelli, D. 2007. Efficient and equitable airport slot allocation. *Rivista di Politica Economica* **1** 81–104.

Corne, David, Clarisse Dhaenens, Laetitia Jourdan. 2012. Synergies between operations research and data mining: The emerging use of multi-objective approaches. *European Journal of Operational Research* **221**(3) 469 – 479.

Cramton, P. 2002. *Handbook of Telecommunications Economics*, chap. Spectrum Auctions. Elsevier, 605–639.

Cramton, P.C., Y. Shoham, R. Steinberg. 2006. *Combinatorial auctions*. MIT Press.

De la Higuera, Colin. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press, New York, NY, USA.

Dechter, Rina. 2003. *Constraint processing*. Morgan Kaufmann.

Defourny, Boris, Damien Ernst, Louis Wehenkel. 2012. Scenario trees and policy selection for multistage stochastic programming using machine learning. *Journal on Computing* Published online before print.

Dietterich, Thomas G. 2002. Machine learning for sequential data: A review. *Structural, syntactic, and statistical pattern recognition*. Springer, 15–30.

Duinkerken, M. B., R. Dekker, S. T. G. L. Kurstjens, J. A. Ottjes, N. P. Dellaert. 2006. Comparing transportation systems for inter-terminal transport at the maasvlakte container terminals. *OR Spectrum* **28** 469–493.

Dyckhoff, Harald, Witold Pedrycz. 1984. Generalized means as model of compensative connectives. *Fuzzy Sets and Systems* **14** 143–154.

Edmonds, J., R. M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)* **19**(2) 248–264.

Eiselt, H.A., V. Marianov. 2008. Employee positioning and workload allocation. *Computers & Operations Research* **35**(2) 513–524.

Elkind, Edith, Shaheen Fatima. 2007. Maximizing revenue in sequential auctions. *Proceedings of the 3rd international conference on Internet and network economics*. WINE'07, Springer-Verlag, Berlin, Heidelberg, 491–502.

Elmaghraby, Wedad. 2003. The importance of ordering in sequential auctions. *Management Science* **49** 673–682.

Eriksson, L., J. Garvill, A. M. Nordlund. 2008. Acceptability of single and combined transport policy measures: the importance of environmental and policy specific beliefs. *Transportation Research Part A* **42**(8) 1117–1128.

Ernst, A.T., H. Jiang, M. Krishnamoorthy, D. Sier. 2004. Staff scheduling and rostering: a review of applications, methods and models. *European Journal of Operational Research* **153**(1) 3–27.

Fan, W., R. Machemehl, N. Lownes. 2008. Carsharing: dynamic decision-making problem for vehicle allocation. *Transportation Research Record: Journal of the Transportation Research Board* **2063** 97–104.

Fersini, E., E. Messina, G. Felici, D. Roth. 2014. Soft-constrained inference for named entity recognition. *Information Processing & Management* **50**(5) 807–819. doi: 10.1016/j.ipm.2014.04.005. URL `http://dx.doi.org/10.1016/j.ipm.2014.04.005`.

Fujii, S., T. Gärling, C. Jakobsson, R.-C. Jou. 2004. A cross-country study of fairness and infringement on freedom as determinants of car owners' acceptance of road pricing. *Transportation* **31**(3) 285–295.

Gabel, Thomas, Martin Riedmiller. 2008. Adaptive reactive job-shop scheduling with learning agents. *International Journal of Information Technology and Intelligent Computing* **2**(4) 1–30.

Gallien, J., L. M. Wein. 2005a. A smart market for industrial procurement with capacity constraints. *Management Science* **51**(1) 76–91.

Gallien, Jérémie, Lawrence M. Wein. 2005b. A smart market for industrial procurement with capacity constraints. *Management Science* **51** 76–91.

Garey, M R, D S Johnson. 1979. *Computers and intractability – a guide to the theory of NP-completeness*. W.H. Freeman and company.

Ghodsi, A., M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, I. Stoica. 2011. Dominant resource fairness: fair allocation of multiple resource types. *NSDI*, vol. 11. 24–24.

Goes, Paulo B., Gilbert G. Karuga, Arvind K. Tripathi. 2010. Understanding willingness-to-pay formation of repeat bidders in sequential online auctions. *Information Systems Research* **21** 907–924.

Goldberg, A. V., R. E. Tarjan. 1989. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM (JACM)* **36**(4) 873–886.

Goldman, T., R. Gorham. 2006. Sustainable urban transport: four innovative directions. *Technology in Society* **28**(1) 261–273.

González-Pachón, Jacinto, Carlos Romero. 2016. Bentham, marx and rawls ethical principles: in search for a compromise. *Omega* **62** 47–51.

Gopinathan, A., Z. Li. 2011. Strategyproof auctions for balancing social welfare and fairness in secondary spectrum markets. *INFOCOM, 2011 Proceedings IEEE*. IEEE, 3020–3028.

Grether, David M., Charles R. Plott. 2009. Sequencing strategies in large, competitive, ascending price automobile auctions: An experimental examination. *Journal of Economic Behavior & Organization* **71**(2) 75–88.

Heck, Eric Van, Pieter M. A. Ribbers. 1997. Experiences with electronic auctions in the dutch flower industry. *Electronic Markets* **7**(4) 29–34.

Heule, Marijn J.H., Sicco Verwer. 2010. Exact DFA identification using SAT solvers. *Grammatical Inference: Theoretical Results and Applications*, *Lecture Notes in Computer Science*, vol. 6339. Springer Berlin Heidelberg, 66–79.

Hoffman, R., J. Burke, T. Lewis, A. Futer, M. Ball. 2005. Resource allocation principles for airspace flow control. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 6470.

Huyet, A.L. 2006. Optimization and analysis aid via data-mining for simulated production systems. *European Journal of Operational Research* **173**(3) 827–838.

IBM. 2014. Ibm ilog cplex optimizer. URL `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

Jaffe, J. M. 1981. Bottleneck flow control. *Communications, IEEE Transactions on* **29**(7) 954–962.

Jap, S. D., E. Haruvy. 2008. Interorganizational relationships and bidding behavior in industrial online reverse auctions. *Journal of Marketing Research* **45**(5) 550–561.

JGraphT. 2014. Jgrapht. URL `http://jgrapht.org/`.

Jones, Donald R, Matthias Schonlau, William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization* **13**(4) 455–492.

Karp, R. M. 1978. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics* **23**(3) 309–311.

Kaymak, U., H. R. van Nauta Lemke. 1998. A sensitivity analysis approach to introducing weight factors into decision functions in fuzzy multicriteria decision making. *Fuzzy Sets and Systems* **97**(2) 169–182.

Kaymak, Uzay. 2017. On practical applicability of the generalized averaging operator in fuzzy decision making. *Granular, Soft and Fuzzy Approaches for Intelligent Systems: Dedicated to Professor Ronald R. Yager*. Springer International Publishing, Cham, 101–118. doi: 10.1007/978-3-319-40314-4_6. URL `http://dx.doi.org/10.1007/978-3-319-40314-4_6`.

Kek, A.G.H., R.L. Cheu, Q. Meng, C.H. Fung. 2009. A decision support system for vehicle relocation operations in carsharing systems. *Transportation Research Part E: Logistics and Transportation Review* **45**(1) 149–158.

Kim, A., M. Hansen. 2015. Some insights into a sequential resource allocation mechanism for en route air traffic management. *Transportation Research Part B* **79** 1–15.

Klemperer, Paul. 2002. What really matters in auction design. *Journal of Economic Perspectives* **16**(1) 169–189.

Kovalerchuk, B., V. Taliansky. 1992. Comparison of empirical and computed values of fuzzy conjunction. *Fuzzy Sets and Systems* **46** 49–53.

Koziel, Slawomir, David Echeverría Ciaurri, Leifur Leifsson. 2011. Surrogate-based methods. *Computational Optimization, Methods and Algorithms*. Springer, 33–59.

Kubiak, W. 2009. Proportional optimization and fairness, international series in operations research and management science.

Kumar, A., J. Kleinberg. 2000. Fairness measures for resource allocation. *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*. IEEE, 75–85.

Lahdelma, Risto, Pekka Salminen. 2009. Prospect theory and stochastic multicriteria acceptability analysis (smaa). *Omega* **37**(5) 961–971.

Li, Xiaonan, Sigurdur Ólafsson. 2005. Discovering dispatching rules using data mining. *Journal of Scheduling* **8**(6) 515–527.

List, J. A., J. F. Shogren. 1999. Price information and bidding behavior in repeated second-price auctions. *American Journal of Agricultural Economics* **81**(4) 942–949.

Litman, T. 2002. Evaluating transportation equity. *World Transport Policy & Practice* **8**(2) 50–65.

Lovric, M., R. J. Almeida, U. Kaymak, J. Spronk. 2009. Modeling investor optimism with fuzzy connectives. J. P. Carvalho, D. Dubois, U. Kaymak, J. M. C. Sousa, eds., *Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference (IFSA/EUSFLAT 2009)*. Lisbon, Portugal, 1803–1808.

Lulli, G., A. Odoni. 2007. The european air traffic flow management problem. *Transportation Science* **41**(4) 431–443.

Mailath, G. J., L. Samuelson. 2006. *Repeated games and reputations*, vol. 2. Oxford University Press.

Meisel, Stephan, Dirk Mattfeld. 2010. Synergies of operations research and data mining. *European Journal of Operational Research* **206**(1) 1–10.

Meng, Q., H. Yang. 2002. Benefit distribution and equity in road network design. *Transportation Research Part B* **36**(1) 19–35.

Nash, J. F. 1950. The bargaining problem. *Econometrica* 155–162.

Nisan, Noam, Amir Ronen. 1999. Algorithmic mechanism design (extended abstract). *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*. STOC '99, ACM, New York, NY, USA, 129–140. doi: 10.1145/301250.301287. URL `http://doi.acm.org/10.1145/301250.301287`.

Ogryczak, W., H. Luss, M. Pióro, D. Nace, A. Tomaszewski. 2014. Fair optimization and networks: a survey. *Journal of Applied Mathematics* **2014**.

Ogryczak, W., M. Pióro, A. Tomaszewski. 2005. Telecommunications network design and max-min optimization problem. *Journal of Telecommunications and Information Technology* 43–56.

Ogryczak, W., A. Wierzbicki, M. Milewski. 2008. A multi-criteria approach to fair and efficient bandwidth allocation. *Omega* **36**(3) 451 – 463. doi: http://dx.doi.org/10.1016/j.omega.2005.12.005.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau,

M. Brucher, M. Perrot, E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12** 2825–2830.

Perugia, A., L. Moccia, J.-F. Cordeau, G. Laporte. 2011. Designing a home-to-work bus service in a metropolitan area. *Transportation Research Part B* **45**(10) 1710–1726.

Pinker, Edieal J., Abraham Seidmann, Yaniv Vakrat. 2010. Using bid data for the management of sequential, multi-unit, online auctions with uniformly distributed bidder valuations. *European Journal of Operational Research* **202**(2) 574–583.

Pitchik, Carolyn. 2009. Budget-constrained sequential auctions with incomplete information. *Games and Economic Behavior* **66**(2) 928–949.

Puterman, Martin L. 2009. *Markov decision processes: discrete stochastic dynamic programming*, vol. 414. John Wiley & Sons.

Raedt, Luc De, Tias Guns, Siegfried Nijssen. 2010. Constraint programming for data mining and machine learning. *AAAI*. 1671–1675.

Rawls, J. 1971. *A theory of justice*. Harvard University Press.

Rios, Luis Miguel, Nikolaos V Sahinidis. 2013. Derivative-free optimization: A review of algorithms and comparison of software implementations. *Journal of Global Optimization* **56**(3) 1247–1293.

Roth, Dan, Wen-tau Yih. 2005. Integer linear programming inference for conditional random fields. *Proceedings of the 22Nd International Conference on Machine Learning*. ICML '05, ACM, New York, NY, USA, 736–743. doi: 10.1145/1102351.1102444. URL `http://doi.acm.org/10.1145/1102351.1102444`.

Rothkopf, M. H. 1999. Daily repetition: a neglected factor in the analysis of electricity auctions. *The Electricity Journal* **12**(3) 60–70.

Shaheen, S., A. Cohen, E. Martin. 2010. Carsharing parking policy: review of north american practices and san francisco, california, bay area case study. *Transportation Research Record: Journal of the Transportation Research Board* **2187** 146–156.

Shan, Songqing, G Gary Wang. 2010. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization* **41**(2) 219–241.

Sheffi, Y. 2004. Combinatorial auctions in the procurement of transportation services. *Interfaces* **34**(4) 245–252.

Smits, M., R. Janssen. 2008. Impact of electronic auctions on health care markets. *Electronic Markets* **18**(1) 19–29.

Sra, Suvrit, Sebastian Nowozin, Stephen J Wright. 2012. *Optimization for machine learning*. Mit Press.

Subramaniam, Ramanathan, R. Venkatesh. 2009. Optimal bundling strategies in multi-object auctions of complements or substitutes. *Marketing Science* **28** 264–273. doi: 10.1287/mksc.1080.0394.

SURFsara. 2014. Surfsara - the lisa system. URL `https://surfsara.nl/systems/lisa`.

Thole, U., H.-J. Zimmermann, P. Zysno. 1979. On the suitability of minimum and product operators for the intersection of fuzzy sets. *Fuzzy Sets and Systems* **2** 167–180.

Tibshirani, Robert. 1994. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B* **58** 267–288.

Tomaszewski, A. 2005. A polynomial algorithm for solving a general max-min fairness problem. *European Transactions on Telecommunications* **16**(3) 233–240.

Tversky, A., D. Kahneman. 1992. Advances in prospect theory: cumulative representation of uncertainty. *Journal of Risk and Uncertainty* **5**(4) 297–323.

Uney, Fadime, Metin Turkay. 2006. A mixed-integer programming approach to multi-class data classification problem. *European Journal of Operational Research* **173**(3) 910–920.

Van Der Krogt, Roman, Mathijs De Weerdt, Yingqian Zhang. 2008. Of mechanism design and multiagent planning. *Proceedings of the Eighteenth European Conference on Artificial Intelligence*. IOS Press, 423–427.

van Nauta Lemke, Hans R., Jaap G. Dijkman, H. van Haeringen, M. Pleeging. 1983. A characteristic optimism factor in fuzzy decision-making. *Proc. IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*. Marseille, France, 283–288.

Verbeeck, Denny, Francis Maes, Kurt De Grave, Hendrik Blockeel. 2013. Multi-objective optimization with surrogate trees. *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*. ACM, 679–686.

Verwer, Sicco, Yingqian Zhang. 2012. Revenue prediction in budget-constrained sequential auctions with complementarities. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*. 1399–1400.

Verwer, Sicco, Yingqian Zhang, Qing Chuan Ye. 2017. Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence* **244** 368–395.

Vetsikas, Ioannis A. 2013. Sequential auctions with budget-constrained bidders. *IEEE 10th International Conference on e-Business Engineering*. IEEE, 17–24.

Vetsikas, Ioannis A, Nicholas R Jennings. 2010. Sequential auctions with partially substitutable goods. *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*. Springer, 242–258.

Vickrey, William. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance* **16**(1) 8–37.

Weerdt, Mathijs M., Yingqian Zhang, Tomas Klos. 2012. Multiagent task allocation in social networks. *Autonomous Agents and Multi-Agent Systems* **25**(1) 46–86.

Ye, Q. C., Y. Zhang. 2016. Participation behavior and social welfare in repeated task allocations. *IEEE International Conference on Agents (ICA)*. IEEE, 94–97.

Ye, Qing Chuan, Yingqian Zhang, Rommert Dekker. 2017a. Fair task allocation in transportation. *Omega* **68** 1–16.

Ye, Qing Chuan, Yingqian Zhang, Uzay Kaymak. 2017b. Modeling participation behavior in repeated task allocations with fuzzy connectives. *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE, 3219–3234.

Zimmermann, H.-J., P. Zysno. 1980. Latent connectives in human decision making. *Fuzzy Sets and Systems* **4** 37–51.

Zukerman, M., L. Tan, H. Wang, I. Ouveysi. 2005. Efficiency-fairness tradeoff in telecommunications networks. *Communications Letters, IEEE* **9**(7) 643–645.

Nederlandse Samenvatting (Summary in Dutch)

In dit proefschrift bestuderen we twee verschillende aspecten van veilingen en we maken gebruik van technieken en methoden vanuit zowel de besliskunde als de informatica. Ten eerste bestuderen we de toewijzing van taken aan agenten aan het einde van een veiling. Er zijn verschillende manieren om taken toe te wijzen gebaseerd op de biedingen die zijn ingediend door de agenten. Meestal worden de taken toegewezen op een manier die de kosten voor de veilingmeester minimaliseert. In een eenmalige veiling is deze toewijzing optimaal, maar als de veiling meerdere malen wordt uitgevoerd, kan dit gevolgen hebben voor de participatiegraad van agenten en uiteindelijk de kosten voor de veilingmeester. Hierom beschouwen we een gelijke toewijzing, die wat meer kost in een enkele veiling, maar welke een positieve invloed heeft op de participatiegraad van agenten en de uiteindelijke kosten gemaakt door de veilingmeester over alle veilingen. Ten tweede kijken we naar het ontwerp van de veiling. Hoe een veiling werkt, zoals welke taken het eerst aan bod komen, of wat de startprijs is, heeft invloed op de uitkomst. Meestal zijn er experts die verstand van zaken hebben en weten wat er in voorgaande veilingen heeft plaatsgevonden en hoe een toekomstige veiling moet worden ingericht om de beste resultaten te verkrijgen. Voorgaande veilingen bevatten echter zoveel informatie dat zelfs experts soms dingen over het hoofd zien. We gebruiken een combinatie van machinaal leren en optimalisatiemodellen om informatie uit voorgaande veilingen te onttrekken en deze te gebruiken om toekomstige veilingen in te richten voor betere resultaten.

Curriculum Vitae

Charlie (Qing Chuan) Ye (1989) obtained his BSc Econometrics and Operations Research from Erasmus University Rotterdam in 2009. In 2012 he received his MSc Econometrics and Management Science at the same university, with a specialization in Operations Research and Quantitative Logistics.

Charlie joined the Erasmus Research Institute of Management (ERIM) in October 2012 as a PhD student under the supervision of prof.dr.ir. Rommert Dekker and dr. Yingqian Zhang. He worked on multi-objective optimization problems in task allocations and auctions. His work has been published in the journals Artificial Intelligence and Omega. His paper in the journal Omega has received the Best Paper Award for 2017. He has presented his research at various national and international conferences. His research interests include operations research, optimization, algorithm design and machine learning.

During his PhD project he assisted in and taught various courses, primarily numerical methods and programming courses.

Portfolio

Publications

Peer-reviewed journal articles:

Verwer, Sicco, Yingqian Zhang, and **Qing Chuan Ye**. (2017) Auction optimization using regression trees and linear models as integer programs. *Artificial Intelligence* 244: 368-395.

Ye, Qing Chuan, Yingqian Zhang, and Rommert Dekker. (2017) Fair task allocation in transportation. *Omega* 68: 1-16.

Peer-reviewed conference papers:

Ye, Qing Chuan, and Yingqian Zhang. (2016) Participation behavior and social welfare in repeated task allocations. *Agents (ICA), IEEE International Conference on*, pp. 94-97.

Ye, Qing Chuan, Yingqian Zhang, and Uzay Kaymak. (2017) Modeling participation behavior in repeated task allocations with fuzzy connectives. *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pp. 3219-3234.

Working papers:

Ye, Qing Chuan, Yingqian Zhang, Sicco Verwer, Michiel Hilgeman. (2018) Decision Support System for Auction Design using Multi-Objective Optimization of Decision Trees.

Teaching

Lecturer:

Introduction to Programming, Erasmus School of Economics, minor Computer Science / pre-master, 2013, 2016.

Advanced Programming, Erasmus School of Economics, minor Computer Science, 2016.

Numerieke Methoden (Numerical Methods), Erasmus School of Economics, BA Econometrics and Operations Research, 2014-2016.

Tutorial lecturer:

Introduction to Programming, Erasmus School of Economics, minor Computer Science, 2012.

ICT, Erasmus School of Economics, BA Economics and Business Economics, 2013.

Numerieke Methoden (Numerical Methods), Erasmus School of Economics, BA Econometrics and Operations Research, 2013.

Programmeren (Programming), Erasmus School of Economics, BA Econometrics and Operations Research, 2013.

Voortgezet Programmeren (Advanced Programming), Erasmus School of Economics, BA Econometrics and Operations Research, 2013.

PhD courses

SIKS
Advanced Course on Data Mining
Methods and Methodology for IKS
Learning and Reasoning
Information Retrieval
Agent Systems

LNMB
Cooperative Games
OR Games
Robust Optimization
Integer Programming Methods
Noncooperative Games
Algorithmic Game Theory

ERIM
Scientific Integrity
English: CPE level
Publishing Strategy

Summer school
EASSS 2013

Conferences attended

BNAIC 2012, Maastricht, the Netherlands

BENELEARN 2013, Nijmegen, the Netherlands

OR 2013, Rotterdam, the Netherlands

BNAIC 2013, Delft, the Netherlands

LOGMS 2014, Rotterdam, the Netherlands

INFORMS 2014, San Francisco, CA, USA

EURO 2015, Glasgow, Scotland

ICCL 2015, Delft, the Netherlands

INFORMS 2015, Philadelphia, PA, USA

LNMB Conference 2016, Lunteren, the Netherlands

IEEE ICA 2016, Matsue, Japan

Workshop on Data Driven Operations Management 2016, Eindhoven, the Netherlands

LNMB Conference 2017, Lunteren, the Netherlands

IEEE SMC 2017, Banff, Canada

BNAIC 2017, Groningen, the Netherlands

The ERIM PhD Series

The ERIM PhD Series contains PhD dissertations in the field of Research in Management defended at Erasmus University Rotterdam and supervised by senior researchers affiliated to the Erasmus Research Institute of Management (ERIM). All dissertations in the ERIM PhD Series are available in full text through the ERIM Electronic Series Portal: http://repub.eur.nl/pub. ERIM is the joint research institute of the Rotterdam School of Management (RSM) and the Erasmus School of Economics (ESE) at the Erasmus University Rotterdam (EUR).

Dissertations in the last four years

Akemu, O., *Corporate Responses to Social Issues: Essays in Social Entrepreneurship and Corporate Social Responsibility,* Promotors:Prof. G.M. Whiteman & Dr S.P. Kennedy, EPS-2017-392-ORG, https://repub.eur.nl/pub/95768

Alexiou, A., *Management of Emerging Technologies and the Learning Organization: Lessons from the Cloud and Serious Games Technology*, Promotors: Prof. S.J. Magala, Prof. M.C. Schippers and Dr I. Oshri, EPS-2016-404-ORG, http://repub.eur.nl/pub/93818

Alserda, G.A.G., *Choices in Pension Management*, Promotors: Prof. S.G. van der Lecq & Dr O.W. Steenbeek, EPS-2017-432-F&A, https://repub.eur.nl/pub/103496

Arampatzi, E., *Subjective Well-Being in Times of Crises: Evidence on the Wider Impact of Economic Crises and Turmoil on Subjective Well-Being,* Promotors: Prof. H.R. Commandeur, Prof. F. van Oort & Dr. M.J. Burger, EPS-2018-459-S&E, https://repub.eur.nl/pub/111830

Avci, E., *Surveillance of Complex Auction Markets: a Market Policy Analytics Approach*, Promotors: Prof. W. Ketter, Prof. H.W.G.M. van Heck & Prof. D.W. Bunn, EPS-2018-426-LIS, https://repub.eur.nl/pub/106286

Benschop, N, *Biases in Project Escalation:Names, frames & construal levels*, Promotors: Prof. K.I.M. Rhode, Prof. H.R. Commandeur, Prof. M.Keil & Dr A.L.P. Nuijten, EPS-2015-375-S&E, http://repub.eur.nl/pub/79408

Bernoster, I., *Essays at the Intersection of Psychology, Biology, and Entrepreneurship,* Promotors: Prof. A.R. Thurik, Prof. I.H.A. Franken & Prof. P.J.F Groenen, EPS-2018-463-S&E, https://repub.eur.nl/pub/113907

Beusichem, H.C. van, *Firms and Financial Markets:Empirical Studies on the Informational Value of Dividends, Governance and Financial Reporting*, Promotors: Prof. A. de Jong & Dr G. Westerhuis, EPS-2016-378-F&A, http://repub.eur.nl/pub/93079

Bliek, R. de, *Empirical Studies on the Economic Impact of Trust*,
Promotor: Prof.J. Veenman & Prof. Ph.H.B.F. Franses, EPS-2015-324-ORG,
http://repub.eur.nl/pub/78159

Bouman, P., *Passengers,Crowding and Complexity:Models for Passenger Oriented Public Transport*, Prof. L.G. Kroon, Prof. A. Schöbel & Prof. P.H.M. Vervest,
EPS-2017-420-LIS, https://repub.eur.nl/

Brazys, J., *Aggregated Marcoeconomic News and Price Discovery*,
Promotor: Prof.W.F.C. Verschoor, EPS-2015-351-F&A, http://repub.eur.nl/pub/78243

Bunderen, L. van, *Tug-of-War: Why and when teams get embroiled in power struggles*,
Promotors: Prof. D.L. van Knippenberg & Dr. L. Greer, EPS-2018-446-ORG,
https://repub.eur.nl/pub/105346

Burg, G.J.J. van den, *Algorithms for Multiclass Classification and Regularized Regression*,
Promotors: Prof. P.J.F. Groenen & Dr. A. Alfons, EPS-2018-442-MKT,
https://repub.eur.nl/pub/103929

Chammas, G., *Portfolio concentration*, Promotor: Prof. J. Spronk, EPS-2017-410-F&E,
https://repub.eur.nl/pub/94975

Cranenburgh, K.C. van, *Money or Ethics: Multinational corporations and religious organisations operating in an era of corporate responsibility*, Prof. L.C.P.M. Meijs,
Prof. R.J.M. van Tulder &Dr D. Arenas, EPS-2016-385-ORG,
http://repub.eur.nl/pub/93104

Consiglio, I., *Others: Essays on Interpersonal and Consumer Behavior*,
Promotor: Prof. S.M.J. van Osselaer, EPS-2016-366-MKT, http://repub.eur.nl/pub/79820

Darnihamedani, P., *Individual Characteristics, Contextual Factors and Entrepreneurial Behavior*, Promotors: Prof. A.R.Thurik & S.J.A. Hessels, EPS-2016-360-S&E,
http://repub.eur.nl/pub/93280

Dennerlein, T., *Empowering Leadership and Employees' Achievement Motivations: the Role of Self-Efficacy and Goal Orientations in the Empowering Leadership Process*,
Promotors: Prof. D.L. van Knippenberg & Dr J. Dietz, EPS-2017-414-ORG,
https://repub.eur.nl/pub/98438

Deng, W., *Social Capital and Diversification of Cooperatives*,
Promotor: Prof. G.W.J. Hendrikse, EPS-2015-341-ORG, http://repub.eur.nl/pub/77449

Depecik, B.E., *Revitalizing brands and brand: Essays on Brand and Brand Portfolio Management Strategies*, Promotors: Prof. G.H. van Bruggen, Dr Y.M. van Everdingen
and Dr M.B. Ataman, EPS-2016-406-MKT,http://repub.eur.nl/pub/93507

Duijzer, L.E., *Mathematical Optimization in Vaccine Allocation*,
Promotors: Prof. R. Dekker & Dr W.L. van Jaarsveld, EPS-2017-430-LIS,
https://repub.eur.nl/pub/101487

Duyvesteyn, J.G., *Empirical Studies on Sovereign Fixed Income Markets,*
Promotors: Prof. P.Verwijmeren & Prof.M.P.E. Martens, EPS-2015-361-F&A,
https://repub.eur.nl/pub/79033

Elemes, A., *Studies on Determinants and Consequences of Financial Reporting Quality,* Promotor: Prof. E.Peek, EPS-2015-354-F&A,https://repub.eur.nl/pub/79037

Ellen, S. ter, *Measurement, Dynamics, and Implications of Heterogeneous Beliefs in Financial Markets*, Promotor: Prof. W.F.C. Verschoor, EPS-2015-343-F&A,
http://repub.eur.nl/pub/78191

El Nayal, O.S.A.N., *Firms and the State: An Examination of Corporate Political Activity and the Business-Government Interface,* Promotor: Prof. J. van Oosterhout & Dr. M. van Essen, EPS-2018-469-S&E, https://repub.eur.nl/pub/114683

Erlemann, C., *Gender and Leadership Aspiration: The Impact of the Organizational Environment*, Promotor: Prof. D.L. van Knippenberg, EPS-2016-376-ORG,
http://repub.eur.nl/pub/79409

Eskenazi, P.I., *The Accountable Animal*, Promotor: Prof. F.G.H. Hartmann,
EPS-2015-355-F&A, http://repub.eur.nl/pub/78300

Evangelidis, I., *Preference Construction under Prominence*,
Promotor: Prof. S.M.J.van Osselaer, EPS-2015-340-MKT, http://repub.eur.nl/pub/78202

Faber, N., *Structuring Warehouse Management*, Promotors: Prof. M.B.M. de Koster & Prof. A. Smidts, EPS-2015-336-LIS, http://repub.eur.nl/pub/78603

Feng, Y., *The Effectiveness of Corporate Governance Mechanisms and Leadership Structure: Impacts on strategic change and firm performance*,
Promotors: Prof. F.A.J. van den Bosch, Prof. H.W. Volberda & Dr J.S. Sidhu,
EPS-2017-389-S&E, https://repub.eur.nl/pub/98470

Fernald, K., *The Waves of Biotechnological Innovation in Medicine: Interfirm Cooperation Effects and a Venture Capital Perspective,* Promotors: Prof.E.Claassen, Prof. H.P.G. Pennings & Prof. H.R. Commandeur, EPS-2015-371-S&E,
http://hdl.handle.net/1765/79120

Fisch, C.O., *Patents and trademarks: Motivations, antecedents, and value in industrialized and emerging markets*, Promotors: Prof. J.H. Block, Prof. H.P.G. Pennings & Prof. A.R. Thurik, EPS-2016-397-S&E, http://repub.eur.nl/pub/94036

Fliers, P.T., *Essays on Financing and Performance: The role of firms, banks and board*, Promotors: Prof. A. de Jong & Prof. P.G.J. Roosenboom, EPS-2016-388-F&A, http://repub.eur.nl/pub/93019

Frick, T.W., *The Implications of Advertising Personalization for Firms, Consumer, and Ad Platfroms,* Promotors: Prof. T. Li & Prof. H.W.G.M. van Heck, EPS-2018-452-LIS, https://repub.eur.nl/pub/110314

Fytraki, A.T., *Behavioral Effects in Consumer Evaluations of Recommendation Systems*, Promotors: Prof. B.G.C. Dellaert & Prof. T. Li, EPS-2018-427-MKT, https://repub.eur.nl/pub/110457

Gaast, J.P. van der, *Stochastic Models for Order Picking Systems*, Promotors: Prof. M.B.M de Koster & Prof. I.J.B.F. Adan, EPS-2016-398-LIS, http://repub.eur.nl/pub/93222

Ghazizadeh, P., *Empirical Studies on the Role of Financial Information in Asset and Capital Markets,* Promotors: Prof. A. de Jong & Prof. E. Peek. EPS-2019-470-F&A https://repub.eur.nl/pub/114023

Giurge, L., *A Test of Time; A temporal and dynamic approach to power and ethics*, Promotors: Prof. M.H. van Dijke & Prof. D. De Cremer, EPS-2017-412-ORG, https://repub.eur.nl/

Gobena, L., *Towards Integrating Antecedents of Voluntary Tax Compliance*, Promotors: Prof. M.H. van Dijke & Dr P. Verboon, EPS-2017-436-ORG,https://repub.eur.nl/pub/103276

Groot, W.A., *Assessing Asset Pricing Anomalies*, Promotors: Prof. M.J.C.M. Verbeek & Prof. J.H. van Binsbergen, EPS-2017-437-F&A, https://repub.eur.nl/pub/103490

Hanselaar, R.M., *Raising Capital: On pricing, liquidity and incentives,* Promotors: Prof. M.A. van Dijk & Prof. P.G.J. Roosenboom, EPS-2018-429-F&A-9789058925404, https://repub.eur.nl/pub/113274

Harms, J. A., *Essays on the Behavioral Economics of Social Preferences and Bounded Rationality*, Prof. H.R. Commandeur & Dr K.E.H. Maas, EPS-2018-457-S&E, https://repub.eur.nl/pub/108831

Hekimoglu, M., *Spare Parts Management of Aging Capital Products,* Promotor: Prof. R. Dekker, EPS-2015-368-LIS, http://repub.eur.nl/pub/79092

Hendriks, G., *Multinational Enterprises and Limits to International Growth: Links between Domestic and Foreign Activities in a Firm's Portfolio,* Promotors: Prof. P.P.M.A.R. Heugens & Dr. A.H.L Slangen, EPS-2019-464-S&E, https://repub.eur.nl/pub/114981

Hengelaar, G.A., *The Proactive Incumbent: Holy grail or hidden gem?Investigating whether the Dutch electricity sector can overcome the incumbent's curse and lead the sustainability transition*, Promotors: Prof. R.J. M. van Tulder & Dr K. Dittrich, EPS-2018-438-ORG, https://repub.eur.nl/pub/102953

Hogenboom, A.C., *Sentiment Analysis of Text Guided by Semantics and Structure,* Promotors: Prof. U. Kaymak & Prof. F.M.G. de Jong, EPS-2015-369-LIS, http://repub.eur.nl/pub/79034

Hollen, R.M.A*., Exploratory Studies into Strategies to Enhance Innovation-Driven International Competitiveness in a Port Context: Toward Ambidextrous Ports,* Promotors: Prof. F.A.J. Van Den Bosch & Prof. H.W. Volberda, EPS-2015-372-S&E, http://repub.eur.nl/pub/78881

Hurk, E. van der, *Passengers, Information, and Disruptions*, Promotors: Prof. L.G.Kroon & Prof. P.H.M. Vervest, EPS-2015-345-LIS, http://repub.eur.nl/pub/78275

Jacobs, B.J.D., *Marketing Analytics for High-Dimensional Assortments*, Promotors: Prof. A.C.D. Donkers & Prof. D. Fok, EPS-2017-445-MKT, https://repub.eur.nl/pub/103497

Kahlen, M. T., *Virtual Power Plants of Electric Vehicles in Sustainable Smart Electricity Markets*, Promotors: Prof. W. Ketter & Prof. A. Gupta, EPS-2017-431-LIS, https://repub.eur.nl/pub/100844

Kampen, S. van, *The Cross-sectional and Time-series Dynamics of Corporate Finance: Empirical evidence from financially constrained firms*, Promotors: Prof. L. Norden & Prof. P.G.J. Roosenboom, EPS-2018-440-F&A, https://repub.eur.nl/pub/105245

Karali, E., *Investigating Routines and Dynamic Capabilities for Change and Innovation,* Promotors: Prof. H.W. Volberda, Prof. H.R. Commandeur and Dr J.S. Sidhu, EPS-2018-454-S&E, https://repub.eur.nl/pub/106274

Keko. E, *Essays on Innovation Generation in Incumbent Firms*, Promotors: Prof. S. Stremersch & Dr N.M.A. Camacho, EPS-2017-419-MKT, https://repub.eur.nl/pub/100841

Kerkkamp, R.B.O., *Optimisation Models for Supply Chain Coordination under Information Asymmetry,* Promotors: Prof. A.P.M. Wagelmans & Dr. W. van den Heuvel, EPS-2018-462-LIS

Khattab, J., *Make Minorities Great Again: a contribution to workplace equity by identifying and addressing constraints and privileges*, Promotors: Prof. D.L. van Knippenberg & Dr A. Nederveen Pieterse, EPS-2017-421-ORG, https://repub.eur.nl/pub/99311

Kim, T. Y., *Data-driven Warehouse Management in Global Supply Chains*, Promotors: Prof. R. Dekker & Dr C. Heij, EPS-2018-449-LIS, https://repub.eur.nl/pub/109103

Klitsie, E.J., *Strategic Renewal in Institutional Contexts: The paradox of embedded agency*, Promotors: Prof. H.W. Volberda & Dr. S. Ansari, EPS-2018-444-S&E, https://repub.eur.nl/pub/106275

Kong, L. *Essays on Financial Coordination,* Promotors: Prof. M.J.C.M. Verbeek, Dr. D.G.J. Bongaerts & Dr. M.A. van Achter. EPS-2019-433-F&A, https://repub.eur.nl/pub/114516

Krämer, R., *A license to mine? Community organizing against multinational corporations*, Promotors: Prof. R.J.M. van Tulder & Prof. G.M. Whiteman, EPS-2016-383-ORG, http://repub.eur.nl/pub/94072

Kysucky, V., *Access to Finance in a Cros-Country Context*, Promotor: Prof. L.Norden, EPS-2015-350-F&A, http://repub.eur.nl/pub/78225

Lee, C.I.S.G., *Big Data in Management Research: Exploring New Avenues*, Promotors: Prof. S.J. Magala & Dr W.A. Felps, EPS-2016-365-ORG, http://repub.eur.nl/pub/79818

Legault-Tremblay, P.O., *Corporate Governance During Market Transition: Heterogeneous responses to Institution Tensions in China*, Promotor: Prof. B. Krug, EPS-2015-362-ORG, http://repub.eur.nl/pub/78649

Lenoir, A.S., *Are You Talking to Me? Addressing Consumers in a Globalised World,* Promotors: Prof. S. Puntoni & Prof. S.M.J. van Osselaer, EPS-2015-363-MKT, http://repub.eur.nl/pub/79036

Li, D., *Supply Chain Contracting for After-sales Service and Product Support*, Promotor: Prof. M.B.M. de Koster, EPS-2015-347-LIS, http://repub.eur.nl/pub/78526

Li, X., *Dynamic Decision Making under Supply Chain Competition,* Promotors: Prof. M.B.M de Koster, Prof. R. Dekker & Prof. R. Zuidwijk. EPS-2018-466-LIS, https://repub.eur.nl/pub/114028

Liu, N., *Behavioral Biases in Interpersonal Contexts*, Supervisors: Prof. A. Baillon & Prof. H. Bleichrodt, EPS-2017-408-MKT, https://repub.eur.nl/pub/95487

Ma, Y., *The Use of Advanced Transportation Monitoring Data for Official Statistics,* Promotors: Prof. L.G.Kroon& Dr J. van Dalen, EPS-2016-391-LIS, http://repub.eur.nl/pub/80174

Maira, E., *Consumers and Producers*, Promotors: Prof. S. Puntoni &Prof. C. Fuchs, EPS-2018-439-MKT, https://repub.eur.nl/pub/104387

Mell, J.N., *Connecting Minds: On The Role of Metaknowledge in Knowledge Coordination*, Promotor: Prof. D.L. van Knippenberg, EPS-2015-359-ORG, http://hdl.handle.net/1765/78951

Meulen,van der, D., *The Distance Dilemma: the effect of flexible working practices on performance in the digital workplace*, Promotors: Prof. H.W.G.M. van Heck & Prof. P.J. van Baalen, EPS-2016-403-LIS,http://repub.eur.nl/pub/94033

Micheli, M.R., *Business Model Innovation: A Journey across Managers' Attention and Inter-Organizational Networks*, Promotor: Prof. J.J.P. Jansen, EPS-2015-344-S&E, http://repub.eur.nl/pub/78241

Moniz, A, *Textual Analysis of Intangible Information*, Promotors: Prof. C.B.M. van Riel, Prof. F.M.G de Jong & Dr G.A.J.M. Berens, EPS-2016-393-ORG, http://repub.eur.nl/pub/93001

Mulder, J., *Network design and robust scheduling in liner shipping*, Promotors: Prof. R. Dekker & Dr W.L. van Jaarsveld, EPS-2016-384-LIS, http://repub.eur.nl/pub/80258

Neerijnen, P., *The Adaptive Organization: the socio-cognitive antecedents of ambidexterity and individual exploration*, Promotors: Prof. J.J.P. Jansen, P.P.M.A.R. Heugens & Dr T.J.M. Mom, EPS-2016-358-S&E, http://repub.eur.nl/pub/93274

Okbay, A., *Essays on Genetics and the Social Sciences*, Promotors: Prof. A.R. Thurik, Prof. Ph.D. Koellinger & Prof. P.J.F. Groenen, EPS-2017-413-S&E, https://repub.eur.nl/pub/95489

Oord, J.A. van, *Essays on Momentum Strategies in Finance,* Promotor: Prof. H.K. van Dijk, EPS-2016-380-F&A, http://repub.eur.nl/pub/80036

Peng, X., *Innovation, Member Sorting, and Evaluation of Agricultural Cooperatives*, Promotor: Prof. G.W.J. Hendriks, EPS-2017-409-ORG, https://repub.eur.nl/pub/94976

Pennings, C.L.P., *Advancements in Demand Forecasting: Methods and Behavior*, Promotors: Prof. L.G. Kroon, Prof. H.W.G.M. van Heck & Dr J. van Dalen, EPS-2016-400-LIS, http://repub.eur.nl/pub/94039

Petruchenya, A., *Essays on Cooperatives: Emergence, Retained Earnings, and Market Shares*, Promotors: Prof. G.W.J. Hendriks & Dr Y. Zhang, EPS-2018-447-ORG, https://repub.eur.nl/pub/105243

Plessis, C. du, *Influencers:The Role of Social Influence in Marketing*, Promotors: Prof. S. Puntoni & Prof. S.T.L.R. Sweldens, EPS-2017-425-MKT, https://repub.eur.nl/pub/103265

Pocock, M., *Status Inequalities in Business Exchange Relations in Luxury Markets*, Promotors: Prof. C.B.M. van Riel & Dr G.A.J.M. Berens, EPS-2017-346-ORG, https://repub.eur.nl/pub/98647

Pozharliev, R., *Social Neuromarketing: The role of social context in measuring advertising effectiveness*, Promotors: Prof. W.J.M.I. Verbeke & Prof. J.W. van Strien, EPS-2017-402-MKT, https://repub.eur.nl/pub/95528

Protzner, S., *Mind the gap between demand and supply: A behavioral perspective on demand forecasting,* Promotors: Prof. S.L. van de Velde &Dr L. Rook, EPS-2015-364-LIS, *http*://repub.eur.nl/pub/79355

Pruijssers, J.K., *An Organizational Perspective on Auditor Conduct*, Promotors: Prof. J. van Oosterhout & Prof. P.P.M.A.R. Heugens, EPS-2015-342-S&E, http://repub.eur.nl/pub/78192

Reh, S.G., *A Temporal Perspective on Social Comparisons in Organizations,* Promotors: Prof. S.R. Giessner, Prof. N. van Quaquebeke & Dr. C. Troster, EPS-2018-471-ORG, https://repub.eur.nl/pub/114522

Riessen, B. van, *Optimal Transportation Plans and Portfolios for Synchromodal Container Networks*, Promotors: Prof. R. Dekker & Prof. R.R. Negenborn, EPS-2018-448-LIS, https://repub.eur.nl/pub/105248

Rietdijk, W.J.R., *The Use of Cognitive Factors for Explaining Entrepreneurship,* Promotors: Prof. A.R. Thurik & Prof. I.H.A. Franken, EPS-2015-356-S&E, http://repub.eur.nl/pub/79817

Rösch, D., *Market Efficiency and Liquidity,* Promotor: Prof. M.A. van Dijk, EPS-2015-353-F&A, http://repub.eur.nl/pub/79121

Roza, L., *Employee Engagement in Corporate Social Responsibility: A collection of essays*, Promotor: Prof. L.C.P.M. Meijs, EPS-2016-396-ORG, http://repub.eur.nl/pub/93254

Schie, R. J. G. van, *Planning for Retirement: Save More or Retire Later?* Promotors: Prof. B. G. C. Dellaert & Prof. A.C.D. Donkers, EOS-2017-415-MKT, https://repub.eur.nl/pub/100846

Schoonees, P., *Methods for Modelling Response Styles*, Promotor: Prof. P.J.F. Groenen, EPS-2015-348-MKT, http://repub.eur.nl/pub/79327

Schouten, K.I.M., *Semantics-driven Aspect-based Sentiment Analysis,* Promotors: Prof. F.M.G. de Jong, Prof. R. Dekker & Dr. F. Frasincar, EPS-2018-453-LIS, https://repub.eur.nl/pub/112161

Schouten, M.E., *The Ups and Downs of Hierarchy: the causes and consequences of hierarchy struggles and positional loss*, Promotors; Prof. D.L. van Knippenberg & Dr L.L. Greer, EPS-2016-386-ORG, http://repub.eur.nl/pub/80059

Smit, J., *Unlocking Business Model Innovation: A look through the keyhole at the inner workings of Business Model Innovation*, Promotor: Prof. H.G. Barkema, EPS-2016-399-S&E, http://repub.eur.nl/pub/93211

Straeter, L.M., *Interpersonal Consumer Decision Making*, Promotors: Prof. S.M.J. van Osselaer & Dr I.E. de Hooge, EPS-2017-423-MKT, https://repub.eur.nl/pub/100819

Stuppy, A., *Essays on Product Quality,* Promotors: Prof. S.M.J. van Osselaer & Dr. N.L. Mead. EPS-2018-461-MKT, https://repub.eur.nl/pub/111375

Subaşi, B., *Demographic Dissimilarity, Information Access and Individual Performance*, Promotors: Prof. D.L. van Knippenberg & Dr W.P. van Ginkel, EPS-2017-422-ORG, https://repub.eur.nl/pub/103495

Suurmond, R., *In Pursuit of Supplier Knowledge: Leveraging capabilities and dividing responsibilities in product and service contexts,* Promotors: Prof. J.Y.F Wynstra & Prof. J. Dul. EPS-2018-475-LIS, https://repub.eur.nl/pub/115138

Szatmari, B., *We are (all) the champions: The effect of status in the implementation of innovations*, Promotors: Prof. J.C.M van den Ende & Dr D. Deichmann, EPS-2016-401-LIS, http://repub.eur.nl/pub/94633

Toxopeus, H.S., *Financing sustainable innovation: From a principal-agent to a collective action perspective,* Promotors: Prof. H.R. Commandeur & Dr. K.E.H. Maas. EPS-2019-458-S&E, https://repub.eur.nl/pub/114018

Tuijl, E. van, *Upgrading across Organisational and Geographical Configurations*, Promotor: Prof. L. van den Berg, EPS-2015-349-S&E, http://repub.eur.nl/pub/78224

Turturea, R., *Overcoming Resource Constraints: The Role of Creative Resourcing and Equity Crowdfunding in Financing Entrepreneurial Ventures,* Promotors: Prof. P.P.M.A.R Heugens, Prof. J.J.P. Jansen & Dr. I. Verheuil, EPS-2019-472-S&E, https://repub.eur.nl/pub/112859

Valogianni, K., *Sustainable Electric Vehicle Management using Coordinated Machine Learning*, Promotors: Prof. H.W.G.M. van Heck & Prof. W. Ketter, EPS-2016-387-LIS, http://repub.eur.nl/pub/93018

Vandic, D., *Intelligent Information Systems for Web Product Search*, Promotors: Prof. U. Kaymak & Dr Frasincar, EPS-2017-405-LIS, https://repub.eur.nl/pub/95490

Verbeek, R.W.M., *Essays on Empirical Asset Pricing*, Promotors: Prof. M.A. van Dijk & Dr M. Szymanowska, EPS-2017-441-F&A, https://repub.eur.nl/pub/102977

Vermeer, W., *Propagation in Networks:The impact of information processing at the actor level on system-wide propagation dynamics,* Promotor: Prof. P.H.M.Vervest, EPS-2015-373-LIS, http://repub.eur.nl/pub/79325

Versluis, I., *Prevention of the Portion Size Effect*, Promotors: Prof. Ph.H.B.F. Franses & Dr E.K. Papies, EPS-2016-382-MKT, http://repub.eur.nl/pub/79880

Vishwanathan, P., *Governing for Stakeholders: How Organizations May Create or Destroy Value for their Stakeholders*, Promotors: Prof. J. van Oosterhout &Prof. L.C.P.M. Meijs, EPS-2016-377-ORG, http://repub.eur.nl/pub/93016

Vlaming, R. de., *Linear Mixed Models in Statistical Genetics*, Prof. A.R. Thurik, Prof. P.J.F. Groenen & Prof. Ph.D. Koellinger, EPS-2017-416-S&E,https://repub.eur.nl/pub/100428

Vries, H. de, *Evidence-Based Optimization in Humanitarian Logistics*, Promotors: Prof. A.P.M. Wagelmans & Prof. J.J. van de Klundert, EPS-2017-435-LIS, https://repub.eur.nl/pub/102771

Vries, J. de, *Behavioral Operations in Logistics,* Promotors: Prof. M.B.M de Koster & Prof. D.A. Stam, EPS-2015-374-LIS, http://repub.eur.nl/pub/79705

Wagenaar, J.C., *Practice Oriented Algorithmic Disruption Management in Passenger Railways*, Prof. L.G. Kroon & Prof. A.P.M. Wagelmans, EPS-2016-390-LIS, http://repub.eur.nl/pub/93177

Wang, P., *Innovations, status, and networks*, Promotors: Prof. J.J.P. Jansen & Dr V.J.A. van de Vrande, EPS-2016-381-S&E, http://repub.eur.nl/pub/93176

Wang, R., *Corporate Environmentalism in China*, Promotors: Prof. P.P.M.A.R Heugens & Dr F. Wijen, EPS-2017-417-S&E, https://repub.eur.nl/pub/99987

Wang, T., *Essays in Banking and Corporate Finance*, Promotors: Prof. L. Norden & Prof. P.G.J. Roosenboom, EPS-2015-352-F&A, http://repub.eur.nl/pub/78301

Wasesa, M., *Agent-based inter-organizational systems in advanced logistics operations*, Promotors: Prof. H.W.G.M van Heck, Prof. R.A. Zuidwijk & Dr A. W. Stam, EPS-2017-LIS-424, https://repub.eur.nl/pub/100527

Wessels, C., *Flexible Working Practices: How Employees Can Reap the Benefits for Engagement and Performance*, Promotors: Prof. H.W.G.M. van Heck, Prof. P.J. van Baalen & Prof. M.C. Schippers, EPS-2017-418-LIS, https://repub.eur.nl/

Wiegmann, P.M., *Setting the Stage for Innovation: Balancing Diverse Interests through Standardisation,* Promotors: Prof. H.J. de Vries & Dr. K. Blind, EPS-2019-473-LIS, https://repub.eur.nl/pub/114519

Williams, A.N., *Make Our Planet Great Again: A Systems Perspective of Corporate Sustainability,* Promotors: Prof. G.M. Whiteman & Dr. S. Kennedy, EPS-2018-456-ORG, https://repub.eur.nl/pub/111032

Witte, C.T., *Bloody Business: Multinational investment in an increasingly conflict-afflicted world*, Promotors: Prof. H.P.G. Pennings, Prof. H.R. Commandeur & Dr M.J. Burger, EPS-2018-443-S&E, https://repub.eur.nl/pub/104027

Yuan, Y., *The Emergence of Team Creativity: a social network perspective*, Promotors: Prof. D. L. van Knippenberg & Dr D. A. Stam, EPS-2017-434-ORG, https://repub.eur.nl/pub/100847

Ypsilantis, P., *The Design, Planning and Execution of Sustainable Intermodal Port-hinterland Transport Networks*, Promotors: Prof. R.A. Zuidwijk & Prof. L.G. Kroon, EPS-2016-395-LIS, http://repub.eur.nl/pub/94375

Yuferova, D., *Price Discovery, Liquidity Provision, and Low-Latency Trading*, Promotors: Prof. M.A. van Dijk & Dr D.G.J. Bongaerts, EPS-2016-379-F&A, http://repub.eur.nl/pub/93017

Zhang, Q., *Financing and Regulatory Frictions in Mergers and Acquisitions*, Promotors: Prof. P.G.J. Roosenboom & Prof. A. de Jong, EPS-2018-428-F&A, https://repub.eur.nl/pub/103871

Zuber, F.B.*, Looking at the Others: Studies on (un)ethical behavior and social relationships in organizations*, Promotor: Prof. S.P. Kaptein, EPS-2016-394-ORG, http://repub.eur.nl/pub/94388

SIKS Dissertation Series

2011 01 Botond Cseke (RUN), Variational Algorithms for Bayesian Inference in Latent Gaussian Models

02 Nick Tinnemeier (UU), Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language

03 Jan Martijn van der Werf (TUE), Compositional Design and Verification of Component-Based Information Systems

04 Hado van Hasselt (UU), Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference

05 Bas van der Raadt (VU), Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.

06 Yiwen Wang (TUE), Semantically-Enhanced Recommendations in Cultural Heritage

07 Yujia Cao (UT), Multimodal Information Presentation for High Load Human Computer Interaction

08 Nieske Vergunst (UU), BDI-based Generation of Robust Task-Oriented Dialogues

09 Tim de Jong (OU), Contextualised Mobile Media for Learning

10 Bart Bogaert (UvT), Cloud Content Contention

11 Dhaval Vyas (UT), Designing for Awareness: An Experience-focused HCI Perspective

12 Carmen Bratosin (TUE), Grid Architecture for Distributed Process Mining

13 Xiaoyu Mao (UvT), Airport under Control. Multiagent Scheduling for Airport Ground Handling

14 Milan Lovric (EUR), Behavioral Finance and Agent-Based Artificial Markets

15 Marijn Koolen (UvA), The Meaning of Structure: the Value of Link Evidence for Information Retrieval

16 Maarten Schadd (UM), Selective Search in Games of Different Complexity

17 Jiyin He (UVA), Exploring Topic Structure: Coherence, Diversity and Relatedness

18 Mark Ponsen (UM), Strategic Decision-Making in complex games

19 Ellen Rusman (OU), The Mind's Eye on Personal Profiles

20 Qing Gu (VU), Guiding service-oriented software engineering - A view-based approach

21 Linda Terlouw (TUD), Modularization and Specification of Service-Oriented Systems

22 Junte Zhang (UVA), System Evaluation of Archival Description and Access

23 Wouter Weerkamp (UVA), Finding People and their Utterances in Social Media

24 Herwin van Welbergen (UT), Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior

25 Syed Waqar ul Qounain Jaffry (VU), Analysis and Validation of Models for Trust Dynamics

26 Matthijs Aart Pontier (VU), Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots

27 Aniel Bhulai (VU), Dynamic website optimization through autonomous management of design patterns

28 Rianne Kaptein (UVA), Effective Focused Retrieval by Exploiting Query Context and Document Structure

29 Faisal Kamiran (TUE), Discrimination-aware Classification

30 Egon van den Broek (UT), Affective Signal Processing (ASP): Unraveling the mystery of emotions

31 Ludo Waltman (EUR), Computational and Game-Theoretic Approaches for Modeling Bounded Rationality

32 Nees-Jan van Eck (EUR), Methodological Advances in Bibliometric Mapping of Science

33 Tom van der Weide (UU), Arguing to Motivate Decisions

34 Paolo Turrini (UU), Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations

35 Maaike Harbers (UU), Explaining Agent Behavior in Virtual Training

36 Erik van der Spek (UU), Experiments in serious game design: a cognitive approach
37 Adriana Burlutiu (RUN), Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference
38 Nyree Lemmens (UM), Bee-inspired Distributed Optimization
39 Joost Westra (UU), Organizing Adaptation using Agents in Serious Games
40 Viktor Clerc (VU), Architectural Knowledge Management in Global Software Development
41 Luan Ibraimi (UT), Cryptographically Enforced Distributed Data Access Control
42 Michal Sindlar (UU), Explaining Behavior through Mental State Attribution
43 Henk van der Schuur (UU), Process Improvement through Software Operation Knowledge
44 Boris Reuderink (UT), Robust Brain-Computer Interfaces
45 Herman Stehouwer (UvT), Statistical Language Models for Alternative Sequence Selection
46 Beibei Hu (TUD), Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work
47 Azizi Bin Ab Aziz (VU), Exploring Computational Models for Intelligent Support of Persons with Depression
48 Mark Ter Maat (UT), Response Selection and Turn-taking for a Sensitive Artificial Listening Agent
49 Andreea Niculescu (UT), Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality

---

2012 01 Terry Kakeeto (UvT), Relationship Marketing for SMEs in Uganda
02 Muhammad Umair (VU), Adaptivity, emotion, and Rationality in Human and Ambient Agent Models
03 Adam Vanya (VU), Supporting Architecture Evolution by Mining Software Repositories
04 Jurriaan Souer (UU), Development of Content Management System-based Web Applications
05 Marijn Plomp (UU), Maturing Interorganisational Information Systems
06 Wolfgang Reinhardt (OU), Awareness Support for Knowledge Workers in Research Networks
07 Rianne van Lambalgen (VU), When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions
08 Gerben de Vries (UVA), Kernel Methods for Vessel Trajectories
09 Ricardo Neisse (UT), Trust and Privacy Management Support for Context-Aware Service Platforms
10 David Smits (TUE), Towards a Generic Distributed Adaptive Hypermedia Environment
11 J.C.B. Rantham Prabhakara (TUE), Process Mining in the Large: Preprocessing, Discovery, and Diagnostics
12 Kees van der Sluijs (TUE), Model Driven Design and Data Integration in Semantic Web Information Systems
13 Suleman Shahid (UvT), Fun and Face: Exploring non-verbal expressions of emotion during playful interactions
14 Evgeny Knutov (TUE), Generic Adaptation Framework for Unifying Adaptive Web-based Systems
15 Natalie van der Wal (VU), Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.
16 Fiemke Both (VU), Helping people by understanding them - Ambient Agents supporting task execution and depression treatment
17 Amal Elgammal (UvT), Towards a Comprehensive Framework for Business Process Compliance
18 Eltjo Poort (VU), Improving Solution Architecting Practices
19 Helen Schonenberg (TUE), What's Next? Operational Support for Business Process Execution
20 Ali Bahramisharif (RUN), Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing
21 Roberto Cornacchia (TUD), Querying Sparse Matrices for Information Retrieval

22 Thijs Vis (UvT), Intelligence, politie en veiligheidsdienst: verenigbare grootheden?
23 Christian Muehl (UT), Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction
24 Laurens van der Werff (UT), Evaluation of Noisy Transcripts for Spoken Document Retrieval
25 Silja Eckartz (UT), Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application
26 Emile de Maat (UVA), Making Sense of Legal Text
27 Hayrettin Gurkok (UT), Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games
28 Nancy Pascall (UvT), Engendering Technology Empowering Women
29 Almer Tigelaar (UT), Peer-to-Peer Information Retrieval
30 Alina Pommeranz (TUD), Designing Human-Centered Systems for Reflective Decision Making
31 Emily Bagarukayo (RUN), A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure
32 Wietske Visser (TUD), Qualitative multi-criteria preference representation and reasoning
33 Rory Sie (OUN), Coalitions in Cooperation Networks (COCOON)
34 Pavol Jancura (RUN), Evolutionary analysis in PPI networks and applications
35 Evert Haasdijk (VU), Never Too Old To Learn – On-line Evolution of Controllers in Swarm- and Modular Robotics
36 Denis Ssebugwawo (RUN), Analysis and Evaluation of Collaborative Modeling Processes
37 Agnes Nakakawa (RUN), A Collaboration Process for Enterprise Architecture Creation
38 Selmar Smit (VU), Parameter Tuning and Scientific Testing in Evolutionary Algorithms
39 Hassan Fatemi (UT), Risk-aware design of value and coordination networks
40 Agus Gunawan (UvT), Information Access for SMEs in Indonesia
41 Sebastian Kelle (OU), Game Design Patterns for Learning
42 Dominique Verpoorten (OU), Reflection Amplifiers in self-regulated Learning
43 Withdrawn
44 Anna Tordai (VU), On Combining Alignment Techniques
45 Benedikt Kratz (UvT), A Model and Language for Business-aware Transactions
46 Simon Carter (UVA), Exploration and Exploitation of Multilingual Data for Statistical Machine Translation
47 Manos Tsagkias (UVA), Mining Social Media: Tracking Content and Predicting Behavior
48 Jorn Bakker (TUE), Handling Abrupt Changes in Evolving Time-series Data
49 Michael Kaisers (UM), Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions
50 Steven van Kervel (TUD), Ontologogy driven Enterprise Information Systems Engineering
51 Jeroen de Jong (TUD), Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching

---

2013 01 Viorel Milea (EUR), News Analytics for Financial Decision Support
02 Erietta Liarou (CWI), MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing
03 Szymon Klarman (VU), Reasoning with Contexts in Description Logics
04 Chetan Yadati (TUD), Coordinating autonomous planning and scheduling
05 Dulce Pumareja (UT), Groupware Requirements Evolutions Patterns
06 Romulo Goncalves (CWI), The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience
07 Giel van Lankveld (UvT), Quantifying Individual Player Differences
08 Robbert-Jan Merk (VU), Making enemies: cognitive modeling for opponent agents in fighter pilot simulators
09 Fabio Gori (RUN), Metagenomic Data Analysis: Computational Methods and Applications

10 Jeewanie Jayasinghe Arachchige (UvT), A Unified Modeling Framework for Service Design.
11 Evangelos Pournaras (TUD), Multi-level Reconfigurable Self-organization in Overlay Services
12 Marian Razavian (VU), Knowledge-driven Migration to Services
13 Mohammad Safiri (UT), Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly
14 Jafar Tanha (UVA), Ensemble Approaches to Semi-Supervised Learning Learning
15 Daniel Hennes (UM), Multiagent Learning - Dynamic Games and Applications
16 Eric Kok (UU), Exploring the practical benefits of argumentation in multi-agent deliberation
17 Koen Kok (VU), The PowerMatcher: Smart Coordination for the Smart Electricity Grid
18 Jeroen Janssens (UvT), Outlier Selection and One-Class Classification
19 Renze Steenhuizen (TUD), Coordinated Multi-Agent Planning and Scheduling
20 Katja Hofmann (UvA), Fast and Reliable Online Learning to Rank for Information Retrieval
21 Sander Wubben (UvT), Text-to-text generation by monolingual machine translation
22 Tom Claassen (RUN), Causal Discovery and Logic
23 Patricio de Alencar Silva (UvT), Value Activity Monitoring
24 Haitham Bou Ammar (UM), Automated Transfer in Reinforcement Learning
25 Agnieszka Anna Latoszek-Berendsen (UM), Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System
26 Alireza Zarghami (UT), Architectural Support for Dynamic Homecare Service Provisioning
27 Mohammad Huq (UT), Inference-based Framework Managing Data Provenance
28 Frans van der Sluis (UT), When Complexity becomes Interesting: An Inquiry into the Information eXperience
29 Iwan de Kok (UT), Listening Heads
30 Joyce Nakatumba (TUE), Resource-Aware Business Process Management: Analysis and Support
31 Dinh Khoa Nguyen (UvT), Blueprint Model and Language for Engineering Cloud Applications
32 Kamakshi Rajagopal (OUN), Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development
33 Qi Gao (TUD), User Modeling and Personalization in the Microblogging Sphere
34 Kien Tjin-Kam-Jet (UT), Distributed Deep Web Search
35 Abdallah El Ali (UvA), Minimal Mobile Human Computer Interaction
36 Than Lam Hoang (TUe), Pattern Mining in Data Streams
37 Dirk Börner (OUN), Ambient Learning Displays
38 Eelco den Heijer (VU), Autonomous Evolutionary Art
39 Joop de Jong (TUD), A Method for Enterprise Ontology based Design of Enterprise Information Systems
40 Pim Nijssen (UM), Monte-Carlo Tree Search for Multi-Player Games
41 Jochem Liem (UVA), Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning
42 Léon Planken (TUD), Algorithms for Simple Temporal Reasoning
43 Marc Bron (UVA), Exploration and Contextualization through Interaction and Concepts

---

2014 01 Nicola Barile (UU), Studies in Learning Monotone Models from Data
02 Fiona Tuliyano (RUN), Combining System Dynamics with a Domain Modeling Method
03 Sergio Raul Duarte Torres (UT), Information Retrieval for Children: Search Behavior and Solutions
04 Hanna Jochmann-Mannak (UT), Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation
05 Jurriaan van Reijsen (UU), Knowledge Perspectives on Advancing Dynamic Capability
06 Damian Tamburri (VU), Supporting Networked Software Development
07 Arya Adriansyah (TUE), Aligning Observed and Modeled Behavior

08 Samur Araujo (TUD), Data Integration over Distributed and Heterogeneous Data Endpoints
09 Philip Jackson (UvT), Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language
10 Ivan Salvador Razo Zapata (VU), Service Value Networks
11 Janneke van der Zwaan (TUD), An Empathic Virtual Buddy for Social Support
12 Willem van Willigen (VU), Look Ma, No Hands: Aspects of Autonomous Vehicle Control
13 Arlette van Wissen (VU), Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains
14 Yangyang Shi (TUD), Language Models With Meta-information
15 Natalya Mogles (VU), Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare
16 Krystyna Milian (VU), Supporting trial recruitment and design by automatically interpreting eligibility criteria
17 Kathrin Dentler (VU), Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability
18 Mattijs Ghijsen (UVA), Methods and Models for the Design and Study of Dynamic Agent Organizations
19 Vinicius Ramos (TUE), Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support
20 Mena Habib (UT), Named Entity Extraction and Disambiguation for Informal Text: The Missing Link
21 Kassidy Clark (TUD), Negotiation and Monitoring in Open Environments
22 Marieke Peeters (UU), Personalized Educational Games - Developing agent-supported scenario-based training
23 Eleftherios Sidirourgos (UvA/CWI), Space Efficient Indexes for the Big Data Era
24 Davide Ceolin (VU), Trusting Semi-structured Web Data
25 Martijn Lappenschaar (RUN), New network models for the analysis of disease interaction
26 Tim Baarslag (TUD), What to Bid and When to Stop
27 Rui Jorge Almeida (EUR), Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty
28 Anna Chmielowiec (VU), Decentralized k-Clique Matching
29 Jaap Kabbedijk (UU), Variability in Multi-Tenant Enterprise Software
30 Peter de Cock (UvT), Anticipating Criminal Behaviour
31 Leo van Moergestel (UU), Agent Technology in Agile Multiparallel Manufacturing and Product Support
32 Naser Ayat (UvA), On Entity Resolution in Probabilistic Data
33 Tesfa Tegegne (RUN), Service Discovery in eHealth
34 Christina Manteli (VU), The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.
35 Joost van Ooijen (UU), Cognitive Agents in Virtual Worlds: A Middleware Design Approach
36 Joos Buijs (TUE), Flexible Evolutionary Algorithms for Mining Structured Process Models
37 Maral Dadvar (UT), Experts and Machines United Against Cyberbullying
38 Danny Plass-Oude Bos (UT), Making brain-computer interfaces better: improving usability through post-processing.
39 Jasmina Maric (UvT), Web Communities, Immigration, and Social Capital
40 Walter Omona (RUN), A Framework for Knowledge Management Using ICT in Higher Education
41 Frederic Hogenboom (EUR), Automated Detection of Financial Events in News Text
42 Carsten Eijckhof (CWI/TUD), Contextual Multidimensional Relevance Models
43 Kevin Vlaanderen (UU), Supporting Process Improvement using Method Increments

44 Paulien Meesters (UvT), Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.
45 Birgit Schmitz (OUN), Mobile Games for Learning: A Pattern-Based Approach
46 Ke Tao (TUD), Social Web Data Analytics: Relevance, Redundancy, Diversity
47 Shangsong Liang (UVA), Fusion and Diversification in Information Retrieval

2015 01 Niels Netten (UvA), Machine Learning for Relevance of Information in Crisis Response
02 Faiza Bukhsh (UvT), Smart auditing: Innovative Compliance Checking in Customs Controls
03 Twan van Laarhoven (RUN), Machine learning for network data
04 Howard Spoelstra (OUN), Collaborations in Open Learning Environments
05 Christoph Bösch (UT), Cryptographically Enforced Search Pattern Hiding
06 Farideh Heidari (TUD), Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes
07 Maria-Hendrike Peetz (UvA), Time-Aware Online Reputation Analysis
08 Jie Jiang (TUD), Organizational Compliance: An agent-based model for designing and evaluating organizational interactions
09 Randy Klaassen (UT), HCI Perspectives on Behavior Change Support Systems
10 Henry Hermans (OUN), OpenU: design of an integrated system to support lifelong learning
11 Yongming Luo (TUE), Designing algorithms for big graph datasets: A study of computing bisimulation and joins
12 Julie M. Birkholz (VU), Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks
13 Giuseppe Procaccianti (VU), Energy-Efficient Software
14 Bart van Straalen (UT), A cognitive approach to modeling bad news conversations
15 Klaas Andries de Graaf (VU), Ontology-based Software Architecture Documentation
16 Changyun Wei (UT), Cognitive Coordination for Cooperative Multi-Robot Teamwork
17 André van Cleeff (UT), Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs
18 Holger Pirk (CWI), Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories
19 Bernardo Tabuenca (OUN), Ubiquitous Technology for Lifelong Learners
20 Lois Vanhée (UU), Using Culture and Values to Support Flexible Coordination
21 Sibren Fetter (OUN), Using Peer-Support to Expand and Stabilize Online Learning
22 Zhemin Zhu (UT), Co-occurrence Rate Networks
23 Luit Gazendam (VU), Cataloguer Support in Cultural Heritage
24 Richard Berendsen (UVA), Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation
25 Steven Woudenberg (UU), Bayesian Tools for Early Disease Detection
26 Alexander Hogenboom (EUR), Sentiment Analysis of Text Guided by Semantics and Structure
27 Sándor Héman (CWI), Updating compressed colomn stores
28 Janet Bagorogoza (TiU), Knowledge Management and High Performance; The Uganda Financial Institutions Model for HPO
29 Hendrik Baier (UM), Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains
30 Kiavash Bahreini (OU), Real-time Multimodal Emotion Recognition in E-Learning
31 Yakup Koç (TUD), On the robustness of Power Grids
32 Jerome Gard (UL), Corporate Venture Management in SMEs
33 Frederik Schadd (TUD), Ontology Mapping with Auxiliary Resources
34 Victor de Graaf (UT), Gesocial Recommender Systems
35 Jungxao Xu (TUD), Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction

2016 01 Syed Saiden Abbas (RUN), Recognition of Shapes by Humans and Machines
02 Michiel Christiaan Meulendijk (UU), Optimizing medication reviews through decision support: prescribing a better pill to swallow
03 Maya Sappelli (RUN), Knowledge Work in Context: User Centered Knowledge Worker Support
04 Laurens Rietveld (VU), Publishing and Consuming Linked Data
05 Evgeny Sherkhonov (UVA), Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers
06 Michel Wilson (TUD), Robust scheduling in an uncertain environment
07 Jeroen de Man (VU), Measuring and modeling negative emotions for virtual training
08 Matje van de Camp (TiU), A Link to the Past: Constructing Historical Social Networks from Unstructured Data
09 Archana Nottamkandath (VU), Trusting Crowdsourced Information on Cultural Artefacts
10 George Karafotias (VUA), Parameter Control for Evolutionary Algorithms
11 Anne Schuth (UVA), Search Engines that Learn from Their Users
12 Max Knobbout (UU), Logics for Modelling and Verifying Normative Multi-Agent Systems
13 Nana Baah Gyan (VU), The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach
14 Ravi Khadka (UU), Revisiting Legacy Software System Modernization
15 Steffen Michels (RUN), Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments
16 Guangliang Li (UVA), Socially Intelligent Autonomous Agents that Learn from Human Reward
17 Berend Weel (VU), Towards Embodied Evolution of Robot Organisms
18 Albert Meroño Peñuela (VU), Refining Statistical Data on the Web
19 Julia Efremova (Tu/e), Mining Social Structures from Genealogical Data
20 Daan Odijk (UVA), Context & Semantics in News & Web Search
21 Alejandro Moreno Célleri (UT), From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground
22 Grace Lewis (VU), Software Architecture Strategies for Cyber-Foraging Systems
23 Fei Cai (UVA), Query Auto Completion in Information Retrieval
24 Brend Wanders (UT), Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach
25 Julia Kiseleva (TU/e), Using Contextual Information to Understand Searching and Browsing Behavior
26 Dilhan Thilakarathne (VU), In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains
27 Wen Li (TUD), Understanding Geo-spatial Information on Social Media
28 Mingxin Zhang (TUD), Large-scale Agent-based Social Simulation - A study on epidemic prediction and control
29 Nicolas Höning (TUD), Peak reduction in decentralised electricity systems - Markets and prices for flexible planning
30 Ruud Mattheij (UvT), The Eyes Have It
31 Mohammad Khelghati (UT), Deep web content monitoring
32 Eelco Vriezekolk (UT), Assessing Telecommunication Service Availability Risks for Crisis Organisations
33 Peter Bloem (UVA), Single Sample Statistics, exercises in learning from just one example
34 Dennis Schunselaar (TUE), Configurable Process Trees: Elicitation, Analysis, and Enactment
35 Zhaochun Ren (UVA), Monitoring Social Media: Summarization, Classification and Recommendation
36 Daphne Karreman (UT), Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies

37 Giovanni Sileno (UvA), Aligning Law and Action - a conceptual and computational inquiry
38 Andrea Minuto (UT), Materials that Matter - Smart Materials meet Art & Interaction Design
39 Merijn Bruijnes (UT), Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect
40 Christian Detweiler (TUD), Accounting for Values in Design
41 Thomas King (TUD), Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance
42 Spyros Martzoukos (UVA), Combinatorial and Compositional Aspects of Bilingual Aligned Corpora
43 Saskia Koldijk (RUN), Context-Aware Support for Stress Self-Management: From Theory to Practice
44 Thibault Sellam (UVA), Automatic Assistants for Database Exploration
45 Bram van de Laar (UT), Experiencing Brain-Computer Interface Control
46 Jorge Gallego Perez (UT), Robots to Make you Happy
47 Christina Weber (UL), Real-time foresight - Preparedness for dynamic innovation networks
48 Tanja Buttler (TUD), Collecting Lessons Learned
49 Gleb Polevoy (TUD), Participation and Interaction in Projects. A Game-Theoretic Analysis
50 Yan Wang (UVT), The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains

---

2017 01 Jan-Jaap Oerlemans (UL), Investigating Cybercrime
02 Sjoerd Timmer (UU), Designing and Understanding Forensic Bayesian Networks using Argumentation
03 Daniël Harold Telgen (UU), Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines
04 Mrunal Gawade (CWI), Multi-core Parallelism in a Column-store
05 Mahdieh Shadi (UVA), Collaboration Behavior
06 Damir Vandic (EUR), Intelligent Information Systems for Web Product Search
07 Roel Bertens (UU), Insight in Information: from Abstract to Anomaly
08 Rob Konijn (VU) , Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery
09 Dong Nguyen (UT), Text as Social and Cultural Data: A Computational Perspective on Variation in Text
10 Robby van Delden (UT), (Steering) Interactive Play Behavior
11 Florian Kunneman (RUN), Modelling patterns of time and emotion in Twitter #anticipointment
12 Sander Leemans (TUE), Robust Process Mining with Guarantees
13 Gijs Huisman (UT), Social Touch Technology - Extending the reach of social touch through haptic technology
14 Shoshannah Tekofsky (UvT), You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior
15 Peter Berck (RUN), Memory-Based Text Correction
16 Aleksandr Chuklin (UVA), Understanding and Modeling Users of Modern Search Engines
17 Daniel Dimov (UL), Crowdsourced Online Dispute Resolution
18 Ridho Reinanda (UVA), Entity Associations for Search
19 Jeroen Vuurens (UT), Proximity of Terms, Texts and Semantic Vectors in Information Retrieval
20 Mohammadbashir Sedighi (TUD), Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility
21 Jeroen Linssen (UT), Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)
22 Sara Magliacane (VU), Logics for causal inference under uncertainty

23 David Graus (UVA), Entities of Interest — Discovery in Digital Traces
24 Chang Wang (TUD), Use of Affordances for Efficient Robot Learning
25 Veruska Zamborlini (VU), Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search
26 Merel Jung (UT), Socially intelligent robots that understand and respond to human touch
27 Michiel Joosse (UT), Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors
28 John Klein (VU), Architecture Practices for Complex Contexts
29 Adel Alhuraibi (UvT), From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT"
30 Wilma Latuny (UvT), The Power of Facial Expressions
31 Ben Ruijl (UL), Advances in computational methods for QFT calculations
32 Thaer Samar (RUN), Access to and Retrievability of Content in Web Archives
33 Brigit van Loggem (OU), Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity
34 Maren Scheffel (OU), The Evaluation Framework for Learning Analytics
35 Martine de Vos (VU), Interpreting natural science spreadsheets
36 Yuanhao Guo (UL), Shape Analysis for Phenotype Characterisation from High-throughput Imaging
37 Alejandro Montes Garcia (TUE), WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy
38 Alex Kayal (TUD), Normative Social Applications
39 Sara Ahmadi (RUN), Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR
40 Altaf Hussain Abro (VUA), Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems
41 Adnan Manzoor (VUA), Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle
42 Elena Sokolova (RUN), Causal discovery from mixed and missing data with applications on ADHD datasets
43 Maaike de Boer (RUN), Semantic Mapping in Video Retrieval
44 Garm Lucassen (UU), Understanding User Stories - Computational Linguistics in Agile Requirements Engineering
45 Bas Testerink (UU), Decentralized Runtime Norm Enforcement
46 Jan Schneider (OU), Sensor-based Learning Support
47 Jie Yang (TUD), Crowd Knowledge Creation Acceleration
48 Angel Suarez (OU), Collaborative inquiry-based learning

2018 01 Han van der Aa (VUA), Comparing and Aligning Process Representations
02 Felix Mannhardt (TUE), Multi-perspective Process Mining
03 Steven Bosems (UT), Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction
04 Jordan Janeiro (TUD), Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks
05 Hugo Huurdeman (UVA), Supporting the Complex Dynamics of the Information Seeking Process
06 Dan Ionita (UT), Model-Driven Information Security Risk Assessment of Socio-Technical Systems
07 Jieting Luo (UU), A formal account of opportunism in multi-agent systems
08 Rick Smetsers (RUN), Advances in Model Learning for Software Systems
09 Xu Xie (TUD), Data Assimilation in Discrete Event Simulations

10 Julienka Mollee (VUA), Moving forward: supporting physical activity behavior change through intelligent technology
11 Mahdi Sargolzaei (UVA), Enabling Framework for Service-oriented Collaborative Networks
12 Xixi Lu (TUE), Using behavioral context in process mining
13 Seyed Amin Tabatabaei (VUA), Computing a Sustainable Future
14 Bart Joosten (UVT), Detecting Social Signals with Spatiotemporal Gabor Filters
15 Naser Davarzani (UM), Biomarker discovery in heart failure
16 Jaebok Kim (UT), Automatic recognition of engagement and emotion in a group of children
17 Jianpeng Zhang (TUE), On Graph Sample Clustering
18 Henriette Nakad (UL), De Notaris en Private Rechtspraak
19 Minh Duc Pham (VUA), Emergent relational schemas for RDF
20 Manxia Liu (RUN), Time and Bayesian Networks
21 Aad Slootmaker (OUN), EMERGO: a generic platform for authoring and playing scenario-based serious games
22 Eric Fernandes de Mello Araujo (VUA), Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks
23 Kim Schouten (EUR), Semantics-driven Aspect-Based Sentiment Analysis
24 Jered Vroon (UT), Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots
25 Riste Gligorov (VUA), Serious Games in Audio-Visual Collections
26 Roelof Anne Jelle de Vries (UT),Theory-Based and Tailor-Made: Motivational Messages for Behavior Change Technology
27 Maikel Leemans (TUE), Hierarchical Process Mining for Scalable Software Analysis
28 Christian Willemse (UT), Social Touch Technologies: How they feel and how they make you feel
29 Yu Gu (UVT), Emotion Recognition from Mandarin Speech
30 Wouter Beek, The "K" in "semantic web" stands for "knowledge": scaling semantics to the web

2019 01 Rob van Eijk (UL),Comparing and Aligning Process Representations
02 Emmanuelle Beauxis Aussalet (CWI, UU), Statistics and Visualizations for Assessing Class Size Uncertainty
03 Eduardo Gonzalez Lopez de Murillas (TUE), Process Mining on Databases: Extracting Event Data from Real Life Data Sources
04 Ridho Rahmadi (RUN), Finding stable causal structures from clinical data
05 Sebastiaan van Zelst (TUE), Process Mining with Streaming Data
06 Chris Dijkshoorn (VU), Nichesourcing for Improving Access to Linked Cultural Heritage Datasets
07 Soude Fazeli (TUD),
08 Frits de Nijs (TUD), Resource-constrained Multi-agent Markov Decision Processes
09 Fahimeh Alizadeh Moghaddam (UVA), Self-adaptation for energy efficiency in software systems