Approximation Algorithms in Allocation, Scheduling and Pricing

APPROXIMATION ALGORITHMS IN ALLOCATION, SCHEDULING AND PRICING

TIM OOSTERWIJK

Cover design by Patrique Mordant. Part of it is the cover of the Nightwish album Endless Forms Most Beautiful, whose artwork is by Janne Pitkänen.

This book was typeset by the author using LaTeX and the classicthesis package.

Published by Universitaire Pers Maastricht
ISBN: 978 94 6159 783 0
Printed in The Netherlands by Datawyse Maastricht

APPROXIMATION ALGORITHMS IN ALLOCATION, SCHEDULING AND PRICING

DISSERTATION

to obtain the degree of Doctor at Maastricht University,
on the authority of the Rector Magnificus,
Prof. dr. Rianne M. Letschert,
in accordance with the decision of the Board of Deans,
to be defended in public
on Friday, 19th of January 2018, at 16.00 hours

by

Tim Oosterwijk

Promotor:
Prof. dr. R.J. Müller

Co-Promotor:
Dr. T. Vredeveld

Assessment Committee:
Prof. dr. ir. C.P.M. van Hoesel (chair)
Dr. A. Berger
Prof. dr. B. Peis
Prof. dr. G. Schäfer

This research was financially supported by the Graduate School of Business and Economics (GSBE).

To my family and friends.

ACKNOWLEDGEMENTS

This thesis represents the culmination of my research effors during the years of my PhD, and it would not be in its current shape had it not been for the help of many people. Though words cannot express the gratitude I feel towards some of you, I thank everyone who contributed to any part of this journey. In this section, I will thank some people in particular.

First of all I want to express my huge appreciation for my supervisor Tjark Vredeveld. Where to start? Thank you for offering me this PhD track. You made sure I expanded my academic network, attended nice conferences, and collaborated with numerous researchers. You were so understanding and helpful when my private life hindered my work life. We cooperated very well in both research and teaching. I have come to see you more as a friend than as my supervisor and I thank you for everything. I hope we spend more time together, doing research or drinking wine; introducing me to that is, as you say yourself, perhaps your greatest achievement.

I would like to thank my promotor Rudolf Müller for having me as his PhD student, giving me freedom in my research, and of course for reading and approving the thesis. I would also like to thank the members of my assessment committee: Stan van Hoesel, André Berger, Britta Peis and Guido Schäfer. Thank you for taking the time to read and approve my thesis. An extra thank you for Stan and André for the nice teaching we did together and to Guido Schäfer for the great course on Algorithmic Game Theory.

Also I would like to thank my co-authors. I am very grateful to Tobias Harks for introducing me to the field of congestion games, and of course for the nice years as a direct colleague and the research visit in Augsburg. A big thank you to José Correa for the great time I had during my research visit in Chile; it was really inspiring to work with you and I hope we will collaborate more in the future. A huge thanks to Nikhil Bansal for our nice cooperation and of course for your supervision of my Master thesis. Vincent Kreuzen, thank you so much for being a great friend all these years and of course for being one of my paranymphs during the official ceremony. Also Alexander Grigoriev, thanks for being such a nice colleague and of course for the teaching. A thank you to Ruben Hoeksma and Patricio Foncea Araneda for your contributions to my nice stay in Chile. And thanks Judith Keijsper, Ruben van der Zwaan and Michaël Gabay for everything.

A big thank you also to all other senior staff members in our department. I would like to start by thanking Karin van den Boorn and Yolanda Paulissen for being the most professional and fun secretaries I could have hoped for. Also, I want to thank János Flesch and Hans Peters for your contributions to my training Grants for Individuals. Thank you, Aida Abiad, for your nice company and encouraging me to speak Spanish. A big thank you to Dries Vermeulen for your nice company during all those lunches, and to Matthias Mnich for being such a nice colleague.

Let me turn my attention to my fellow PhD students over these years. Of course I want to thank my office mate, great friend and paranymph Veerle Timmermans for all the good times we had at the office and our homes, and for dragging me to the gym every time. Thank you, Andrej Winokurow, for being a cool office mate. A big thank you to Anna Zseleva for your constant energy and great company, you were the best office neighbour I could wish for. A huge thanks to Abhinaba Lahiri, Alexander Heinemann, Andrey Kateshov, Anne Balter, Artem Duplinskiy, Caterina Schiavoni, Etiënne Wijler, Gergely Csapó, Hanno Reuvers, Jan Lohmeyer, Martijn van Brink, Marc Schröder, Oksana Balabay, Rasmus Lönn, Roland Vincze, Sean Telg, Shashwat Khare, Swarnendu Chatterjee and Verena Jung for the nice lunches, good atmosphere and awesome activities we did outside of work. Thanks to all other PhDs for being great colleagues.

A big thank you to Marc Uetz, Anupam Gupta and Viswanath Nagarajan for your interesting lectures during the autumn school, and to Nicole Megow for the organiation. I want to thank Johann Hurink, Andreas Wierz, Annette Ficker, Bart de Keijzer, Bart Kamphorst, Denise Tönissen, Jannik Matuschke, Jasper de Jong, Kevin Schewior, Loe Schlicher, Martijn van Ee, Pieter Kleer, Suzanne van der Ster and Thijs van der Klauw and many others for their nice company during many conferences. Thanks, Anja Huber and Manuel Surek, for the cocktails in Augsburg. Thanks, José Verschae, for your company in Chile and the invitation to speak at the (cake) seminar.

Last but not least, I would like to express an enormous amount of gratitude towards all people outside of my academic circles whom I did not thank yet. A huge thanks to Patrique Mordant for your great support during these years and the cover you designed. A big thanks to my best friends Hans van den Hurk, Max Vroomen, Milou Vroomen-Bettings and Pascal Lardinois, and to all other friends for the awesome times we had. And of course a big thank you to my mother, father, brother and sister for everything you did in all the years leading to this moment. I cannot express how deeply grateful I am for everything you have done. Thank you.

CONTENTS

1 INTRODUCTION

1.1 OPTIMIZATION

This thesis concerns itself with four optimization problems stemming from the fields of algorithmic game theory, scheduling and mechanism design. In these optimization problems, the objective is to select the best element from a given set of alternatives. An optimization problem is determined by its collection of instances, defined as follows.

Definition 1.1. An instance $\mathcal{I}$ of an optimization problem $\mathcal{P}$ is defined by a set of feasible solutions S and a cost function $f : S \to \mathbb{R}$ mapping each feasible solution to a real value.

The goal is to select a feasible solution $s^* \in S$ such that $f(s^*) \leqslant f(s)$ for all $s \in S$ in case $\mathcal{P}$ is a minimization problem, or such that $f(s^*) \geqslant f(s)$ for all $s \in S$ in case $\mathcal{P}$ is a maximization problem.

Usually, both the set of feasible solutions S and the cost function f are given implicitly.

There is an important subfield within mathematical optimization that concerns itself with optimization problems with the additional requirement that the set of feasible solutions S is either finite or countably infinite. Such problems are called *combinatorial optimization problems*. Let us look at the following famous combinatorial optimization problem.

Problem 1.2 (Travelling Salesperson Problem (TSP))**.** Let a list of n cities $1, 2, \ldots, n$ be given, and the distances between every pair of cities. What is the shortest possible route that starts in city 1, visits all other cities exactly once and returns to city 1?

The initial goal of researchers studying this kind of problems was to find optimal solutions in the least possible amount of time. This amount of time usually grows with the size of the input of the instance at hand. Formally, the *input size* $|\mathcal{I}|$ of an instance $\mathcal{I}$ is the number of bits needed to succinctly encode the instance. To quantify the time efficiency of an algorithm depending on the input size, the notion of the running time of an algorithm was introduced. Roughly speaking, the *running time* of an algorithm is the

worst-case number of elementary operations the algorithm performs before terminating, expressed as a function of the input size.

For some problems, researchers succeeded in developing algorithms that find an optimal solution and have an efficient running time, while other problems, like Problem 1.2, seem to be notoriously hard. The distinction between these two types of problems is one of the topics of the field of computational complexity theory.

1.2 COMPUTATIONAL COMPLEXITY THEORY

Computational complexity theory is the branch of computer science that tries to classify computational problems according to their difficulty. For easy problems we can find an algorithm that solves the problem to optimality and whose running time is bounded by a polynomial in the size of the input.

Definition 1.3 (Polynomial time algorithm)**.** Consider a problem $\mathcal{P}$. An algorithm $\mathcal{A}$ is said to run in *polynomial time* if there exists a polynomial p such that for every instance $\mathcal{I}$ of size $|\mathcal{I}|$ the running time of $\mathcal{A}$ is bounded from above by $p(|\mathcal{I}|)$.

Consider Problem 1.2, the TSP. In spite of immense efforts, no researcher has yet been able to find a polynomial time algorithm solving this problem. In fact, it is generally believed that there does not exist a polynomial time algorithm that solves the TSP, although no one has been able to prove or disprove this despite decades of research. Formally, the TSP belongs to a class of problems called *NP-hard* problems; we refer to Garey and Johnson [43] for an extensive treatment of this topic. Problems that are NP-hard are commonly believed to be unsolvable in polynomial time.

This notion of the difficulty of a problem is widely used. There is a whole zoo of other complexity classes and assumptions, some of which will be used throughout the different chapters of this thesis. These hardness assumptions allow us to prove that other problems are also likely to be hard, by means of a *reduction*. Consider a problem $\mathcal{P}$ and suppose it is possible to transform any instance $\mathcal{I}'$ of the TSP into some instance $\mathcal{I}$ of $\mathcal{P}$ and to transform a solution of $\mathcal{I}$ back into the corresponding solution of $\mathcal{I}'$. Now assume that there exists a polynomial time algorithm that solves any instance of $\mathcal{P}$. Then we could solve any instance $\mathcal{I}'$ of the TSP by formulating it as an instance $\mathcal{I}$ of $\mathcal{P}$, solving $\mathcal{I}$ and translating its solution back to a solution of $\mathcal{I}'$. If both transformations can be done in polynomial time, we

could solve any instance of the TSP in polynomial time by reducing it to an instance of $\mathcal{P}$ that we can solve and translate back. Hence, $\mathcal{P}$ is at least as hard as TSP, since a polynomial time algorithm for $\mathcal{P}$ implies a polynomial time algorithm for TSP.

Using these kind of reductions, thousands of problems have been shown to be NP-hard or to belong to another complexity class of problems that are believed to be hard. No one has been able to formally prove whether or not there exist polynomial time algorithms solving NP-hard problems, but this is deemed highly unlikely by most scientists. Although it seems the structures underlying NP-hard problems do not offer sufficient support to find optimal solutions efficiently, it is often the case that they do offer sufficient support to find *near-optimal* solutions efficiently. This is the topic of the field of approximation algorithms.

1.3 APPROXIMATION ALGORITHMS

One of the directions researchers can take to deal with hard problems, is to search for polynomial time algorithms that find solutions whose values are not too far off from the optimal value.

Definition 1.4. Let $\mathcal{P}$ be a minimization problem. An algorithm is called an *approximation algorithm* for $\mathcal{P}$ with *approximation guarantee* $\alpha > 1$ if for any instance of the given problem, the value of the output of the algorithm is at most a factor α times the optimum value for that instance.

In case $\mathcal{P}$ is a maximization problem, the value of the output of the algorithm needs to be at least a factor $\alpha < 1$ times the optimum value for that instance.

The closer α is to 1, the closer the solution value of the approximation algorithm is to the optimal solution value. α is also called the approximation factor, approximation ratio or performance guarantee, and the algorithm is an *α-approximation algorithm*. In the literature, sometimes the definition of an approximation algorithm requires a polynomial running time; in this thesis we do not make this assumption and state the running time explicitly when necessary.

An extension of an approximation algorithm is an *approximation scheme*.

Definition 1.5. An *approximation scheme* is a family of approximation algorithms, such that for every $\varepsilon > 0$ there is an algorithm in this family that

returns a $(1+\varepsilon)$-approximation in case of a minimization problem, or a $(1-\varepsilon)$-approximation in case of a maximization problem.

A *polynomial time approximation scheme (PTAS)* is an approximation scheme for which each algorithm in this family has a running time bounded by a polynomial in the input size $|\mathcal{I}|$ for every fixed ε.

An *efficient polynomial time approximation scheme (EPTAS)* is a PTAS where the running time of each algorithm is bounded by $f(1/\varepsilon)\text{poly}(|\mathcal{I}|)$, where f is some not necessarily polynomial function and $\text{poly}(|\mathcal{I}|)$ is a polynomial function in the input size $|\mathcal{I}|$.

If the function f is also a polynomial, we refer to the approximation scheme as a *fully polynomial time approximation scheme (FPTAS)*.

Choosing ε close to 0 yields a solution whose value is close to the optimum, but this comes at the expense of extra running time of the algorithm. The goal in approximation algorithms is to find approximation algorithms with the approximation guarantee as close to 1 as possible, or approximation schemes with the lowest possible running time. Some optimization problems, however, are also hard to approximate, meaning that under some complexity assumption there does not exist a polynomial time approximation algorithm achieving a performance guarantee better than some value, or there does not exist a PTAS, EPTAS or FPTAS. The ultimate goal for researchers is to develop an approximation algorithm with a certain approximation ratio, or an approximation scheme, that matches the inapproximability of the problem. For an extensive treatment of approximation algorithms we refer to Williamson and Shmoys [99] and Vazirani [97].

1.4 OUTLINE OF THIS THESIS

1.4.1 *Polymatroid congestion models*

Chapter 2 is based on Harks, Oosterwijk, and Vredeveld [54]. It is dedicated to minimizing a monotone separable function over an integral polymatroid. A *polymatroid* is an abstract structure that incorporates many other structures, such as spanning trees in a graph. Roughly speaking, it is a polyhedron P whose boundaries are defined by a function that is monotone, normalized and submodular (cf. Section 2.2.2). The corresponding integral polymatroid base polyhedron is then comprised of the set of integer points

in P with maximum L_1-norm. This maximum value is referred to as the rank of the polymatroid.

The problem we consider stems from polymatroid congestion games. *Congestion games* are a widely used model to represent a situation in which different players compete over a set of exhaustible resources. In these games, every resource has a load-dependent cost function. We assume these to be non-decreasing, meaning costs can go up with higher resource utilization. In this chapter we consider *polymatroid congestion games*, in which the strategy space of every player consists of a player-specific integral polymatroid base polyhedron on the ground set of resources. A simultaneous choice of the strategies of all players induces a strategy profile, whose social cost is defined as the sum of the costs of the resources given the loads of the strategy profile. In this chapter, our objective is to find a strategy profile minimizing its social costs.

For *convex* cost functions, it was already known that an optimal strategy profile can be found in polynomial time in spanning tree congestion games [98], matroid congestion games [2] and even polymatroid congestion games [40, 47]. For *fixed* resource costs, Wolsey [100] proves that the greedy algorithm yields a logarithmic approximation of the optimal social costs. Contrary to these results, we consider general *non-decreasing* cost functions. For these cost functions, Harks and Falkenhausen [52] give a logarithmic approximation for singleton congestion games.

In this chapter, we extend the greedy algorithm for optimization over polymatroids to the setting of general non-decreasing cost functions, by devising an H_ρ-approximation algorithm, where ρ is the sum of the ranks of the polymatroids and H_ρ denotes the ρ-th harmonic number. Since we show this problem is as hard as Set Cover, the approximation guarantee is best possible up to a constant factor.

1.4.2 *High multiplicity scheduling*

Parts of Chapter 3 are based on Gabay, Grigoriev, Kreuzen, and Oosterwijk [41]. It studies the single machine capacitated lot-sizing problem, where n types of products need to be scheduled on one machine. Each product is associated with a constant demand rate, maximum production rate and holding costs per time unit. Every time the machine switches production, a sequencing cost is incurred. The goal is to find a cyclic schedule

minimizing total average costs, subject to the condition that all demands are satisfied.

This problem is a HIGH MULTIPLICITY SCHEDULING PROBLEM, as its input is succinctly encoded: all product attributes are specified per product type, and the multiplicities of the products to be produced are implicitly given through the demand rates.

For our problem with fixed production rates, Haase [48] gives a heuristic, and Haase and Kimms [49] solve it to optimality by making some additional assumptions on the problem instances.

For a fixed time horizon and sequence-independent sequencing costs and product-independent holding costs, Goyal [46] solves the problem to optimality. Madigan [78] gives a heuristic for this problem including setup times. When both the sequencing costs and the holding costs are product-dependent, Boctor [14] gives an exact algorithm for the case of two products.

In contrast to this last work, our sequencing costs do not only depend on the last produced product, but also on the new product to be produced. We prove a number of structural properties largely characterizing optimal solutions to the problem. We use these properties to optimally solve instances with few products. Then we present two algorithms approximating optimal schedules for an arbitrary number of products by augmenting the problem input.

It is known that high multiplicity scheduling problems, even without sequencing costs, become highly non-trivial with respect to the output sizes and computational complexity. Particularly, the length of an optimal solution can be exponential in the input size of the problem. Nevertheless, our algorithms produce good quality polynomial length schedules compared to the exponential length optimal schedules.

1.4.3 *Vector scheduling*

Chapter 4 is based on Bansal, Oosterwijk, Vredeveld, and van der Zwaan [10]. It considers the VECTOR SCHEDULING problem, a natural generalization of the classical makespan minimization problem to multiple resources. In that problem, a set of jobs needs to be scheduled on a set of machines. Every job has a processing time and needs to be processed by one machine, and every machine can process at most one job at a time. The goal is to find a schedule in which the machine that finishes last, finishes as early as possi-

ble. This problem is strongly NP-hard [43] and therefore does not admit an FPTAS unless P=NP. The best known EPTAS is due to Jansen [65] with a running time of $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\varepsilon^2)} + n^{\mathcal{O}(1)})$ ($\tilde{\mathcal{O}}$ suppresses polylogarithmic terms).

In the Vector Scheduling problem, a job's processing time is no longer measured by a single parameter but involves multiple aspects. As an example, consider computer tasks that require a certain amount of both CPU and memory, which we need to schedule on a set of computers that we do not want to overload in either way. Formally, we are given n jobs, represented as d-dimensional vectors in $[0,1]^d$, and m identical machines, and the goal is to assign the jobs to machines such that the maximum load of each machine over all the coordinates is at most 1.

In this context, an approximation algorithm with performance guarantee α is understood to return an assignment with maximum load at most α whenever there exists a feasible schedule of maximum load at most 1. For fixed d, the problem admits an approximation scheme, and the best known running time is $n^{f(\varepsilon,d)}$ where $f(\varepsilon,d) = (1/\varepsilon)^{\tilde{\mathcal{O}}(d)}$ [23]. In particular, the dependence on d is double exponential.

In this chapter we show that a double exponential dependence on d is necessary, and give an improved algorithm with essentially optimal running time. Specifically, we show that for any $\varepsilon < 1$, there is no $(1+\varepsilon)$-approximation with running time $\exp(o(\lfloor 1/\varepsilon \rfloor^{d/3}))$, where $\exp(x)$ denotes 2^x, unless the Exponential Time Hypothesis (ETH) fails. The ETH is a complexity assumption that states that a certain problem (3-SAT) cannot be solved in a certain amount of time (cf. Hypothesis 4.6 for the details). Like the assumption that P $\neq$ NP, the ETH has yet to be proved or disproved.

Moreover, we prove that there exists no $(1+\varepsilon)$-approximation with running time $\exp(\lfloor 1/\varepsilon \rfloor^{o(d)})$, unless NP has subexponential time algorithms. This is a more standard complexity-theoretic assumption than the ETH. We prove that similar lower bounds also hold even if εm extra machines are allowed (i.e. with resource augmentation), for sufficiently small $\varepsilon > 0$. Finally, we complement these lower bounds with a $(1+\varepsilon)$-approximation that runs in time $\exp((1/\varepsilon)^{\mathcal{O}(d \log\log d)}) + nd$. This gives the first efficient approximation scheme (EPTAS) for the problem.

1.4.4 *Optimal stopping and posted prices*

Chapter 5 is based on Correa, Foncea, Hoeksma, Oosterwijk, and Vredeveld [28]. It concerns itself with optimal stopping theory and posted price mech-

anisms, and in particular the classic prophet inequality. In optimal stopping theory, we consider a gambler facing a finite sequence of non-negative independent random variables. These random variables arrive in an iterative fashion and upon arrival a prize is drawn from the distribution underlying the random variable. The gambler is allowed to stop the sequence at any time to obtain the drawn prize he currently faces. The question is which strategy he should employ to maximize his expected reward. A classical example is the secretary problem, in which the possible candidates arrive one by one and the hiring committee only learns the true value of each candidate upon arrival.

The classic prophet inequality states that the gambler can obtain, in expectation, at least half as much reward as a *prophet* who knows the values of each random variable and can choose the largest one. The fraction $1/2$ is also best possible [73, 74].

In this chapter we consider both a *non-adaptive* and an *adaptive* version of this setting. In the former case the gambler sets a threshold for every random variable a priori, so the thresholds can only depend on the underlying distribution of the random variable. In the latter case, the thresholds are set when a random variable arrives, thus enabling the gambler to include information about previously rejected random variables in the new threshold he needs to set. The gambler then receives a reward equal to the first arriving random variables with a realization exceeding the threshold. We assume the random variables arrive uniformly at random.

For the non-adaptive case, we obtain an algorithm computing thresholds achieving an expected reward within at least a $1 - 1/e$ fraction of the expected maximum and prove this constant is optimal. For the adaptive case with independent and identically distributed (i.i.d.) random variables, we obtain a tight 0.745-approximation, proving a conjecture of Hill and Kertz [58] from 1982. They characterize a sequence α_n, the best possible factor using a threshold rule when faced with n i.i.d. random variables. They prove α_n is at least $1 - 1/e$ for all n, conjecture that the sequence is monotone and leave open the existence and computation of its limit. Very recently, Abolhassani et al. [1] improved the bound to 0.738. We prove the sequence is monotone and prove its limit is 0.745, thereby establishing a tight bound for all n on the performance of the best stopping rule for i.i.d. random variables.

We continue by revealing a reduction from optimal stopping theory to *posted price mechanisms*. In this setting, a seller wants to sell a single item to a set of n potential customers in order to maximize his expected revenue.

Since the optimal auction from Myerson [83] is quite involved, it is rarely used in practice and researchers started to look for simple mechanisms that approximate the optimal expected revenue well. One of these simple mechanisms is a posted price mechanism, in which the customers arrive one by one and the seller sets a take-it-or-leave-it price for everyone. If the customer accepts, he makes a profit equal to the price he charged. Otherwise, the customer leaves the system and he makes an offer to the next customer.

Also here, we consider both a non-adaptive and an adaptive setting. In the non-adaptive setting, the seller sets the prices a priori, so they only depend on the underlying distribution from which the valuation of the customer is drawn upon arrival. Previously, it was known that if the seller is allowed to choose the order of the customers himself, it is possible to achieve an expected reward of at least a $1 - 1/e$-fraction of the optimal expected revenue [22]. Using the reduction we provide in this chapter, we prove that the same bound can be obtained in the random arrival setting. As is the case in the corresponding result in the optimal stopping theory setting, this result is tight.

On the other hand, in the adaptive setting, the seller is allowed to set the prices upon arrival of the customers. With this more powerful pricing scheme, we prove he can obtain an expected revenue of at least a 0.745 fraction of the optimal expected revenue. The previously best known result for this adaptive setting was a $1 - 1/e$-fraction [34] and the best known upper bound was 0.79 [13]. We believe the tight instances of Hill and Kertz [58] can be transformed from the optimal stopping theory setting to the posted price mechanisms setting, which would make this bound also tight in this setting.

1.5 PUBLICATIONS

The chapters in this dissertation are based upon the following publications.

Published

- T. Harks, T. Oosterwijk, and T. Vredeveld. "A Logarithmic Approximation for Polymatroid Congestion Games". In: *Operations Research Letters* 44.6 (2016), pp. 712–717.
- M. Gabay, A. Grigoriev, V. J. C. Kreuzen and T. Oosterwijk. "High Multiplicity Scheduling with Switching Costs for Few Products". In:

Operations Research Proceedings 2014, Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), RWTH Aachen University, Germany, September 2-5, 2014. 2014, pp. 437–443.

- N. Bansal, T. Oosterwijk, T. Vredeveld, and R. van der Zwaan. "Approximating Vector Scheduling: Almost Matching Upper and Lower Bounds". In: *Algorithmica* 76.4 (2016), pp. 1077–1096.

- J. R. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, and T. Vredeveld. "Posted Price Mechanisms for a Random Stream of Customers". In: *Proceedings of the 2017 ACM Conference on Economics and Computation*. EC '17. Cambridge, Massachusetts, USA: ACM, 2017, pp.169–186.

Submitted for publication

- A. Grigoriev, V. J. C. Kreuzen and T. Oosterwijk. "High Multiplicity Scheduling with Sequencing Costs". Submitted for publication in: *Institute for Industrial and Systems Engineers (IISE) Transactions*.

- J. R. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, and T. Vredeveld. "Optimal Threshold Strategies and Posted Price Mechanisms for Random Arrivals". Submitted for publication in: *Journal of the ACM*.

2 POLYMATROID CONGESTION MODELS

2.1 INTRODUCTION

Congestion games have become a standard game-theoretic model describing the allocation of exhaustible resources by selfish players. In the basic model of Rosenthal [86], there is a finite set of players and resources and each player is associated with a set of allowable subsets of resources. A pure strategy of a player consists of an allowable subset. Congestion on a resource is modelled by a load-dependent cost function which is usually non-decreasing and solely depends on the number of players using the resource. In the context of *network games*, the resources may correspond to edges of a graph, the allowable subsets correspond to the simple paths connecting a source and a sink and players choose minimum cost paths. Rosenthal proved in his seminal paper that congestion games always admit a pure Nash equilibrium.

In this chapter, we focus on so-called *polymatroid congestion games*, where the strategy space of every player consists of the set of vectors in a player-specific integral polymatroid base polyhedron defined on the ground set of resources. This can be viewed as a game in which every player chooses a multiset of the resources rather than a subset. These games have numerous applications as they include for instance matroid congestion games, singleton congestion games (rank-1 matroids) or spanning tree congestion games, where every player selects a spanning tree of a player-specific subgraph of a given graph. We consider the problem of computing a minimum cost solution in polymatroid congestion games.

2.1.1 *Applications in Optimization*

Computing a minimum cost solution over an integral polymatroid base polytope is a core problem in combinatorial optimization that has received considerable attention in the past, cf. [40, 47, 100]. These works pose quite restrictive assumptions on the used cost functions, namely, that they either have fixed-cost structure ([100]), or are separable convex ([40, 47]).

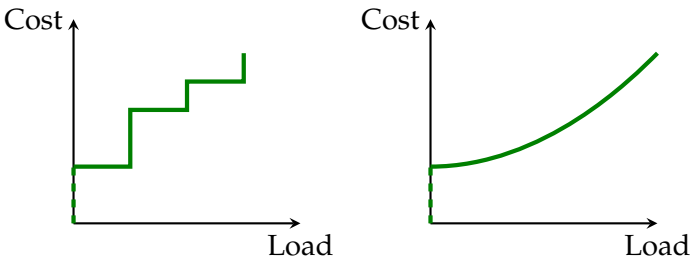

Figure 2.1: Cost functions with non-concave/convex cost functions.

For several practical applications, however, these assumptions do not apply. Consider the *tree packing problem* in network design, where there is an undirected graph $G = (V, E)$ with non-negative and non-decreasing edge cost functions $c_e(\ell), e \in E$. The goal is to compute integral capacities on edges so that we can serve n minimum spanning trees in G with minimum cost. In contrast to previous works, by allowing arbitrary non-decreasing cost functions we are able to model more realistic cost functions occurring in practice. Typical cost functions are step functions (see Fig. 2.1 (left)), where every cost level corresponds to a different cable type that can be installed (cf. [6]). Andrews et al. [5], for instance, discussed telecommunication network design problems, where cost functions with "diseconomies of scale" are used to model the cost of energy consumption when routers apply speed scaling to process packets. The used cost function has the form $c_r(\ell) = \sigma + \delta \cdot \ell^\alpha, \alpha > 1, \sigma, \delta > 0$, if $\ell > 0$ and $c_r(0) = 0$. See Fig. 2.1 (right) for an illustration. Clearly, this function and also the previous function are neither concave nor convex.

2.1.2 *Applications in Game Theory*

The problem of computing minimum cost solutions in polymatroid congestion games is important for scenarios where a central planner can implement a solution or when players collaborate. Additionally, minimum cost solutions serve as building blocks for other cost-efficient solutions, e.g., as in [36], where a minimum cost solution is used for defining cost sharing protocols with low price of stability/anarchy. In fact, Ackermann et al. [2, Section 2.2.] as well as von Falkenhausen and Harks [36, Section 5] state as an open problem to characterize the computational complexity

of computing a minimum cost solution in matroid congestion games with non-decreasing cost functions, which is a special case of the problem we consider in this chapter (cf. Section 2.4). However, in the case computing an optimal solution is NP-hard, one of the common approaches is to find approximation algorithms. Meyers and Schulz [80] investigated the limitations of such an approach by studying the inapproximability of several of these problems. We complement this type of research by showing a logarithmic approximation that is best possible up to a constant factor.

2.1.3 *Our Results*

We devise an H_ρ-approximation algorithm, where ρ is the sum of the ranks of the player-specific polymatroids (the value of the polymatroid function on the entire set of resources) and H_ρ denotes the ρ-th harmonic number. This approximation guarantee is best possible (up to a constant factor) for algorithms with polynomial running time, unless NP $\subseteq$ TIME$(n^{\mathcal{O}(\log\log n)})$. As a by-product we show that matroid congestion games are also H_ρ-approximable in polynomial time, a result that partially settles an open problem of [2, 36]. For matroid congestion games, we only leave a gap if ρ is not polynomially bounded in the number of players.

Our algorithm maintains data structures for *target loads* and *preliminary cost-per-unit values* for every resource and its respective load. The algorithm iteratively increases the target loads on the resources by selecting the resource (with corresponding target load) having the lowest cost per unit. After this greedy choice, a covering oracle is invoked that checks whether or not there exists a feasible strategy profile (a vector in the sum of the player-specific integral polymatroid base polyhedra) covering the currently computed target loads. If the covering oracle returns a feasible strategy profile, we update target loads and preliminary cost-per-unit values and proceed. If the oracle returns infeasibility, we reduce the maximum target load for the selected resource and proceed. Denoting the number of resources by m, pseudopolynomial running time in terms of oracle calls follows as there are only $m\,\rho$ possible target loads and in each iteration a target load is increased or a maximum target load is decreased. If ρ is polynomial in the input size (as is the case for matroids, arborescences, etc.) the algorithm runs in polynomial time. The oracle itself can also be implemented in linear oracle time using an adaptation of the polymatroid greedy algorithm.

2.1.4 *Literature Review*

COMPUTING SOCIAL OPTIMA Werneck et al. [98] studied the complexity of computing a social optimum in spanning tree congestion games. For convex cost functions they devised an efficient algorithm computing an optimal solution. Essentially, convex cost functions allow to linearize the cost function and then to apply a greedy algorithm. Ackermann et al. [2] extended the work of [98] by observing that the same idea is applicable to matroid congestion games still requiring that cost functions are convex. For spanning tree games with non-monotonic cost functions, they showed that computing a social optimum is NP-hard. The case of matroid congestion games with general non-decreasing cost functions is posed as an open problem [2, Section 2.2.].

It should be noted that the positive results for convex cost functions were already implied by previous works, perhaps not so obvious. Groenevelt [47] and Fujishige [40] presented polynomial time algorithms to minimize a convex separable function over an integral polymatroid base polyhedron. Since the matroid rank function is submodular, the strategy space for every player can equivalently be represented as an integral polymatroid base polyhedron. Using that the sum of polymatroid base polyhedra is again a polymatroid base polyhedron, the results of Groenevelt [47] and Fujishige [40] thus already imply a polynomial time algorithm for computing a social optimum for matroid congestion games with convex cost functions. For polymatroids with fixed costs for all resources, Wolsey [100] showed that the greedy algorithm gives a logarithmic approximation. In contrast to these works, we consider the case of arbitrary non-decreasing cost functions.

A special case of polymatroid congestion games is that of singleton congestion games with arbitrary non-decreasing cost functions. Harks and von Falkenhausen [52] devised an H_n-approximation approximation algorithm for the social cost, where n is the number of players. Their algorithm is based on successive network flow computations on a suitably defined capacitated graph. Moreover, they showed this result is essentially best possible up to a constant factor, as they show that this optimization problem is hard to approximate within a factor of $c \log n$ for any $c < 1$, a reduction we will extend for our hardness result (cf. Lemma 2.8).

Meyers and Schulz [80] classified the complexity of computing a social optimum for general congestion games as well as network congestion games and differentiated between asymmetric and symmetric strategy spaces. In the case of network congestion games, they also distin-

guished the case in which all players share a common source. Regarding the cost functions, they differentiated between five types: non-decreasing, convex non-decreasing, non-increasing, concave non-increasing, and non-monotonic cost functions. For all combinations of strategy spaces and cost functions they established the complexity of finding the social optimum. Most of the resulting problems are inapproximable to any finite factor. In particular, the asymmetric case with non-decreasing costs is not approximable to any finite factor. Very recently, Roughgarden [89] studied the impact of the computational complexity of computing socially optimal solutions on the price of anarchy. He derived a reduction that translates inapproximability results to corresponding lower bounds on the price of anarchy. In the context of congestion games, he derived stronger inapproximability bounds for Rosenthal's congestion model involving polynomial cost functions with non-negative coefficients.

Computing a socially optimal profile has also been studied in the congestion model of Milchtaich [82], where resource costs are player-specific. Chakrabarty et al. [20] proved that the social optimum is inapproximable within any finite factor, unless $P = NP$. They exhibited some special cases in which a minimum cost solution can be found in polynomial time, e.g. when the number of strategies is bounded. Blumrosen and Dobzinski [12] considered the problem of maximizing welfare instead of minimizing costs and presented an 18-approximation for this problem. Assuming non-decreasing cost functions, they improved the approximation guarantee to $\frac{e}{e-1}$. In another study, De Keijzer and Schäfer [69] studied congestion games with positive externalities, where players benefit from other players choosing the same resource. They showed even very special cases of the problem are NP-hard and provided several approximation algorithms.

COMPUTING EQUILIBRIA In the past decade the computational complexity to find a pure Nash equilibrium (PNE) in congestion games has been studied extensively. Ackermann et al. [2] proved that for congestion games with non-decreasing cost functions, matroids are the maximal property on the strategy space of every player that guarantees that best responses for players converge to a PNE in polynomial time. Fabrikant et al. [35] showed that a PNE can be found in polynomial time in symmetric network congestion games with non-decreasing cost functions. However, for general network games with non-decreasing cost functions, finding a PNE is PLS-complete [35]. These results have been strengthened to hold even when the cost functions are non-decreasing and linear [2]. It is also

PLS-complete, for any $\alpha > 1$, to find α-approximate PNEs in congestion games [95], in which no player can unilaterally improve his cost by more than a factor α.

In singleton congestion games where all players have the same strategy space, the best Nash equilibrium can be found in polynomial time for any cost function [63]. Sperber [96] showed that both the best and the worst PNE can be found in polynomial time for non-decreasing cost functions with a greedy algorithm. However, she proved that in series-parallel graphs this is NP-hard, except in the case of finding the best PNE in a 2-player game. Also the setting in which the social cost is not determined by the sum of the costs but by the maximum cost (*makespan* social cost) has been studied. In this setting, the worst PNE can be found in series-parallel graphs, however, finding the best PNE is NP-complete [44]. Another related class of games are so-called *bottleneck congestion games*, in which the total cost of a player is not the sum but the maximum of the costs of the resources he chose. Harks et al. [55] devised an algorithm computing PNEs and strong equilibria which relies on the idea of a *strategy packing oracle*, which is similar to the strategy covering oracle in this chapter. They also maintain target loads that are iteratively updated, and they also use that for matroids, this strategy packing oracle can be implemented in polynomial time.

2.2 THE MODEL

2.2.1 *Congestion Models*

A *congestion model* is given by the tuple $\mathcal{M} = (N, R, (\mathcal{B}_i)_{i\in N}, (c_r)_{r\in R})$, where $N = \{1, \dots, n\}$ is a non-empty, finite set of players and $R = \{1, \dots, m\}$ is a non-empty, finite set of *resources*. Every player $i \in N$ chooses a *pure strategy* $\mathbf{b}^i \in \mathbb{N}^R$ from a non-empty bounded integral polyhedron $\mathcal{B}_i \subseteq \mathbb{N}^R$ (which can be seen as a non-empty, finite collection of multisets). In this chapter, we consider an integral polyhedron $P \subseteq \mathbb{N}^m$ to be the intersection of the lattice $\mathbb{N}^m$ with the polyhedron in $\mathbb{R}^m$. We denote the set of joint strategy profiles by the Minkowski sum of the integral polyhedra $\mathcal{B} = \mathcal{B}_1 + \ldots + \mathcal{B}_n$. Given a strategy profile $\mathbf{b} \in \mathcal{B}$, we define the *load* on a resource $r \in R$ as b_r, which is the r-th component of the vector $\mathbf{b}$. All resources $r \in R$ have a load-dependent *cost function* $c_r : \mathbb{N} \to \mathbb{N}$. Abusing notation, we write $c_r(\mathbf{b}) = c_r(b_r)$ for all $\mathbf{b} \in \mathcal{B}$. We assume for all $r \in R$ that $c_r(0) = 0$ and $c_r(i) \geqslant c_r(j)$ whenever $i \geqslant j$. We denote the *social cost* by $C(\mathbf{b}) = \sum_{r\in R} c_r(\mathbf{b})$.

Example 2.1. In spanning tree congestion games, the resources R are the edges of a given graph $G = (V, R)$. Players choose a spanning tree $\mathbf{b}^i \in 2^{R_i}$ of the set of possible spanning trees $\mathcal{B}_i$ on a player-specific subgraph $G_i = (V_i, R_i)$. The simultaneous choice $\mathbf{b}$ of n spanning trees induces a load b_r on every edge r defined as the number of chosen spanning trees containing r.

2.2.2 *Polymatroids*

A function $f : 2^R \to \mathbb{N}$ is called submodular if $f(U) + f(V) \geqslant f(U \cup V) + f(U \cap V)$ for all $U, V \subseteq R$. It is called monotone if $f(U) \leqslant f(V)$ for all $U \subseteq V$, and normalized if $f(\emptyset) = 0$. A pair (R, f) is an *integral polymatroid* if $f : 2^R \to \mathbb{N}$ is submodular, monotone and normalized. f is then called an *integral polymatroid rank function* and the associated integral polyhedron is defined as

$$\mathbb{P}_f := \left\{ \mathbf{b} \in \mathbb{N}^R \mid b(U) \leqslant f(U) \; \forall U \subseteq R \right\},$$

where we define the load on a set U of resources as $b(U) = \sum_{r \in U} b_r$ (note that $b(\emptyset) = 0$). Given the integral polyhedron $\mathbb{P}_f$ and the integer $\rho = f(R)$, which we refer to as the *rank* of the polymatroid, the corresponding *integral polymatroid base polyhedron* is

$$\mathbb{B}_f(\rho) := \left\{ \mathbf{b} \in \mathbb{N}^R \mid b(U) \leqslant f(U) \; \forall U \subseteq R, \; b(R) = \rho \right\}.$$

For an extensive treatment of polymatroids and submodular functions, see Schrijver [93, Chapters 44–49].

Example 2.2. Fig. 2.2 depicts a graphical example of a polymatroid.

Example 2.3. Matroids are special cases of polymatroids. For an extensive treatment of matroid theory, see Schrijver [93, Chapters 39–43]. There are several equivalent definitions, and in this thesis we will work with the following.

Definition 2.4. A matroid is a tuple $(R, \mathcal{I})$ where R is a ground set and $\mathcal{I} \subseteq 2^R$ is a family of subsets of R, that satisfies the following properties:

1. Non-empty: $\emptyset \in \mathcal{I}$.
2. Hereditary: For each $A \subseteq B \subseteq R$ it holds that $B \in \mathcal{I}$ implies $A \in \mathcal{I}$.
3. Augmentation: For each $A, B \in \mathcal{I}$ with $|A| > |B|$, there exists an $r \in A$ such that $B \cup \{r\} \in \mathcal{I}$.

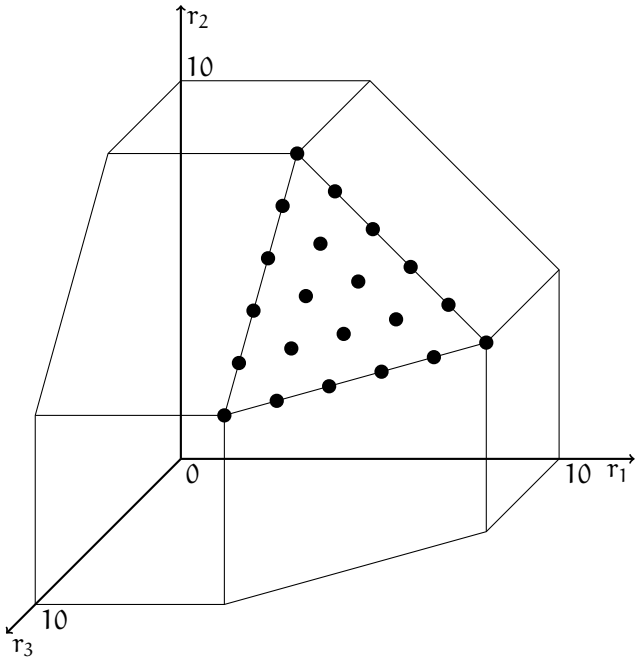

Figure 2.2: A graphical representation of the polymatroid over $R = \{r_1, r_2, r_3\}$ determined by the function $f : 2^R \to \mathbb{N}^R$ defined by $f(\emptyset) = 0$ and $f(S) = 5|S| + 5$ otherwise. This depicts the convex hull of all points in $\mathbb{P}_f$ and the black dots are the elements of $\mathbb{B}_f(\rho)$, where $\rho = f(R) = 20$.

The elements of $\mathcal{I}$ are called the *independent sets* of the resources. An inclusionwise maximal independent set is called a *basis*. Because of the augmentation property, all bases have the same cardinality, which is called the *rank* of the matroid. Every matroid has a *rank function* $f : 2^R \to \mathbb{N}$ that maps every subset S to the maximum size of an independent set in S. Like for polymatroids, this rank function is submodular, monotone and normalized.

An example of a matroid is the graphic matroid, which consists of all forests in an undirected graph. Formally, let $G = (V, E)$ be a given undirected graph. The ground set of the matroid is the set of edges E. A subset A of the edges is called independent if and only if A does not contain any cycles. The bases of this matroid are all full spanning forests of G (if G consists of only one connected component, these are all spanning trees) and the rank of this matroid is $|V| - c$, where c is the number of connected components of G.

Two other examples are the k-uniform matroid, whose independent sets are simply all sets of cardinality at most k, and the vector matroid, whose independent sets are all linearly independent subsets of a given subset of some vector space. The properties of Definition 2.4 are easy to verify.

Matroids are special cases of polymatroids in the following sense. Let (R, f) be an integral polymatroid. If f is 1-Lipschitz, it is the rank function of a matroid over R and the elements of $\mathbb{P}_f$ are its independent sets.

2.2.3 *Polymatroid Congestion Models*

To obtain a polymatroid congestion model, we associate an integral polymatroid rank function f_i with every player $i \in N$, we denote $f_i(R) = \rho_i$ and let $\rho = \sum_{i \in N} \rho_i$. We denote the Minkowski sum of the polyhedra by $\mathbb{B}_f(\rho) := \sum_{i \in N} \mathbb{B}_{f_i}(\rho_i)$ and from [93, Theorem 44.6] we know that this is also an integral polymatroid base polyhedron with rank function $f = \sum_{i \in N} f_i$.

We define a *polymatroid congestion model* $\mathcal{M} = (N, R, (\mathbb{B}_{f_i}(\rho_i))_{i \in N}, (c_r)_{r \in R})$ as a congestion model, where every player $i \in N$ chooses an element $\mathbf{b}^i \in \mathbb{B}_{f_i}(\rho_i)$. We study the problem of computing an *optimal strategy profile* minimizing the social cost., i.e., in this polymatroid congestion model we want to find a vector $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\rho)$ that minimizes the normalized monotone cost function $C(\mathbf{b}) = \sum_{r \in R} c_r(\mathbf{b})$. Formally, we arrive at the following combinatorial optimization problem.

Problem 2.5. *Find Optimal Strategy Profile*

INPUT: A polymatroid congestion model $\mathcal{M} = (N, R, (\mathbb{B}_{f_i}(\rho_i))_{i \in N}, (c_r)_{r \in R})$.

OBJECTIVE: Find $\min_{\mathbf{b} \in \mathbb{B}_f(\rho)} C(\mathbf{b})$.

Example 2.6. In spanning tree congestion games, for all $i \in N$, $f_i(U)$ equals the maximum amount of edges from $U \subseteq R_i$ such that these edges do not constitute a cycle. Note that $f_i(R_i) = |V_i| - 1$. Furthermore, $f(U)$ equals the maximum amount of copies of edges from $U \subseteq R$ such that these copies can be decomposed into n forests on $G[U]$, the graph induced by U, implying $\rho = f(R) = \sum_{i=1}^{n} |V_i| - n$. $\mathbb{P}_{f_i}$ and $\mathbb{B}_{f_i}$ are the integral polyhedra corresponding to the incidence vectors of all forests respectively all spanning trees in G_i for all $i \in N$. $\mathbb{P}_f$ and $\mathbb{B}_f$ correspond to the collection of all sets of edges that can be decomposed into n forests respectively into n spanning trees, one in each G_i.

Example 2.7. If players choose subsets rather than multisets of the resources, we obtain a *matroid congestion model* in which every player has a player-specific matroid $M_i = (R_i, \mathcal{I}_i)$ and needs to select a basis B_i from the set of bases $\mathcal{B}_i$, which is the set of the elements of $\mathcal{I}_i \subseteq 2^R$ with maximal cardinality.

We remark that polymatroid congestion games were recently introduced by Harks et al. [53]. In contrast to their model, we do not allow that cost functions are player-specific, but we do allow general non-decreasing cost functions instead of convex cost functions.

2.3 A LOGARITHMIC APPROXIMATION

Before we present our approximation algorithm, we derive the following hardness result. The reduction is based on [52, Theorem 7.1], where the hardness of computing an optimal strategy profile for singleton congestion games is shown.

Lemma 2.8. *Problem 2.5 is strongly NP-complete and there are no* $c \log \rho$ *approximation algorithms for any* $c < 1$*, unless NP* $\subseteq$ *TIME*$(n^{\mathcal{O}(\log \log n)})$.

Proof. We reduce from the HITTING SET problem. An instance of HITTING SET consists of a set $\mathcal{C}$ of n subsets $(C_i)_{i \in N}$ over a finite ground set of elements E. A hitting set is a subset $F \subseteq E$ such that F contains at least one element of every $C_i \in \mathcal{C}$. The goal is to select a minimum cardinality hitting set.

Given an instance $(E, \mathcal{C})$ of HITTING SET, we construct a polymatroid congestion game $\mathcal{M}$ as follows. First construct the game $\mathcal{M}'$ by identifying E with R, and defining the submodular, monotone, normalized function f_i for all $i \in N$ as follows: $f_i(S) = 1$ if $S \cap C_i \neq \emptyset$ and $f_i(S) = 0$ otherwise. We let $c_r(0) = 0$ and $c_r(j) = 1$ for all $r \in R$ and $j \in N$.

From this game $\mathcal{M}'$, we construct another polymatroid congestion game $\mathcal{M}$ with one player, whose integral polymatroid base polyhedron is the Minkowski sum of the polyhedra in $\mathcal{M}'$, i.e. $\mathbb{B}_f(\rho) = \sum_{i \in N} \mathbb{B}_{f_i}(\rho_i)$. Note that $\rho_1 = \rho = n$.

By construction, there is a hitting set of cardinality k if and only if there is a vector $\mathbf{b} \in \mathbb{B}_f(\rho)$ with $C(\mathbf{b}) \leqslant k$ in $\mathcal{M}$. Therefore, if we can approximate the social optimum in $\mathcal{M}$ within a factor of $c \log(\rho)$ for some constant c, we can approximate the HITTING SET instance within $c \log(n)$. The lemma now follows from the fact that the HITTING SET problem is equivalent to

the SET COVER problem [8], for which there are no polynomial time $c \log n$-approximation algorithm for any $c < 1$ unless $NP \subseteq TIME(n^{O(\log\log n)})$ [37]. ■

Now we present our algorithm (see Algorithm 2.1). Intuitively, for every player i we want to distribute ρ_i units over the resources such that the resulting multiset of resources corresponds to a vector $\mathbf{b}^i \in \mathbb{B}_{f_i}(\rho_i)$. Overall, this leads to a distribution of ρ units over the resources such that $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\rho)$. The idea is to incrementally increase the number of units distributed over the resources in a greedy fashion. Initially, we start with an empty distribution, i.e., the load on every resource is zero. Then, we will iteratively increase the load on some resource $r \in R$ to some desired *target load* denoted by $t_r \in \mathbb{N}$ (initially $t_r = 0$ for all $r \in R$). To make sure that the sum of the target loads does not exceed ρ, we set upper bounds $t_r^{\max}$ on the target loads that are initially set to $t_r^{\max} = \rho$ for all $r \in R$. In every iteration, we select a resource $r \in R$ and a target load between $t_r + 1$ and $t_r^{\max}$ minimizing the *cost per unit*, which is defined via the following function:

$$h : R \times ([t_r + 1, t_r^{\max}] \cap \mathbb{N}) \to \mathbb{R}; \quad h(r, j) = \frac{c_r(j) - c_r(t_r)}{j - t_r}. \tag{2.1}$$

Let (r^*, j^*) be a minimizer of h. To check whether or not it is possible to distribute $j^* - t_{r^*}$ additional units on resource r^*, or equivalently, to increase the target load t_{r^*} to j^*, we call a *strategy covering oracle* denoted by $\mathfrak{O}(R, \mathbb{B}_f(\rho), (t_r)_{r\in R})$ (cf. Definition 2.9). This oracle checks if there is a strategy profile covering the current target loads, that is, if there is a $\mathbf{b} = \mathbf{b}^1 + \ldots + \mathbf{b}^n \in \mathbb{B}_f(\rho)$ satisfying $b_r \geqslant t_r$ for all $r \in R$. It is similar to the strategy packing oracle from [55].

Throughout this chapter, for two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{N}^m$, we use $\mathbf{x} \geqslant \mathbf{y}$ in the sense $x_i \geqslant y_i$ for all i, and for convenience we write $x(\{1, \ldots, i\}) = \sum_{j=1}^{i} x_j$.

Definition 2.9. *Strategy covering oracle* $\mathfrak{O}(R, \mathbb{B}_f(\rho), (t_r)_{r\in R})$

INPUT: A finite set R with target loads $t_r \in \mathbb{N}$ for all $r \in R$ and an integral polyhedron $\mathbb{B}_f(\rho) \subseteq \mathbb{N}^R$.

OUTPUT: A vector $\mathbf{b} \in \mathbb{B}_f(\rho)$ such that $\mathbf{b} \geqslant \mathbf{t}$, or the information that no such vector exists.

If the answer of the oracle is negative, it is not possible to increase the load on resource r^* to j^*. We set t_{r^*} back to its old value and update $t_{r^*}^{\max}$

to $j^* - 1$, because this target load j^* was too high for this resource. On the other hand, if the answer is affirmative, we keep the increased target load j^* on r^*, and for all $r \in R$ we update $t_r^{\max}$ to $\min\left\{t_r^{\max}, t_r + \rho - \sum_{r' \in R} t_{r'}\right\}$ to avoid target loads whose sum exceeds ρ. We also update the values for h according to Eq. (2.1). We then continue this procedure by finding a new minimizer of h and calling the oracle again. Note that in every iteration we increase the lower bound t_{r^*} or decrease the upper bound $t_{r^*}^{\max}$, thus, the domain of h becomes strictly smaller in every iteration. Indeed, once we have $\sum_{r \in R} t_r = \rho$ and the oracle outputs a vector $\mathbf{b} \in \mathbb{B}_f(\rho)$ that meets these target loads, the algorithm terminates. The vector $\mathbf{b}$ can be decomposed into a vector for every player. For a formal description see Algorithm 2.1. For an illustration see Fig. 2.3.

Algorithm 2.1: Algorithm for the polymatroid congestion model and game

Input: A polymatroid congestion model
$\mathcal{M} = (N, R, (\mathbb{B}_{f_i}(\rho_i))_{i \in N}, (c_r)_{r \in R})$
Output: A vector $\mathbf{b} \in \mathbb{B}_f(\rho)$

$\mathbf{b} \leftarrow \mathbf{0}$;
$t_r \leftarrow 0$ and $t_r^{\max} \leftarrow \rho$ for all $r \in R$;
while $\sum_{r \in R} t_r < \rho$ **do**
  Choose $(r^*, j^*) \in \operatorname{argmin}_{r \in R, j \in [t_r + 1, t_r^{\max}]} h(r, j)$;
  $\mathrm{temp} \leftarrow t_{r^*}$;
  $t_{r^*} \leftarrow j^*$;
  if $\mathfrak{O}(R, \mathbb{B}_f(\rho), (t_r)_{r \in R}) = \mathbf{b}'$ **then**
    $\mathbf{b} \leftarrow \mathbf{b}'$;
    $t_r^{\max} \leftarrow \min\left\{t_r^{\max}, t_r + \rho - \sum_{r' \in R} t_{r'}\right\}$ for all $r \in R$;
    Update $h(r^*, j)$ for all $j \in [t_{r^*} + 1, t_{r^*}^{\max}]$ as in Eq. (2.1);
  else
    $t_{r^*} \leftarrow \mathrm{temp}$;
    $t_{r^*}^{\max} \leftarrow j^* - 1$;
return $\mathbf{b}$;

The following lemma shows that the strategy covering oracle can be implemented in linear oracle time for our polymatroid congestion model.

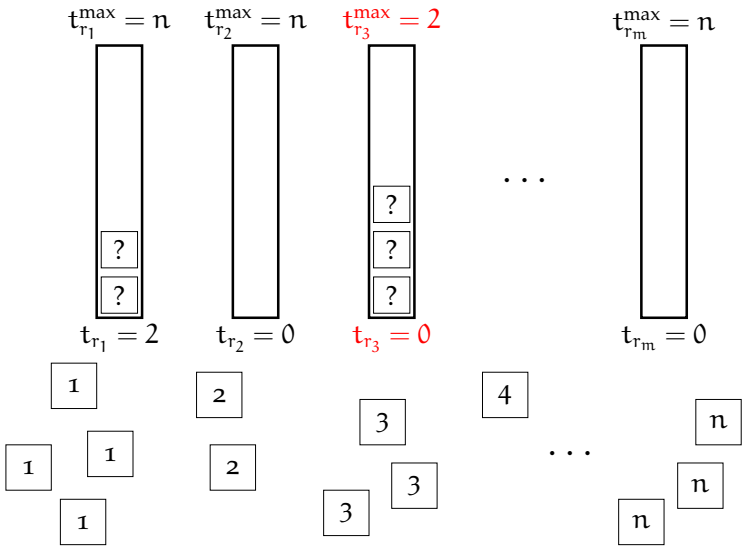

Figure 2.3: An illustration of a possible iteration of Algorithm 2.1. The squares are units each player needs to distribute over the resources represented by the rectangles. In this iteration, $(r_3, 3)$ was the chosen minimum cost per unit, but the oracle returns that these target loads are infeasible. Hence, $t^{max}_{r_3}$ is set to 2 and t_{r_3} is reset to its previous value 0.

Lemma 2.10. *If $\mathbb{B}_f(\rho)$ is an integral polymatroid base polyhedron, then the strategy covering oracle can be implemented in time $\mathcal{O}(m\,Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

Proof. See Algorithm 2.2 for a formal description, which is an adaptation of the polymatroid greedy algorithm (see e.g. [93, Theorem 44.3]).

There exists a vector $\mathbf{b} \in \mathbb{B}_f(\rho)$ such that $\mathbf{b} \geqslant \mathbf{t}$ if and only if $\mathbf{t} \in \mathbb{P}_f(\rho) = \sum_{i=1}^{n} \mathbb{P}_{f_i}(\rho_i)$. To check this membership, we start at $\mathbf{y} = \mathbf{o} \in \mathbb{P}_f(\rho)$ and iteratively increase y_i to t_{r_i}. Following the proof of [93, Theorem 44.3] we know that after iteration i, $\mathbf{y} \in \mathbb{P}_f(\rho)$ if and only if $y(\{1,\dots,i\}) \leqslant f(\{1,\dots,i\})$. After m iterations $\mathbf{y} = \mathbf{t}$, hence $\mathbf{t} \in \mathbb{P}_f(\rho)$ and the target loads are feasible. If at some point $y(\{1,\dots,i\}) > f(\{1,\dots,i\})$, the target loads are infeasible.

To extend the target loads to a strategy profile (i.e. a vector $\mathbf{b} \in \mathbb{B}_f(\rho)$), we define the function $g(U) = f(U) - t(U)$ for all $U \subseteq R$ (for the implementation of the algorithm we only need $g(\{1,\dots,i\})$ for all i). We claim, and this is easy to check, that g is an integral polymatroid rank function with $g(R) = \rho - t(R)$. Hence we can find a vector $\mathbf{z} \in \mathbb{B}_g(g(R))$ using the polymatroid greedy algorithm. Define $\mathbf{b} = \mathbf{t} + \mathbf{z}$. By construction

$b(U) = t(U) + z(U) \leqslant t(U) + f(U) - t(U) = f(U)$ for all $U \subseteq R$, so $\mathbf{b} \in \mathbb{P}_f(\rho)$. Similarly, $b(R) = \rho$ and hence $\mathbf{b} \in \mathbb{B}_f(\rho)$. As $\mathbf{z} \geqslant \mathbf{o}$ we have $\mathbf{b} \geqslant \mathbf{t}$ and the proof is complete.

Denoting the complexity of the value giving oracle for the function f by Q, the running time of both the membership check and the extension is $\mathcal{O}(m\,Q)$. ■

Algorithm 2.2: Strategy covering oracle for polymatroid congestion games

Input: An integral polymatroid base polyhedron $\mathbb{B}_f(\rho)$ and a vector t
Output: A vector $\mathbf{b} \in \mathbb{B}_f(\rho)$ such that $\mathbf{b} \geqslant t$, or the information that no such vector exists.

```
1  y ← o;
2  for i = 1 to m do
3  |   y_i ← t_{r_i};
4  |   if y({1,...,i}) > f({1,...,i}) then
5  |   |   return "Infeasible target loads"
6  z ← o and g(∅) ← 0;
7  for i = 1 to m do
8  |   g({1,...,i}) ← f({1,...,i}) − t({1,...,i});
9  |   z_i ← g({1,...,i}) − g({1,...,i−1});
10 return t + z;
```

We now prove the following theorem.

Theorem 2.11. *Algorithm 2.1 is an H_ρ-approximation algorithm for Problem 2.5 that runs in time $\mathcal{O}(m^2\,\rho^2\,Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

We show this using the following two lemmata. For the proof of the next lemma, we need the following well-known property of polymatroids (cf. [40, Theorem 3.27] and the text following [93, Theorem 44.6]).

Property 2.12. *Let $\mathbb{P}_f(\rho)$ be an integral polymatroid polyhedron. For any two vectors $\mathbf{b}, \mathbf{b}' \in \mathbb{P}_f(\rho)$ and any element $r \in R$ with $b_r > b'_r$, there exists an element $r' \in R$ with $b'_{r'} > b_{r'}$ such that $\mathbf{b} - \chi_r + \chi_{r'} \in \mathbb{P}_f(\rho)$. Here, χ_r is the m-dimensional unit vector corresponding to resource $r \in R$.*

Lemma 2.13. *Let* $\mathbf{b}$ *be the output of Algorithm 2.1 and let* $\mathbf{b}^*$ *be the set of bases minimizing* $C(\mathbf{b}^*)$. *Then* $C(\mathbf{b}) \leqslant H_\rho C(\mathbf{b}^*)$.

Proof. Consider iteration k of Algorithm 2.1, indexed by the while loop. We denote the target load on a resource r at the start of this iteration by t_r^k. In this iteration, we update every target load to t_r^{k+1}, which is only different from t_r^k for one resource r^* (in our analysis we may disregard iterations in which we do not increase t_{r^*} but decrease $t_{r^*}^{\max}$). Denote the increase in the target loads in iteration j by $n_j = \sum_{r\in R} t_r^{j+1} - t_r^j$ and denote the remaining units to distribute over the resources at the beginning of iteration k by $\bar{n}_k = \rho - \sum_{j=1}^{k-1} n_j$. Denote the strategy profile at the start of this iteration, returned by the oracle in the previous iteration, by $\bar{\mathbf{b}}_k = \bar{\mathbf{b}}_k^1 + \ldots + \bar{\mathbf{b}}_k^n$ (initially $\bar{\mathbf{b}}_0^i = \mathbf{o}$ for all i).

Claim 2.14. There exists a vector $\hat{\mathbf{b}} = \hat{\mathbf{b}}^1 + \ldots + \hat{\mathbf{b}}^n$ such that $t_r^k \leqslant \hat{b}_r \leqslant \max\{b_r^*, t_r^k\}$ for all $r \in R$.

Before proving this claim, we show how the lemma follows from it. The analysis is based on the cost-effectiveness of our greedy choice in the algorithm, e.g. as in [97, Chapter 2]. Consider the quantity $\Delta_k = C(\hat{\mathbf{b}}) - \sum_{r\in R} c_r(t_r^k)$. Using monotonicity of the c_r, we obtain

$$\Delta_k \leqslant \sum_{r\in R} \left(\max\left\{c_r(\mathbf{b}^*), c_r(t_r^k)\right\} - c_r(t_r^k)\right) \leqslant C(\mathbf{b}^*)\,.$$

Note that $(r^*, t_{r^*}^{k+1})$ is a minimizer of the preliminary cost-per-unit in iteration k, and in particular, the sequence of per-unit cost of the load increments in Algorithm 2.1 is non-decreasing. Also note that Δ_k is the cost of one possible extension from the target loads in iteration k to a strategy profile. Hence, the average cost of one resource in Δ_k is at least the average increase in cost of one resource in iteration k, and we have

$$\frac{\sum_{r\in R} c_r(t_r^{k+1}) - c_r(t_r^k)}{n_k} = \frac{c_{r^*}(t_{r^*}^{k+1}) - c_{r^*}(t_{r^*}^k)}{n_k} \leqslant \frac{\Delta_k}{\bar{n}_k} \leqslant \frac{C(\mathbf{b}^*)}{\bar{n}_k}\,.$$

Using that $\bar{\mathbf{b}}_0 = \mathbf{o}$ and $c_r(0) = 0$ for all $r \in R$, this yields

$$\begin{aligned} C(\mathbf{b}) &= \sum_k \left(\sum_{r\in R} c_r(t_r^{k+1}) - c_r(t_r^k)\right) \leqslant \sum_k \frac{n_k}{\bar{n}_k} C(\mathbf{b}^*) \\ &= \sum_k \frac{n_k}{\sum_{l\geqslant k} n_l} C(\mathbf{b}^*) = \sum_k \sum_{j=1}^{n_k} \frac{1}{\sum_{l\geqslant k} n_l} C(\mathbf{b}^*) \end{aligned}$$

$$\leqslant \sum_{k} \sum_{j=1}^{n_k} \frac{1}{\sum_{l \geqslant k} n_l - j + 1} C(\mathbf{b}^*) = H_\rho C(\mathbf{b}^*) .$$

It remains to prove Claim 2.14.

Proof of Claim 2.14. Set $\hat{\mathbf{b}} = \bar{\mathbf{b}}_k$ and consider a resource r such that $\hat{b}_r > \max\{b^*_r, t^k_r\}$. We call such a resource *overloaded*. Then by Property 2.12 for $\mathbb{P}_f(\rho)$ there exists a resource r' with $b^*_{r'} > \hat{b}_{r'}$ such that $\hat{\mathbf{b}}' = \hat{\mathbf{b}} - \chi_r + \chi_{r'} \in \mathbb{P}_f(\rho)$. Replace $\hat{\mathbf{b}}$ by $\hat{\mathbf{b}}'$.

We continue this procedure until there is no overloaded resource anymore. Indeed, as $b^*_{r'} > \hat{b}_{r'}$ we know r' is not overloaded in $\hat{\mathbf{b}}'$. Hence, the total overload $\sum_{r \in R} \max\{\hat{\mathbf{b}}_r - \max\{b^*_r, t^k_r\}, 0\}$ becomes strictly smaller after every replacement. As this quantity is non-negative, this procedure ends and at some point there will not be an overloaded resource anymore. Because it is possible to cover the target loads t^k_r, we know $t^k_r \leqslant \hat{b}_r \leqslant \max\{b^*_r, t^k_r\}$ for all $r \in R$. ∎

Lemma 2.15. *Algorithm 2.1 runs in time $\mathcal{O}(m^2 \rho^2 Q)$, where Q is the complexity of the value giving oracle that returns $f(U)$ for any $U \subseteq R$.*

Proof. The number of iterations is upper bounded by $m\rho$. In every iteration we find the minimizer of at most $m\rho$ values and then we call the strategy covering oracle, which runs in time $\mathcal{O}(mQ)$ by Lemma 2.10. So the total running time of Algorithm 2.1 is $\mathcal{O}(m^2 \rho^2 Q + m^2 \rho Q) = \mathcal{O}(m^2 \rho^2 Q)$. ∎

This concludes the proof of Theorem 2.11.

2.4 CONCLUDING REMARKS

This chapter presents an H_ρ-approximation algorithm for polymatroid congestion models. This approximation guarantee is best possible up to a constant factor. The algorithm runs in polynomial time if we assume that ρ is polynomial in the input size, which is the case for special cases such as matroids and graphical polymatroids (arborescences). In particular, for arborescences, the oracle can be implemented in time $\mathcal{O}(|V|^2 m \log(|V|^2/m))$ [42, Theorem 7.1] (where $|V|$ is the number of vertices in the graph underlying this graphical polymatroid), and we obtain a running time of $\mathcal{O}(m^3 \rho^2 |V|^2 \log(|V|^2/m))$. However, for general polymatroids, the running time is pseudopolynomial.

Consider the special case of matroid congestion games as introduced in Example 2.7. In these games, players choose subsets rather than multisets of the resources and therefore $\rho \leqslant n\,m$. The number of iterations is upper-bounded by $n\,m$ and in every iteration we find the minimizer of at most $n\,m$ values. Using the idea of Cunningham [29], Harks et al [55] proved that their strategy packing oracle can be implemented in time $\mathcal{O}(n^{1.5}\,\rho\,Q)$, where Q is the maximum complexity of the independence oracles of the matroids. A similar proof also works for our strategy covering oracle. The running time of the algorithm for matroid congestion games is thus $\mathcal{O}(n^2\,m^2 + n^{2.5}\,m\,\rho\,Q) = \mathcal{O}(n^{3.5}\,m^2\,Q)$.

However, a comment on the tightness of our approximation guarantee for matroid congestion games is in order. By a reformulation of [52, Theorem 7.1], these games are $c \log n$-inapproximable for any $c < 1$, but this cannot be strengthened to a $c \log \rho$ hardness result as in Lemma 2.8. Thus, as $\log \rho \leqslant \log n + \log m$, the gap between our approximation guarantee and the hardness result is a constant factor and an additive error of $\log m$. This is a constant gap if $m = \mathrm{poly}(n)$. However, if $m \neq \mathrm{poly}(n)$, the approximation complexity of the problem is yet to be settled exactly.

3

HIGH MULTIPLICITY SCHEDULING

3.1 INTRODUCTION

In the current competitive economy, companies need to be aware of multiple objectives such as decreasing costs and enhancing customer service. Among the core activities of many companies and supply chains are mechanisms to match supply with demand, to prevent stock-outs and to cut back unnecessary overhead costs. Production companies are required to conduct extensive research into cost reduction to remain competitive within the market. Consequently, a lot of interest has been shown in problems within the area of operations management.

One of these well-studied problems in operations management is the CAPACITATED LOT-SIZING PROBLEM, where one machine needs to produce a set of products to minimize average holding and setup costs. In this problem, the ongoing production of a product can be represented as the repeated scheduling of a single job on the machine, enabling a highly compact encoding of the input of the problem known as *high multiplicity scheduling*. Jobs in the high multiplicity setting are represented by a single job description with a multiplicity, representing the number of individual jobs. It is different from the conventional scheduling setting where every single job, even though identical to many others, is given as part of the problem input. Obviously, the input length of the traditional setting can be exponentially larger than the length of the high multiplicity input, allowing for exponentially slower algorithms. Furthermore, due to the compact encoding of the input, the optimal schedule can have a superpolynomial length, even for very restricted cases e.g. with only 1 or 2 products. Consequently, finding a polynomially sized certificate for these types of problems can be hard in itself.

Not only from the computational complexity point of view, the reasonability of conventional encoding is questionable for practical high multiplicity problems. For many companies, the high multiplicity encoding is a natural way to provide input from real-world data, especially if thousands of jobs are identical. The high multiplicity scheduling setting is found in numerous practical applications. The research in this chapter was inspired by

one of these practical applications; a multinational textile company posed the problem of finding the optimal cycle length for their production, of only three types of lycra in extremely large quantities on a single machine.

In this chapter we study an extended version of the aforementioned real-life problem, the CAPACITATED LOT-SIZING PROBLEM with *sequence-dependent setup costs*. In this problem we have a single machine that is capable of producing a single product at any given time and a set of products that need to be produced. Each product is associated with a demand rate, a maximum production rate and inventory holding costs per unit. The objective is to find a cyclic schedule such that the demand of every product is met, minimizing the average costs per cycle. For any schedule, sequence-dependent setup costs referred to as *sequencing costs* are incurred each time the machine switches production between two different products. Moreover, input is provided under high multiplicity encoding.

We show NP-hardness of the problem and largely characterize optimal solutions by proving a number of structural properties which will be of great use for the design of algorithms for the problem. Furthermore, assuming a fixed number of products, we develop an approximation algorithm which slightly perturbs the input instance to get a polynomial running time and polynomial size of the output schedule.

RELATED WORK The earliest research on problems with high multiplicity encoding dates back to the sixties; see e.g. Rothkopf [87] who considers the TRAVELLING SALESPERSON PROBLEM with multiple visits to cities. Madigan [78] studies a variant of our problem where setup times are introduced, setup costs do not depend on the sequence, and holding costs are product-independent. He proposes an elegant heuristic for the problem and compares it to the results previously published in the literature. Goyal [46] studies the variant of the problem posed by Madigan where no setup times are involved, and solves the problem to optimality for a fixed time horizon. Boctor [14] extends the model to incorporate product-dependent holding costs and setup times, and considers an infinite time horizon. He presents an exact algorithm for the case of two products.

Only in 1991, Hochbaum and Shamir [60] coined the term *high multiplicity* and underlined the added complexity of such encodings. They study single machine high multiplicity scheduling problems with different objective functions, and construct algorithms that are strongly polynomial in the number of types of jobs. At the same time, Narro Lopez and Kingsman [84]

discuss basic solution approaches to high multiplicity scheduling problems and assess their quality and use in practice.

Most papers on high multiplicity scheduling consider discrete variants, in which time and/or quantities are discretized into units. There has also been some work considering the continuous setting, in which production can start and stop at any time, e.g. with fluids. Bertsimas, Gamarnik, and Sethuraman [11] consider the high multiplicity job-shop problem without sequencing costs, and use this continuous setting as a relaxation for the original discrete job-shop problem. They round an optimal solution for the fluid problem to an asymptotically optimal solution for the discrete problem, and provide some computational experiments. In another work on the continuous setting, Haase [48] discusses a problem very closely related to ours, where production rates are fixed. He proposes a local-search heuristic and evaluates it by comparing it to optimal solutions for small instances. Haase and Kimms [49] consider the same problem and, by making additional assumptions on the input instances, solve the problem to optimality. They present a Mixed Integer Programming formulation for their model and a fast enumeration scheme, which they evaluate by a computational study.

Incorporating sequencing costs substantially adds complexity akin to the TRAVELLING SALESPERSON PROBLEM. The techniques we are using in this chapter are closely related to the techniques used in classical single multiplicity scheduling. For instance, Clifford and Posner [25] provide lower bounds and use these to develop heuristics for minimizing tardiness. They extend the problem to parallel, uniform and unrelated machines in Clifford and Posner [26], where their objective is to minimize the makespan or the sum of completion times in either the preemptive, or the non-preemptive variant of the problem. They prove NP-hardness, develop polynomial time and pseudopolynomial time algorithms for special cases, and present heuristics. Filippi and Romanin-Jacur [38] continue their work and present a two-stage approach, in which they first fix most jobs in partial schedules and then solve the residual problem.

Brauner et al. [16] provide a detailed framework for the complexity analysis of high multiplicity scheduling problems. We refer the reader to this paper for an excellent survey of related work in this field. They extend their framework in Brauner et al. [17].

3.2 THE MODEL

We model the general problem for multiple products as follows. We have a single machine that can produce a single type of product at any given time and we are given a set of products $J = \{1, \dots, n\}$. For each product $i \in J$, let p_i be its maximum production rate, i.e., the maximum number of units produced per time unit. Similarly, let d_i be its demand rate and h_i its holding costs per time unit. Furthermore, we are given sequencing costs $s_{i,j}$ that need to be paid when we switch from producing product i to producing product j. The problem we consider is to find an optimal cyclic schedule S^* that minimizes the average costs per unit of time $\bar{c}(S^*)$. Note that for each product i, the rates d_i and p_i and costs h_i are assumed to be constant over time and positive. Observe that the input is very compact. Let m be the largest number in the input, then the input size is $\mathcal{O}(n \log m)$, where n is typically a small number, or even a constant.

We distinguish three variants of the problem: The *continuous* case, in which the machine can switch production at any time; the *discrete* case, in which the machine can switch production only at the end of a fixed unit of time (e.g. a day) and produces some product i at a single rate $r_i \leqslant p_i$ during each unit of time; and the *fixed* case, in which the machine can switch production only at the end of a fixed unit of time, and when the machine produces some product i, a full amount of p_i needs to be produced (in the former two cases, we can lower production rates). Furthermore, in the fixed case we assume integrality of the production and demand rates. Without loss of generality, in all variants we assume $d_i, p_i, h_i, s_{ij} \geqslant 1$ for all $i, j \in J$.

We denote by LSP(A,n) with $A \in \{C, D, F\}, n \in \mathbb{N}$ the lot-sizing problem of scheduling n products in the continuous, discrete or fixed setting respectively. Let $\pi_i^{[a,b]}$ denote the produced amount of product i during time interval $[a, b]$. Let $\pi_i^t = \pi_i^{[t,t+1]}$. Let x_i^t be an indicator function denoting whether product i is produced during time interval $[t, t+1]$. Let q_i^t denote the stock level for product i at time t. We explicitly refer to the stock of product i at time t in a schedule S as $q_i^t(S)$.

Finally, let $H(S)$ denote the total holding costs and $W(S)$ the total switching costs of a schedule S, and $c(S) = H(S) + W(S)$ denote the total costs of S. Denote the average costs of a cyclic schedule S of cycle length ℓ by $\bar{c}(S) = \bar{H}(S) + \bar{W}(S)$, where $\bar{H}(S) = H(S)/\ell$ and $\bar{W}(S) = W(S)/\ell$.

Formally, we arrive at the following problem.

Problem 3.1. *High Multiplicity Scheduling problem*

INPUT. For each product i, a demand rate $d_i \geqslant 1$, a maximum production rate $p_i \geqslant 1$, and inventory holding costs $h_i \geqslant 1$ are given. Sequencing costs $s_{i,j} \geqslant 1$ are given for every pair of products.

TASK. Find a cyclic schedule S which minimizes the average costs per unit of time, $\bar{c}(S)$.

We represent a cyclic schedule of length ℓ as a sequence

$$[t_0, t_1]^{r_0}_{i_0}, [t_1, t_2]^{r_1}_{i_1}, \ldots, [t_s, \ell]^{r_s}_{i_s},$$

where $r_\varphi \leqslant p_{i_\varphi}$ is a production rate of *phase* $\varphi = 0, \ldots, s$, i_φ is the product produced in that phase, and $[t_\varphi, t_{\varphi+1}]$ is the time interval such that no two consecutive phases share the same r_φ and i_φ. A maximal sequence of consecutive phases of the same product $i_\varphi \in J$ is called a *production period*, denoted by $[t_\varphi, t_{\varphi+1}]_{i_\varphi}$. The complete sequence of phases is called the (cyclic) schedule, and we call a schedule a *simple cycle* if there is exactly one production period for each product.

Example 3.2. Consider the following instance of LSP(D,2). Product 1 has a maximum production rate of $p_1 = 5$ and a damand rate of $d_1 = 2$, and product 2 has a maximum production rate of $p_2 = 7$ and a demand rate of $d_2 = 3$. Fig. 3.1 depicts a feasible cyclical schedule S for this instance, which we denote as $S = [0, 2]^5_1, [2, 3]^1_2, [3, 5]^7_2$. Product 1 has one production period from time 0 to 2, consisting of one phase during which it is produced at rate 5. Product 2 has one production period from time 2 to 5, consisting of two phases. During the first phase from time 2 to 3, it is produced at rate 1, and during the second phase from time 3 to 5, it is produced at rate 7. Note that S is a simple cycle.

Suppose $h_1 = 1$, $h_2 = 2$, $s_{12} = 3$ and $s_{21} = 4$. By computing the area under the stock curves in Fig. 3.1 and multiplying it with the respective holding costs per unit, we see that the total holding costs of products 1 and 2 are equal to 15 and 38 respectively. Therefore, $H(S) = 53$. As $W(S) = 7$, we get $c(S) = 60$ and hence $\bar{c}(S) = 12$.

3.3 STRUCTURAL PROPERTIES OF OPTIMAL SOLUTIONS

We now prove some structural properties of optimal schedules of the problem. We show that all variants are NP-hard, even when we restrict ourselves

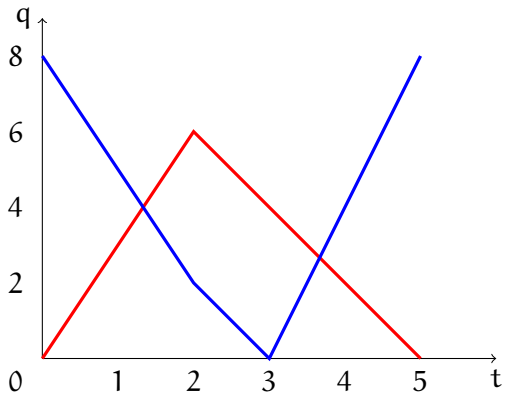

Figure 3.1: A feasible cyclic schedule for Example 3.2. The red and blue line indicate the stock level of product 1 respectively product 2 over time.

to unit demand rates and unit holding costs. Next, we derive a simple necessary and sufficient condition for the existence of a feasible cyclic schedule. Furthermore, we characterize the form of production for the continuous and discrete cases. Also, we show that there is no idle time in an optimal schedule and that every product has at least one point during the schedule where its stock level is zero. Finally, in the last subsection, we present a lower bound on the objective value for the continuous case, and an upper bound on the objective value and the maximum stock level for the discrete case. We use these bounds in the approximation scheme.

3.3.1 *Problem complexity*

The following lemma follows directly from a reduction from the TRAVELLING SALESPERSON PROBLEM (TSP), cf. Problem 1.2.

Lemma 3.3 (Complexity)**.** *All three variants of the lot-sizing problem are strongly NP-hard.*

Proof. We prove NP-hardness for both the discrete and fixed cases by a reduction from the TRAVELLING SALESPERSON PROBLEM (TSP). Consider a TSP instance $I = \{G = (V, E), [c_{i,j}]_{V\times V}\}$. From I we construct an instance $I' = \{J, (d_i, p_i, h_i)_{i\in J}, [s_{i,j}]_{J\times J}\}$ of the LOT-SIZING PROBLEM as follows.

Identify J with V and let $s_{i,j} = c_{i,j}$ for each $i, j \in J$. Let $d_i = 1$, $p_i = |V| = n$, and $h_i = h = \sum_{j,k} s_{j,k} + 1$ for each $i \in J$. Additionally, let $W_{\max} =$

$\sum_{i,j} s_{i,j}$ and $W_{\min} = n \times \min_{i,j} s_{i,j}$. Note that for every feasible schedule S, we have sequencing costs $W(S)$ such that $W(S) \geqslant W_{\min}$. Moreover, for all simple cycles S we have $W(S) \leqslant W_{\max}$. We claim that there exists a TSP tour of length at most B if and only if the corresponding instance of the lot sizing problem admits a solution of total cost at most $hn(n-1)/2 + B/n$.

Clearly, since the total demand and production rates match each other, the stock level is constant over time. Every simple cycle of length n, using the same order of products, can be realised with average holding costs $\bar{H} = hn(n-1)/2$ and average sequencing costs $W_{\min}/n \leqslant \bar{W} \leqslant W_{\max}/n$. In fact, this schedule is minimum regarding the holding costs.

Let S' be a feasible non-simple cycle of length $\ell' > n$ with total costs $c(S') = H(S') + W(S')$. Note that there is a product of which two consecutive production periods are separated by at least $n+1$ time units. Hence, we need at least one additional unit of that product in stock and thus $H(S') \geqslant h\ell' n(n-1)/2 + h\ell'$. Thus, for every minimal simple cycle S, since $W(S) \leqslant W_{\max} < h$, we have that the average costs of S' are $\bar{c}(S') \geqslant H(S')/\ell' > \bar{c}(S)$. Observe that the value of $\bar{H}(S)$ is the same for every minimal simple cycle, and therefore the optimal solution to I is the minimal simple cycle which minimizes $W(S)$.

Let σ be a sequence of visits in the TSP instance with costs B. Producing each product for 1 time unit with the same sequence as σ is a feasible solution for the lot sizing problem with costs $hn(n-1)/2 + B/n$. Conversely, let σ be a solution for the lot sizing problem with costs $hn(n-1)/2 + B/n$. This solution is a simple cycle, thus the production sequence is a tour with costs B. This proves the NP-hardness of the discrete and the fixed case.

We prove the continuous case by a similar reduction from the METRIC TSP. For an instance I of the Metric TSP, we let $J = V$ and $s_{i,j} = c_{i,j}$ for all $i, j \in J$. Let $d_i = 1$, $p_i = n$ and $h_i = 1$ for all $i \in J$.

Let σ be the optimal solution to I with costs $c(\sigma)$. Let S be any feasible schedule for the corresponding instance I' of the lot sizing problem and let the length of the schedule be ℓ. Let S^* be the simple cycle of length ℓ^* where the products are produced in the same order as in σ, with production time ℓ^*/n per product.

Since every product needs to be produced at least once in a feasible schedule and the triangle inequality holds for the sequencing costs, S^* is optimal with respect to the sequencing costs, i.e., $W(S^*) <= W(S)$. Note that compared to the discrete and fixed cases, the continuous case has a complication: We can choose ℓ^* arbitrarily small. By construction, every production

period in schedule S^* consists of one phase of length ℓ^*/n where the product is produced at rate $p_i = n$. Since $h_i = 1$, the total holding costs for every product i are given as

$$\int_0^{\ell^*/n} q_i^t \, dt + \int_{\ell^*/n}^{\ell} q_i^t \, dt = \frac{n-1}{2n}(\ell^*)^2 .$$

Thus, the total holding costs of S^* are $H(S^*) = (\ell^*)^2(n-1)/2$ and the average holding costs are $\bar{H}(S^*) = \ell^*(n-1)/2$. In particular, since holding costs decrease with the cycle length, we can choose ℓ^* such that $\bar{H}(S^*) \leqslant \bar{H}(S)$ and $\bar{c}(S^*) \leqslant \bar{c}(S)$. Thus we have that the optimal solution to I' is a simple cycle S^* using the sequence of σ.

The total average costs $c(\sigma)/\ell^* + \ell^*(n-1)/2$ are minimized with $\ell^* = \sqrt{2c(\phi)/(n-1)}$. Hence we have

$$c(\sigma) = W(S^*) = H(S^*) = \frac{n-1}{2}(\ell^*)^2 .$$

Now, σ is an optimal solution for I with costs $c(\sigma)$, if and only if there is an optimal solution for I' with average costs $\sqrt{2(n-1)c(\sigma)}$. ■

3.3.2 *Feasibility condition*

Observe that d_i/p_i is the fraction of time product i needs to be scheduled on the machine, and thus with necessity $\sum_{i \in J} d_i/p_i$ should be at most 1. The following lemma shows that this is actually also sufficient.

Lemma 3.4 (Feasibility)**.** *For all three variants of the problem, there exists a feasible schedule if and only if*

$$\sum_{i \in J} \frac{d_i}{p_i} \leqslant 1 .$$

Proof. Let S be a feasible cyclic schedule of length ℓ. Then for each product i, the total demand during S equals ℓd_i. Since we can produce at most p_i during a time unit t, we know that

$$\frac{\ell d_i}{p_i} \leqslant \int_0^{\ell} x_i^t \, dt .$$

Summing over all products gives

$$\sum_{i \in J} \frac{\ell d_i}{p_i} \leqslant \sum_{i \in J} \int_0^{\ell} x_i^t \, dt \leqslant \ell .$$

From the feasibility and the indication property of the function x_i, we infer that the right-hand side of the first inequality is at most ℓ. Dividing by ℓ yields $\sum_{i\in J} d_i/p_i \leqslant 1$.

Next, suppose that $\sum_{i\in J} d_i/p_i \leqslant 1$. Following the reverse of the proof above, we know that given some initial stock, we can now construct a feasible schedule. Let S be a schedule of length $\ell = \prod_{i\in J} p_i$. Now, order the products in J from 1 to n. For each product i, produce $\pi_i^{[t_{i-1},t_i]} = \ell d_i$, where $t_i = t_{i-1} + \ell d_i/p_i$ and $t_0 = 0$. Clearly, given enough initial stock, demand is met for each product. Since $\sum_{i\in J}(t_i - t_{i-1}) = \sum_{i\in J} \ell d_i/p_i \leqslant \ell$, all production fits in the cycle. Additionally, given integer demands and production rates, $\ell d_i/p_i$ is integer, ensuring feasibility for the fixed case. ■

Remark 3.5. Note that additionally for LSP(F,n), a schedule of length ℓ is feasible if and only if

$$\frac{\ell d_i}{p_i} = \sum_{t=0}^{\ell-1} x_i^t \in \mathbb{N}, \quad \text{and} \quad \ell \bmod (p_i - d_i) = 0, \quad \forall_{i\in J}\,.$$

3.3.3 *Characterizing optimal production schedules*

In this subsection we prove several properties about the production in continuous and discrete schedules. We start by showing that if there is some idle time in a schedule, we can already start producing the next product at demand rate during the idle time to decrease the holding costs.

Lemma 3.6 (No idle time)**.** *Let S^* be an optimal schedule for* LSP(C,n) *or* LSP(D,n)*, with $n \in \mathbb{N}$. S^* has no idle time.*

Proof. We prove by contradiction. Let S be a counterexample, i.e., there is at least one interval $[t_1, t_2]$ of length $t_2 - t_1 = t$ where the machine is idle. Thus in this interval, each product i has a demand $d_i t$ to fulfill. Therefore, for each i there is a stock of at least $d_i t$ at time t_1, and thus for this interval, we pay at least $\sum_{i\in J} d_i t h_i$.

Let i be the product produced at time t_2. Produce i at the demand rate of i during the interval, i.e., $\pi_i^{[t_1,t_2]} = d_i t$. Consequently, reduce the production during the rest of the schedule, i.e., reduce $\pi_i^{[t_2,\ell]\cup[0,t_1]}$ by $d_i t$. Clearly, the schedule remains feasible, the switching costs are the same as before, and the holding costs are reduced by at least $\frac{1}{2} h_i d_i t^2$. ■

We now provide a short proof for the claim that in an optimal schedule for the continuous case, at any time the production rate is always larger than or equal to the demand rate of the produced product.

Lemma 3.7 (Produce at least the demand rate)**.** *Let* S^* *be an optimal schedule for* LSP(C,n) *with* $n \in \mathbb{N}$*. For each phase* $[t, t']_i^r$ *in* S^**, we have that* $r \geqslant d_i$*.*

Proof. We prove by contradiction. Let S be a counterexample, i.e., there is at least one phase $[t, t']_i^r$ with $r < d_i$. Since S^* is feasible, we know that $q_i^t \geqslant (d_i - r)(t' - t) > 0$. Now let $\pi_i^{[t',\ell]\cup[0,t]} \leftarrow \pi_i^{[t',\ell]\cup[0,t]} - (d_i - r)\,(t - t')$ and replace $[t, t']_i^r$ by $[t, t']_i^{d_i}$. Clearly the schedule is feasible and the costs are decreased, and thus S^* was not optimal. ■

The next property ensures that the machine produces every product i only at rates d_i and p_i to minimize holding costs in the continuous case.

Lemma 3.8 (Two phase production)**.** *Consider* LSP(C,n) *for any* $n \geqslant 2$*. There is an optimal cycle* S^* *such that for every product* $i \in J$*, every production period of* i *in* S^* *consists of at most two phases. For every production period, in the first phase the machine produces* i *at a rate of* d_i*. During the second (non-empty) phase* i *is produced at a rate of* p_i*.*

Proof. We know from Lemma 3.7 that for each phase $[t, t']_i^r$, we have that $r \geqslant d_i$. Now suppose there is a phase $[t_1, t_3]_i^r$ such that $d_i < r < p_i$ with holding costs $h_i \frac{1}{2}(t_3 - t_1)(q_i^{t_3} + q_i^{t_1})$. Now, let $[t_1, t_2]_i^{d_i}$ and $[t_2, t_3]_i^{p_i}$ where $t_2 = \frac{t_3(r-p_i)+t_1(d_i-r)}{(d_i-p_i)}$, with $\int_{t_1}^{t_2} q_i^t \, dt = (t_2 - t_1)q_i^{t_1}$ and $q_i^{t_3}$ remains unchanged. The holding costs are now $h_i \frac{1}{2}(t_3 - t_2)(q_i^{t_3} + q_i^{t_1}) + h_i(t_2 - t_1)q_i^{t_1} < h_i \frac{1}{2}(t_3 - t_1)(q_i^{t_3} + q_i^{t_1})$ and thus, $[t_1, t_3]_i^r$ is not optimal. Therefore, in an optimal schedule each production period consists of consecutive phases of the form $[t_1, t_2]_i^{d_i}, [t_2, t_3]_i^{p_i}$.

Now suppose that P consists of more than two such phases. In that case there exists a time t such that there are phases $[t_1, t]_i^{p_i}, [t, t_2]_i^{d_i}$ with holding costs $h_i \left(\frac{1}{2}(t - t_1)(q_i^t + q_i^{t_1}) + (t_2 - t)q_i^t\right)$. Now swap the order of the two phases, i.e., let $[t_1, t']_i^{d_i}, [t', t_2]_i^{p_i}$ with $t' = t_1 + (t_2 - t)$, $q_i^{t_2} = q_i^t$ and holding costs $h_i \frac{1}{2}((t_2 - t_1) + (t_2 - t))(q_i^t + q_i^{t_1})$. Since holding costs decrease by $q_i^{t_1} t$, P will consist of at most two phases of the form $[t_1, t_2]_i^{d_i}, [t_2, t_3]_i^{p_i}$. ■

Note that in a *tight* schedule, i.e., $\sum_{i \in J} d_i/p_i = 1$, in order to meet demand for each product, the machine needs to continuously produce at maximum speed. Therefore, in an optimal schedule S for a tight instance of the

problem, each production period consists of a single phase where product i is produced at rate p_i. Furthermore, the proof of Lemma 3.8 also proves that in an optimal schedule for LSP(C,n), for each phase $[t, t']_i^r$, we have that $r = d_i$ or $r = p_i$.

Following the same reasoning as in the previous lemma, we can achieve a similar result for the discrete case of the problem and prove that in an optimal schedule, production periods consist of at most four phases.

Lemma 3.9 (Four phase production)**.** *Consider* LSP(D,n) *for any* $n \geqslant 2$*. There is an optimal cycle* S^* *such that for every product* $i \in J$*, every production period of* i *in* S^* *consists of at most four phases. For every production period, in the first phase the machine produces* i *at a rate of* $r_1 < d_i$ *and this phase has at most length* 1*. During the second phase* i *is produced at a rate of* d_i*. During the third phase,* i *is produced at rate* $d_i < r_2 < p_i$ *and this phase again has at most length* 1*. Finally, during the fourth phase,* i *is produced at a rate of* p_i*. Phases can be empty, but the first and third phase cannot occur sequentially.*

Proof. We prove by contradiction. We claim, and this is easy to check, that phases within the production period can be ordered such that for every pair of phases $[t_j, t_{j+1}]_i^{r_1}, [t_{j'}, t_{j'+1}]_i^{r_2}$ with $j' > j$ we have that $r_1 < r_2$ in order to minimize costs whilst retaining a feasible schedule.

Suppose we have an optimal schedule S with two consecutive phases $[t_j, t_{j+1}]_i^{r_j}, [t_{j+1}, t_{j+2}]_i^{r_{j+1}}$. By definition of a phase, $r_j \neq r_{j+1}$. Since S is optimal, $0 < r_j < r_{j+1} \leqslant p_i$ must hold. Clearly, if $t_{j+2} = t_{j+1} + 1 = t_j + 2$, the lemma holds. We construct a new schedule S^* and we start this construction letting $S^* := S$.

We split $[t_j, t_{j+1}]_i^{r_j}, [t_{j+1}, t_{j+2}]_i^{r_{j+1}}$ in S^* into five new phases as follows. We first deplete the stock by q° and consecutively increase the stock by q^*, with these values depending on the case distinction below. We introduce the indicator function $f_{\mathbb{N}}(x) = \lceil x \rceil - \lfloor x \rfloor$ which takes on the value 1 if $x \notin \mathbb{N}$ and 0 otherwise. The five new phases are

$$
\begin{aligned}
&[t_j, t_1]_i^0, [t_1, t_2]_i^{r_1}, [t_2, t_3]_i^{d_i}, [t_3, t_4]_i^{r_2}, [t_4, t_{j+2}]_i^{p_i}, \\
&\text{where } t_1 = t_j + \left\lfloor \frac{q^\circ}{d_i} \right\rfloor, \text{ and } t_2 = t_1 + f_{\mathbb{N}}\left(\frac{q^\circ}{d_i}\right), \\
&t_4 = t_{j+2} - \left\lfloor \frac{q^*}{p_i - d_i} \right\rfloor, \text{ and } t_3 = t_4 - f_{\mathbb{N}}\left(\frac{q^*}{p_i - d_i}\right), \\
&r_1 = d_i - \left(q_i^{t_j} - (t_1 - t_j)d_i\right) \\
&\text{and } r_2 = d_i + \left(q_i^{t_{j+2}} - (t_{j+2} - t_4)(p_i - d_i)\right).
\end{aligned}
$$

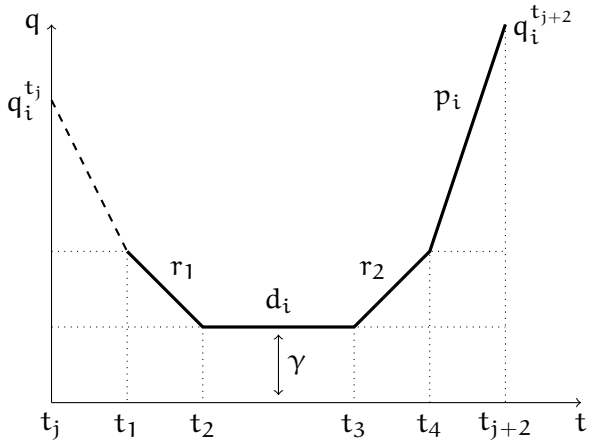

Figure 3.2: A depiction of an optimal production period of schedule S^* for LSP(C,n) (where phases $[t_1, t_2]_i^{r_1}$ and $[t_3, t_4]_i^{r_2}$ must be empty) and for LSP(D,n), with $n \geqslant 2$. Notice that the stock level during demand production can only be positive, i.e., $\gamma > 0$, if $(t_3 - t_2) \leqslant 1$ for LSP(D,n).

We refer the reader to Fig. 3.2 for a depiction of the new set of phases.

Firstly, suppose $d_i \leqslant r_j < r_{j+1}$. Now let $q^* = (q_i^{t_{j+2}} - q_i^{t_j})$ and $q^\circ = 0$, consequently producing stock, which results in a production period of at most 3 phases.

Secondly, suppose $r_j < r_{j+1} \leqslant d_i$. Now let $q^\circ = (q_i^{t_j} - q_i^{t_{j+2}})$ and $q^* = 0$, consequently depleting stock, which results in a production period of at most 3 phases.

Lastly, suppose $r_j < d_i < r_{j+1}$. Now let $q^\circ = q_i^{t_j}$ and $q^* = q_i^{t_{j+2}}$, consequently first depleting and consecutively producing stock, which results in a production period of at most 4 phases.

If completely depleting and consecutively producing the stock takes longer than the production period, we get $t_2 > t_3$. In this case, denote the total amount of stock which was produced in this production period by $q = (t_{j+2} - t_j)d_i + (q_i^{t_{j+2}} - q_i^{t_j}) = r_j(t_{j+1} - t_j) + r_{j+1}(t_{j+2} - t_{j+1})$. Then let $t_4 = t_{j+2} - \left\lfloor \frac{q}{p_i} \right\rfloor$ and $t_1 = t_2 = t_3 = t_{j+2} - \left\lceil \frac{q}{p_i} \right\rceil$ and $r_2 = d_i + q - p_i \left\lfloor \frac{q}{p_i} \right\rfloor$, resulting in a production period of at most 3 phases.

Clearly, in all cases S^* is feasible. If S^* is different from S then $H(S^*) < H(S)$, and thus S is not optimal. Note that the phase $[t_j, t_1]_i^0$ is idle and can be removed as in the proof of Lemma 3.6 by extending or introducing demand production for some other product, thereby delaying its stock

production, leaving a production period of four phases and proving the lemma. ■

Note that the proof of Lemma 3.9 also shows that in an optimal schedule for LSP(D,n), for each phase $[t, t']_i^r$ with $t' > t + 1$, we have that $r = d_i$ or $r = p_i$.

We now show that in the continuous case, the machine produces product i at rate d_i only if the stock for i is empty.

Lemma 3.10 (Level production for continuous case)**.** *In an optimal schedule* S^* *for an instance of* LSP(C,n)*, for any product* $i \in J$ *there exists a non-empty phase* $[t_j, t_{j+1}]_i^{d_i}$ *(i.e., with* $t_{j+1} > t_j$*) only if* $q_i^{t_j} = 0$.

Proof. We prove by contradiction. Suppose we have an optimal schedule S with a phase $[t_j, t_{j+1}]_i^{d_i}$ with $t_{j+1} > t_j$ and $q_i^{t_j} > 0$. Again, we construct a new schedule S^* starting with $S^* := S$. We split $[t_j, t_{j+1}]_i^{d_i}$ in S^* into three new phases:

$$[t_j, t_1]_i^0, [t_1, t_2]_i^{d_i}, [t_2, t_{j+1}]_i^{p_i},$$
$$\text{where } t_1 = t_j + \frac{q_i^{t_j}}{d_i} \text{ and } t_2 = t_{j+1} - \frac{q_i^{t_{j+1}}}{p_i - d_i}.$$

If the length of the phase is too short to completely deplete the stock, and consecutively completely rebuild the stock, i.e., $t_1 > t_2$, then we reduce the stock as much as possible. In this case, let $t_1 = t_2 = t_{j+1} - t^\circ$, where $t^\circ = (t_{j+1} - t_j)\frac{d_i}{p_i}$ denotes the time required to produce when producing at rate p_i in order to meet demand during the original phase.

Clearly, S^* is feasible and now we have that $H(S^*) < H(S)$ and thus S is not optimal. ■

We now show a similar result for the discrete case, where the machine produces product i at rate d_i only if the stock for i is empty *or* if the production phase has length 1.

Lemma 3.11 (Level production for discrete case)**.** *In an optimal schedule* S^* *for an instance of* LSP(D,n)*, for any product* $i \in J$ *there exists a non-empty phase* $[t_j, t_{j+1}]_i^{d_i}$ *only if* $q_i^{t_j} = 0$ *or* $t_{j+1} = t_j + 1$.

Proof. We prove by contradiction. Suppose we have an optimal schedule S with a phase $[t_j, t_{j+1}]_i^{d_i}$ with $t_{j+1} = t_j + 2$ and $q_i^{t_j} > 0$. Once again,

we construct a new schedule S^* starting with $S^* := S$. We can now split $[t_j, t_{j+1}]_i^{d_i}$ in S^* into two new phases:

$$[t_j, t_j + 1]_i^{r_1}, [t_j + 1, t_{j+1}]_i^{r_2}.$$

If $q_i^{t_j} < d_i$, let $r_1 = d_i - q_i^{t_j}$ and $r_2 = d_i + q_i^{t_j}$. Otherwise, let $r_1 = \max\{2d_i - p_i, 0\}$ and $r_2 = \min\{p_i, 2d_i\}$. Clearly, S^* is feasible and we have that $H(S^*) < H(S)$ and thus S is not optimal.

Next, suppose $t_{j+1} > t_j + 2$. Now split $[t_j, t_{j+1}]_i^{d_i}$ into the phases $[t_j, t_j + 1]_i^{r_1}[t_j + 1, t_{j+1} - 1]_i^{d_i}[t_{j+1} - 1, t_{j+1}]_i^{r_2}$, where r_1 and r_2 are defined as above. Clearly this process can be iteratively repeated upon the schedule S^* until either the stock level reaches 0, or there is at most one phase left of length 1. ■

We can now show that in an optimal schedule, for every product there is a time where its stock level is zero.

Lemma 3.12 (Zero stock level)**.** *Let S^* be an optimal schedule for an instance of* LSP(C,n) *or* LSP(D,n)*. Then for each $i \in J$ there exists a time t such that $q_i^t = 0$.*

Proof. The proof is by contradiction. Let S be an optimal schedule of length ℓ with at least one product i such that $q_i^t > 0$ for all t. Let t^* be such that $q_i^{t^*} = \min_{0 \leqslant t \leqslant \ell} q_i^t$. Now let S^* be a copy of S, where we decrease the stock level for the entire schedule of this product, i.e., $q_i^t \leftarrow q_i^t - q_i^{t^*}$ for all $0 \leqslant t \leqslant \ell$. Since $q_i^{t^*} \leqslant q_i^t$ for all t in S, we know that S^* is feasible. Clearly, $H(S^*) < H(S)$, and thus S is not optimal. Note that the stock level can be decreased by producing at a rate lower than required by the schedule until the desired level is attained. ■

3.3.4 *Bounding the average costs*

We conclude the basic properties with a lower bound on the average costs of an optimal continuous schedule, and an upper bound on the average costs and maximum stock level of an optimal discrete schedule. To obtain these, we first derive optimality conditions for both cases.

Lemma 3.13 (Continuous cost balancing)**.** *An optimal schedule S for an instance of* LSP(C,n) *has the property that $H(S) = W(S)$.*

Proof. We prove by contradiction. Let S be an optimal schedule s.t. $H(S) \neq W(S)$. Scale the length of each phase in S by a positive factor $\delta \neq 1$, such

that for the resulting feasible schedule S' it holds that $W(S') = H(S')$. The holding costs for product i in S during a phase $[t_1, t_2]_i^r$ are given as

$$H(i, [t_1, t_2]_i^r) = (t_2 - t_1)q_{\min} + \frac{(t_2 - t_1)^2}{2} r,$$

where $q_{\min}$ is the minimum stock level of i during the phase. Thus for the corresponding phase $[t_1', t_2']_i^r$ of the scaled schedule S' we have

$$\begin{aligned} H(i, [t_1', t_2']_i^r) &= (t_2' - t_1')q_{\min}' + \frac{(t_2' - t_1')^2}{2} r \\ &\leqslant (t_2 - t_1)\delta^2 q_{\min} + \frac{(t_2 - t_1)^2\delta^2}{2} r \\ &= H(i, [t_1, t_2]_i^r)\delta^2 . \end{aligned}$$

Summing over all phases and products we get

$$H(S') = \sum_{[t_1', t_2']_i^r \in S'} \sum_{i \in J} H(i, [t_1', t_2']_i^r) \leqslant H(S)\delta^2 .$$

Observe that due to scaling the schedule, we have that $W(S) = W(S') = H(S') \leqslant H(S)\delta^2$. Rewriting gives us $\delta \geqslant \sqrt{W(S)/H(S)}$. We now choose δ such that

$$\sqrt{\frac{W(S)}{H(S)}} \leqslant \delta < \frac{1}{2} + \frac{W(S)}{2H(S)} . \tag{3.1}$$

Observe that for any values of $H(S)$ and $W(S)$ s.t. $H(S) \neq W(S)$, there exists a δ satisfying (3.1). Because of this particular choice of δ, we have that

$$\begin{aligned} \bar{c}(S') = \frac{1}{\ell'} H(S') + \frac{1}{\ell'} W(S') = \frac{1}{\ell\delta} H(S') + \frac{1}{\ell\delta} W(S) &\leqslant \frac{\delta}{\ell} H(S) + \frac{\delta}{\ell} H(S) \\ = \frac{2\delta}{\ell} H(S) < \frac{1}{\ell} H(S) + \frac{1}{\ell} W(S) &= \bar{c}(S) , \end{aligned}$$

and thus S was not optimal, proving the lemma. ■

We now prove a similar result for the discrete case, taking into account that low values of δ might create infeasible schedules.

Lemma 3.14 (Discrete cost balancing). *A schedule S for an instance of* LSP(D,n) *is optimal only if $W(S) < 4 \cdot H(S)$.*

Proof. The lemma follows almost entirely from the proof of Lemma 3.13. The difference is that in the discrete case, we might introduce infeasible schedules S' by stretching with any factor δ. Therefore, we restrict ourselves to factors $\delta \in \mathbb{N}$, $\delta \geqslant 2$. The first inequality in (3.1) now holds only if $W(S) < 4 \cdot H(S)$. ∎

Remark 3.15. Note that instead of simply scaling by $\delta \geqslant 2$, we can also scale by a multiple of $(1 + \frac{1}{\alpha})$, where α is the length of the shortest production period in S. The first inequality in (3.1) now holds only if $W(S) < (1 + \frac{1}{\alpha})^2 \cdot H(S)$.

To obtain a lower bound on the average costs, we first fully characterize the optimal continuous schedule for instances where all products are identical.

Lemma 3.16 (Identical products)**.** *For* LSP(C,n) *with* n *identical products, i.e.,* $d_i = d$, $p_i = p$, $h_i = h$ *and* $s_{i,j} = s$ *for all* $i, j \in J$, *the optimal schedule* S^* *is a simple cycle of average costs* $\bar{c}(S^*)$ *and length* ℓ, *where*

$$\bar{c} = n\alpha\sqrt{\frac{2s(p-d)dh}{p}} \text{ and } \ell = \frac{1}{\alpha}\sqrt{\frac{2sp}{(p-d)dh}},$$
$$\text{where} \alpha = \left(1 - \frac{1}{n} + \frac{d}{p}\right).$$

Proof. Since all products are identical, the optimal schedule is defined by a simple cycle where all products are produced for the same period of time and $H(S) = W(S)$, see Lemma 3.13.

We first look at a single *product block* $[t_1, t_2, t_3]_i$, which denotes the period for a single item i from the moment it starts a production period, until it starts another production period. Here, $[t_1, t_2]$ denotes the production period for product i, and $[t_2, t_3]$ denotes the period during which i is not produced. Note that $q_i^{t_1} = q_i^{t_3} = 0$. See Fig. 3.3.

The holding costs for a single product i are given as $\frac{xq}{2}$, where q is the maximum stock level and x is the time where product i is produced at rate p plus the time it is not produced, during the product block. Given a slope of $p - d$ during production, and a slope of $-d$ during non-production, we have $a = \frac{dx}{p}$, $b = x - a = x(1 - \frac{d}{p})$ and $q = a(p - d)$, resulting in total holding costs of

$$\frac{xq}{2}h = x\frac{a(p-d)}{2}h = x^2\frac{(p-d)d}{2p}h.$$

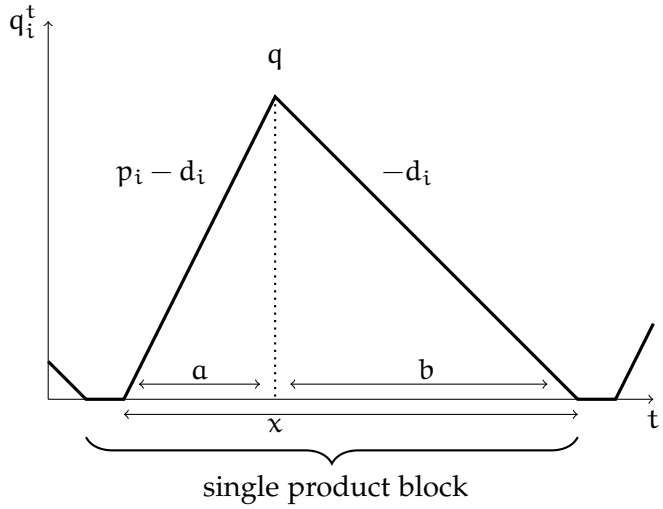

Figure 3.3: An example of a single minimum *product block*.

For each product, the length of the product block is given as the total length ℓ. Note that $1 - \frac{dn}{p}$ is the fraction of time during which the machine produces any product at rate d. Since all products are produced for an equal amount of time, the fraction of time during which one product is produced at rate d is $\frac{1}{n} - \frac{d}{p}$. Define $\alpha := \left(1 - \frac{1}{n} - \frac{d}{p}\right)$, yielding $x = \ell\alpha$. Note that in a tight schedule, $\alpha = 1$.

The optimal schedule S has total sequencing costs ns and total holding costs $x^2 \frac{(p-d)d}{2p} hn = \ell^2 \alpha^2 \frac{(p-d)d}{2p} hn$. Thus the average costs are given as

$$\frac{ns}{\ell} + \ell\alpha^2 \frac{(p-d)d}{2p} hn\,.$$

Again, note the similarities with the classical EOQ formula. We now find the optimal cycle length ℓ using that $W(S) = H(S)$. Given the optimal length, we can calculate the average total costs $\bar{c}(S)$, yielding

$$\ell = \frac{1}{\alpha}\sqrt{\frac{2sp}{(p-d)dh}} \quad \text{and} \quad \bar{c}(S) = n\alpha\sqrt{\frac{2s(p-d)dh}{p}}\,. \qquad \blacksquare$$

Using the characterization for identical products, we can construct a lower bound on the average costs of a schedule.

Lemma 3.17 (Lower bound on average costs). *Consider* LSP(C,n) *for* $n > 1$. *Let* S^* *be the optimal schedule. Let* i *be the product minimizing* $\frac{(p_i-d_i)d_i}{2p_i}h_i$ *and let* $s^{\min} = \min_{i,j\in J} s_{i,j}$ *be the minimum switching costs. Then*

$$\bar{c}(S^*) \geqslant n\alpha\sqrt{\frac{2s^{\min}(p_i-d_i)d_ih_i}{p_i}}, \text{where } \alpha = \left(1-\frac{1}{n}+\frac{d_i}{p_i}\right).$$

Proof. Intuitively, we construct a schedule for n identical products with d, p and h equal to the corresponding value of the least costly product i.

In order to lower bound the holding costs, assume that the stock-level is zero at the beginning and end of the product block, i.e., $q_i^{t_1} = q_i^{t_3} = 0$. Furthermore, assume that we produce n times a certain dummy product i, such that $h_i \leqslant h_j$ and $\frac{d_i}{p_i} \leqslant \frac{d_j}{p_j}$ for all $j \in J$. From Lemma 3.16, we know that the holding costs of a single product block for i are equal to

$$\frac{xq}{2}h_i = x^2\frac{(p_i-d_i)d_i}{2p_i}h_i,$$

where x and q are defined as in Lemma 3.16. Now, let $x^2h^{\min}$ denote the minimum holding costs for each product during the block $[t_1,t_2,t_3]_i$, where

$$h^{\min} = \min_{i\in J}\frac{(p_i-d_i)d_i}{2p_i}h_i.$$

Choosing i as the product minimizing $h^{\min}$ and $s^{\min} = \min_{i,j\neq i\in J} s_{ij}$, we can now use Lemma 3.16 to prove the lemma. ∎

The following lemma bounds the length of a specific class of feasible instances, which is used to create an upper bound on the average costs in Lemma 3.19.

Lemma 3.18 (Upper bound on schedule length for discrete case). *Consider* LSP(D,n) *for* $n \geqslant 2$. *Let* S *be any minimum length feasible simple cycle such that* $c(S) = W(S) + H(S) < 5\,H(S)$. *The length of* S *is bounded by*

$$\ell^{\max} = \left(\prod_{i\in J} p_i^-\right)\sqrt{\frac{\sum_{i,j\in J} s_{ij}}{2\sum_{i\in J}\frac{(p_i^- - d_i)d_i}{p_i^-}h_i}}, \text{ where } p_i^- = p_i\sum_{j\in J}\frac{d_j}{p_j}.$$

Proof. Since we are interested in a minimum length feasible schedule, we can assume the schedule is tight by limiting the maximum production rates. To be precise, for each product $i \in J$ we limit production to

$$p_i^- = p_i\sum_{j\in J}\frac{d_j}{p_j},$$

yielding $\sum_{i\in J} d_i/p_i^- = 1$, which constitutes a tight schedule.

Now, since S is a simple cycle, we can calculate the total holding costs $H(S)$ as

$$H(S) = \sum_{i\in J} h_i \int_0^\ell q_i^t \, dt = \sum_{i\in J} \ell^2 \frac{(p_i^- - d_i)d_i}{2p_i^-} h_i \,,$$

where ℓ is the length of S. Furthermore, we have that $W(S) \leqslant \sum_{i,j\in J} s_{ij}$. Combining the above with the requirement that $c(S) = W(S) + H(S) < 5H(S)$, we get

$$\ell > \ell^{\min} = \sqrt{\frac{\sum_{i,j\in J} s_{ij}}{2\sum_{i\in J} \frac{(p_i^- - d_i)d_i}{p_i^-} h_i}} \,.$$

Observe that in a feasible tight schedule for the discrete case, it must be that $\ell \frac{d_i}{p_i^-} \in \mathbb{N}_+$ for each product $i \in J$. Note that if ℓ is a multiple of $\prod_{i\in J} p_i^-$, this condition is satisfied. Any feasible schedule of length at least $\ell^{\min}$ now constitutes an upper bound on the length of S. Combining this condition with the above inequality yields an upper bound for ℓ of

$$\ell \leqslant \ell^{\max} = \left(\prod_{i\in J} p_i^-\right) \sqrt{\frac{\sum_{i,j\in J} s_{ij}}{2\sum_{i\in J} \frac{(p_i^- - d_i)d_i}{p_i^-} h_i}} \,,$$

proving the lemma. ■

We now present an upper bound on the average costs of an optimal discrete schedule.

Lemma 3.19 (Upper bound on average costs). *Consider* LSP(D,n) *for* $n \geqslant 2$. *The average costs of an optimal schedule* $\bar{c}(S^*)$ *are bounded by*

$$\bar{c}(S^*) \leqslant \frac{5}{2} \sum_{i\in J} h_i \frac{(p_i - d_i)d_i}{p_i} \ell^{\max} \,.$$

Proof. Let S be any minimum length feasible simple cycle such that $c(S) = W(S) + H(S) < 5\, H(S)$. Because of Lemma 3.18, we know that $H(S) \leqslant \sum_{i\in J} \ell^2 \frac{(p_i - d_i)d_i}{2p_i} h_i \leqslant \sum_{i\in J} \ell^{\max} \frac{(p_i - d_i)d_i}{2p_i} h_i \ell$. Recall that by assumption, $c(S) = W(S) + H(S) < 5H(S)$ and S has length $\ell \geqslant n$. Let S^* denote an optimal schedule. Now observe that

$$\bar{c}(S^*) \leqslant \bar{c}(S) = \frac{W(S) + H(S)}{\ell} < \frac{5H(S)}{\ell} \,.$$

Substituting $H(S)$ by its upper bound proves the lemma. ■

Using the previous lemma, we can now bound the maximum stock level.

Lemma 3.20 (Maximum Stock level)**.** *Consider* LSP(D,n) *for* $n \geqslant 2$*. The maximum stock level in an optimal schedule* S^* *is bounded by*

$$Q = 5\left(\sum_{i\in J} h_i \frac{(p_i - d_i)d_i}{p_i}\right)\left(\prod_{i\in J} p_i\right)\ell^{\max}.$$

Proof. Observe that $\frac{Q}{2} \geqslant \bar{c}(S^*) = \frac{H(S^*)+W(S^*)}{\ell} \geqslant \frac{H(S^*)}{\ell} \geqslant \frac{1}{2}\max_{t\in S, i\in J} q_i^t$. ■

3.4 OPTIMAL SOLUTIONS FOR FEW PRODUCTS

In this section, we use the previous results to derive optimal solutions for the fixed case with one product and the continuous case with two products. We can be very short about the discrete and continuous case with one product: it is clear that in these cases it is optimal to deplete any remaining stock and then to produce at demand rate.

3.4.1 *Fixed case with one product*

We first characterize the minimum cycle length for LSP(F,1), followed by the costs of an optimal schedule. The proof shows that for an optimal schedule S^*, the inventory levels for the time units in the schedule are the multiples of $\gcd(p_1, d_1)$ smaller than p_1.

Lemma 3.21. *The minimum cycle length for* LSP(F,1) *is*

$$l^* = \frac{p_1}{\gcd(p_1, d_1)}. \tag{3.2}$$

Proof. Denote $g = \gcd(p_1, d_1)$. Assume without loss of generality that $q_1^0 < p_1$. Since the cycle must be feasible, we have that $d_1 \leqslant p_1$.

Producing p_1 provides stock for $\lfloor p_1/d_1 \rfloor$ time units, with a leftover stock of $p_1 \mod d_1$. Let stock at time t be $q_1^t = q_1^{t-1} + \pi_1^t - d_1$. The schedule is cyclic when $q_1^t = q_1^0$ for $t > 0$. For a minimum cycle length, we want to minimize over t such that $q_1^t = q_1^0 + \sum_{u=1}^t \pi_1^u - d_1 t = q_1^0$. Rewriting gives

$$t = \frac{\sum_{u=1}^t \pi_1^u}{d_1} = \sum_{u=1}^t x_1^u \frac{p_1}{d_1}.$$

Clearly, t is minimized when $\sum_{u=1}^t x_1^u = \frac{d_1}{g}$, and thus $t = \frac{p_1}{g} = l^*$. ■

Using this lemma we compute the costs of an optimal schedule.

Lemma 3.22. *The shortest optimal cyclic schedule* S^* *for* LSP(F,1) *has unit costs of*

$$\bar{c}(S^*) = \frac{h_1}{2}\left(p_1 - \gcd(p_1, d_1)\right) . \tag{3.3}$$

Proof. Denote $\mathcal{G} = \gcd(p_1, d_1)$. Using the reasoning from Lemma 3.12, we can assume without loss of generality that the initial stock $q_1^0 = 0$. Let S^* be the optimal cyclic schedule with length l^*. Since S^* is cyclic, q_1^t has unique values for $t = 0, \ldots, l^* - 1$. Suppose $l^* > p_1/\mathcal{G}$. Then each q_1^t is a multiple of $\mathcal{G}$. Since $l^* > p_1/\mathcal{G}$ and each q_1^t has a unique value, there exists at least one t such that $q_1^t \geqslant p_1$, and thus the schedule is not optimal. Thus the length of the shortest optimal schedule is $l^* = p_1/\mathcal{G}$.

Since the total demand during the cycle is $d_1 l^*$ and each time unit of production produces p_1, we know that we produce during $d_1 l^*/p_1 = d_1/\mathcal{G}$ time units. Since q_1^t has a unique value for each $t < l^*$ and $q_1^0 = 0$, the stock values are all multiples of $\mathcal{G}$. Hence, the values of q_1^t are the multiples of $\mathcal{G}$ smaller than p_1. Since $p_1 = l^*\mathcal{G}$, the total stock for the cycle equals $\sum_{j=0}^{l^*-1} j\mathcal{G}$.

Thus the total costs of S^* are

$$h_1 \sum_{j=0}^{l^*-1} j\mathcal{G} = h_1 \frac{1}{2}\mathcal{G} l^*(l^* - 1) = \frac{h_1 p_1}{2}\left(\frac{p_1}{\mathcal{G}} - 1\right) . \qquad \blacksquare$$

We can summarize the optimal schedule for this case in the following theorem.

Theorem 3.23. *In an optimal schedule* S^* *for* LSP(F,1), $\pi_1^t > 0$ *if and only if* $q_1^{t-1} < d_1$.

The optimal schedule S^* has length l^* as in (3.2), and total costs $l^*\bar{c}$ as in (3.3)). The length of the cycle is linear in $p_1/\gcd(p_1, d_1)$, and Theorem 3.23 yields a polynomial delay list-generating algorithm.

3.4.2 *Continuous case with two products*

We start with the following lemma.

Lemma 3.24. *There exists an optimal schedule for* LSP(C,2) *that is a simple cycle.*

Proof. Consider a minimal counterexample S^* consisting of the production periods $[0,t_1]_1,[t_1,t_2]_2,[t_2,t_3]_1,[t_3,C]_2$, where $t_1 \neq (t_3 - t_2)$. Now denote $A_1 = (t_1 + t_3 - t_2)/2$ and consider the schedule

$$S = [0,A_1]_1,[A_1,C/2]_2,[C/2,C/2+A_1]_1,[C/2+A_1,C]_2\,,$$

which is obtained from S^* by replacing the two production periods of each product by two production periods with averaged length. Since S^* is feasible, we have that $\pi_1^{[0,t_1]} + \pi_1^{[t_2,t_3]} \geqslant Cd_1$ and $\pi_2^{[t_1,t_2]} + \pi_2^{[t_3,C]} \geqslant Cd_2$. Let $\pi_1^{[0,A_1]} = d_1 C/2$ in S to cover the demand for product 1 during the first two production periods. Let the production during the other production periods be similar. Clearly, S is feasible. Note that $(t_2 - t_1) + (C - t_3) = (C/2 - A_1) + (C - C/2 - A_1)$, i.e., the sum of the lengths of the production periods for product i in S, is equal to that in S^*.

Now suppose there is in S^* a production period $[a,b]$ for product 1 with $q_1^a(S^*) > 0$. Then during the production period $[x,a]_2$, holding costs increase by $q_1^a(S^*)h_1(x-a)$ compared to S and thus $\bar{c}(S) < \bar{c}(S^*)$.

Next, suppose $q_i^a(S^*) = 0$ for every production period $[a,b]_i$. It is easy to see that holding costs for product 1 are only paid during production periods for 2 and during the non-empty phase where product 1 is produced at rate p_1. The same result holds for product 2. Note that the sum of the lengths of the production periods for product i in S, is equal to that in S^* and holding costs are linear. Hence, the area under the curve of the function of the holding costs over time, is the same in S as in S^*, thus $\bar{c}(S) \leqslant \bar{c}(S^*)$.

Observe that S consists of two simple cycles S' and S'' with $S' = S''$. Thus S' is a feasible simple cycle with the same unit costs as S. ■

For the rest of this section we assume without loss of generality that $h_1 < h_2$, and we only consider simple cycles. Next we show that an optimal schedule for LSP(C,2) consists of at most three phases.

Lemma 3.25. *There exists an optimal schedule for any* LSP(C,2) *instance of the form*

$$S^* = [0,t_1]_1^{p_1},[t_1,t_2]_2^{d_2},[t_2,C]_2^{p_2}\,, \tag{3.4}$$

where the second phase is empty if and only if $d_1/p_1 + d_2/p_2 = 1$.

Proof. Let S be an optimal cycle with four non-empty phases, i.e.,

$$S = [0,t_1]_1^{p_1},[t_1,t_2]_2^{d_2},[t_2,C]_2^{p_2},[C,t_3]_1^{d_1}\,.$$

Consider the schedule consisting of only the first three phases, i.e., we remove $[C, t_3]_1^{d_1}$. Note that $\pi_2^{[t_2,C]} = d_2(t_1 + (t_3 - C)) > d_2 t_1$. Hence the total amount of production for product 2 can be lowered by $(t_3 - C)d_2$, by decreasing the length of phase $[t_2, C]_2^{p_2}$. Let $\alpha = (t_3 - C)d_2/p_2$ and let

$$S^* = [0, t_1]_1^{p_1}, [t_1, t_2 + \alpha]_2^{d_2}, [t_2 + \alpha, C]_2^{p_2}.$$

Clearly S^* is feasible and $\bar{c}(S^*) < \bar{c}(S)$.

If $d_1/p_1 + d_2/p_2 = 1$ the schedule is tight and demand can only be met by producing at maximum rate, which implies $[t_1, t_2 + \alpha]_2^{d_2}$ is empty.

If $d_1/p_1 + d_2/p_2 < 1$, there has to be a phase in which the machine does not produce at maximum rate, to avoid overproduction. By Lemma 3.24 there are at most two phases of production at rate d_1 and d_2 respectively. Since $h_1 < h_2$, by the above reasoning we introduce only one phase where we produce d_2 in order to minimize costs. ■

Using this result we calculate the optimal cycle length and corresponding costs. Define

$$\bar{c}(t) = \left(\frac{h_1(p_1 - d_1)}{2} + \frac{h_2 d_1 d_2}{2p_1}\left(1 + \frac{d_2}{p_2 - d_2}\right)\right)t + \left((s_{1,2} + s_{2,1})\frac{d_1}{p_1}\right)\frac{1}{t},$$

and

$$t^* = \sqrt{\frac{2(s_{1,2} + s_{2,1})d_1}{h_1 p_1(p_1 - d_1) + h_2 d_1 d_2\left(1 + \frac{d_2}{p_2 - d_2}\right)}}. \tag{3.5}$$

Then we can prove the following theorem.

Theorem 3.26. *For* LSP(C,2) *there exists an optimal schedule of length* $t^* p_1/d_1$ *with average costs* $\bar{c}(t^*)$.

Proof. We denote the optimal schedule by $S^* = [0, t]_1^{p_1}, [t, t']_2^{d_2}, [t', C]_2^{p_2}$.

We parametrize on t. Split the schedule in sub-schedules $S_1 = [0, t]_1^{p_1}$, $S_2 = [t, t']_2^{d_2}$ and $S_3 = [t', C]_2^{p_2}$. Let $c_i(S)$ denote the cost of (sub-)schedule S for product i. Note that $q_1^C = q_1^0 = q_2^t = q_2^{t'} = 0$, $q_2^C = q_2^0 = d_2 t$ and

$q_1^t = t(p_1 - d_1)$ and $q_1^{t'} = q_1^t - d_1(t' - t)$. We calculate the average cost $\bar{c}(t)$ of S^* as follows.

$$\begin{aligned}
\bar{c}(t) &= \frac{1}{C}\left(c_1(S^*) + c_2(S^*) + s_{1,2} + s_{2,1}\right) \\
&= \frac{1}{C}\left(c_1(S_1) + c_1(S_2 + S_3)\right) + \frac{1}{C}\left(c_2(S_1) + c_2(S_3)\right) + \frac{s_{1,2} + s_{2,1}}{C} \\
&= \frac{1}{C}\left(h_1 \frac{1}{2} t^2 (p_1 - d_1) + h_1 \frac{1}{2} t (p_1 - d_1)(C - t)\right) \\
&\quad + \frac{1}{C}\left(h_2 \frac{1}{2} t^2 d_2 + h_2 \frac{1}{2} \frac{d_2 t}{p_2 - d_2} d_2 t\right) + \frac{s_{1,2} + s_{2,1}}{C} \\
&= \frac{h_1 t (p_1 - d_1)}{2} + \frac{h_2 t^2 d_2}{2C}\left(1 + \frac{d_2}{p_2 - d_2}\right) + \frac{s_{1,2} + s_{2,1}}{C} \\
&= \left(\frac{h_1 (p_1 - d_1)}{2} + \frac{h_2 d_1 d_2}{2p_1}\left(1 + \frac{d_2}{p_2 - d_2}\right)\right) t_1 \\
&\quad + \left((s_{1,2} + s_{2,1})\frac{d_1}{p_1}\right)\frac{1}{t_1}.
\end{aligned}$$

Since $c(t_1)$ is convex this expression is minimized when $\frac{dc(t_1)}{dt_1} = 0$. We find Eq. (3.5) and thus the optimal average costs are equal to $\bar{c}(t^*)$. ■

3.5 APPROXIMATION ALGORITHMS

Note that already for two products the optimal schedule can have pseudopolynomial length. This poses an inherent problem in processing the problem in polynomial time, particularly in outputting the schedule in polynomial time.

In this section, we overcome these difficulties and present two approximation algorithms: First, we *augment* the problem and solve this to optimality, yielding an augmented polynomial time approximation scheme (PTAS, cf. Definition 1.5) for the discrete case (cf. Remark 3.29). Next, we convert the augmented discrete solution into a feasible solution for the continuous case, yielding a polynomial time approximation algorithm. In both cases, the schedule produced has polynomial length. The algorithm constructs solutions in polynomial time given a constant number of products. Observe that the latter is a reasonable assumption: in real-life instances the number of products is relatively small. Throughout this section we assume S^* is an optimal cyclic schedule of length ℓ and q_i^t for $i \in J$ and $t = 0, \ldots, \ell - 1$ denotes the optimal stock level in S^*.

The general idea is to *augment* the production and demand rates, i.e., we allow for slightly higher production rates and modestly adjusted demand rates. For a given $\delta > 0$, we lift the stock levels q_i^t for all i and t to powers of $(1+\delta)$, and use augmentation to keep the schedule feasible. For every time unit t we generate states, which are defined by stock levels q_i^t for each product $i \in J$ and the product being produced. By Lemma 3.20, the maximum stock level is bounded by Q, yielding a polynomial number of states. With these states, we create a state-graph and find a minimum mean cycle using Karp's algorithm [68], in order to get an optimal schedule for the augmented version of LSP(D,n). Finally, we balance the resulting schedule such that it becomes a close to optimal solution for LSP(D,n) and a feasible schedule for LSP(C,n). See Algorithm 3.1 for the pseudocode of the algorithm.

Let a state $S_i = (q_1, \ldots, q_n)$ be defined as an ordered set of stock levels q_j for each product $j \in J$, where subscript $i \in J$ denotes the last product which has been produced before reaching the current state. Let d_i^t denote the augmented demand for a product $i \in J$ in time unit t. For each time unit t and a product i which is produced, we allow for augmented production rates r_i^t such that the total augmented production is no more than $(1+\delta)$ times the total production in a feasible schedule. Specifically, we require that augmented production satisfies the following two conditions.

$$r_i^t < p_i + \delta(q_i^t + p_i - d_i)\,, \tag{3.6}$$

$$\sum_{t=0}^{\ell-1} r_i^t - d_i^t < \ell(p_i - d_i)(1+\delta)\,. \tag{3.7}$$

The first equation ensures for each time unit an upper bound on the augmented production rate, such that the next power of $(1+\delta)$ can be reached for the stock level. Note that this actually augments the stock level rather than the production rates. In order to limit the total augmentation in terms of the production rates, the latter equation ensures that the total production in the augmented schedule is not more than $(1+\delta)$ times the maximum possible production in the non-augmented schedule. This is to ensure that in practice we get an augmented schedule which is reasonably achievable with respect to the original input data.

Additionally, for each time-unit t with product i that is not produced during t, for augmented demand rates $d_i^t \geqslant 0$ it must hold that

$$\frac{q_i^t - d_i}{q_i^t - d_i^t} \leqslant 1 + \delta\,. \tag{3.8}$$

Algorithm 3.1: Augmentation Algorithm AugAlg

Input: A set $\mathcal{P}$ of n products with demand rates d_i, maximum production rate p_i and holding costs h_i for all $i \in \mathcal{P}$.

Output: Augmented schedule S_D and schedule S_C.

Create the set $\mathcal{S}$ of all states $S_i = (q_1, \ldots, q_n)$ Let $E = \emptyset$ be the set of state-edges

foreach pair of states $S_i, S_j \in \mathcal{S}$ **do**

  if $S_i(q_j) - d_j < S_j(q_j) \leqslant (S_i(q_j) + p_j - d_j)(1+\delta)$ **then**

    if $(S_i(q_k) - d_k)/(1+\delta) \leqslant S_j(q_k) \leqslant (S_i(q_k) - d_k)$for every $k \neq j \in \mathcal{P}$ **then**

      Create directed edge $e = (S_i, S_j)$ with cost $c_e = s_{ij} + \frac{1}{2}\sum_{k\in J}\left(S_i(q_k) + S_j(q_k)\right)$

Find the minimum mean cycle C^* in $\mathcal{S}$ using Karp's algorithm, cf. [68], discarding edge progressions which do not admit Eqs. (3.6) to (3.9) Extract augmented schedule S_D from C^*

Let all demands $d_i^t \leftarrow d_i$ and decrease production rates such that $r_i^t \leqslant p_i$ and $\sum_{t=0}^{\ell-1} r_i^t - d_i \leqslant 0$ For each product i with $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$, uniformly increase production rates $r_i^t < p_i$ until $\sum_{t=0}^{\ell-1} r_i^t = \ell d_i$ or $r_i^t \leftarrow p_i$ for all t Let $S_C \leftarrow S_D$ be a Continuous schedule with x^t the length of time slot t

foreach $i \in J$ *such that* $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$ **do**

  Simultaneously increase all x^t in S_C where $r_i^t > 0$ and decrease all $x^{t'}$ where $r_j^t > 0$ for all $j \neq j \in J$ such that $\sum_{t=0}^{\ell-1} r_j^t \geqslant \ell d_j$ remains true, until $\sum_{t=0}^{\ell-1} r_i^t = \ell d_i$

return S_D,S_C

This equation ensures that demand rates are not increased more than necessary in order to retain stock levels within a factor of $(1+\delta)$. Moreover, for all time units t and products i, we require the following to ensure that the total demand in the augmented schedule is not more than $(1+\delta)$ times the total demand in the non-augmented schedule, i.e.,

$$\ell d_i \leqslant \sum_{k=0}^{\ell-1} d_i^k \,. \tag{3.9}$$

Note that transgressing from one state to the next is equivalent to a single time-unit in a schedule for the discrete case. Let each edge (S_i, S_j) have costs $s_{ij} + \frac{1}{2}\sum_{k\in J} h_k \left(S_i(q_k) + S_j(q_k)\right)$. Note that here $s_{ii} = 0$.

We now describe the algorithm (cf. Algorithm 3.1). In the first step of AugAlg, an augmented state-graph is constructed, with a state for each combination of stock levels q_i for each product i, such that $q_i \leqslant Q(1+\delta)$ and q_i is a power of $(1+\delta)$, where Q is the maximum stock level as in Lemma 3.20. Let S_i be a state in the optimal schedule with $S_i(q_j)$ the stock level for product j in S_i.

An edge is added from state S_i to S_j if and only if $S_i(q_j) - d_j < S_j(q_j) \leqslant (S_i(q_j) + p_j - d_j)(1+\delta)$ and $(S_i(q_k) - d_k)/(1+\delta) \leqslant S_j(q_k) \leqslant (S_i(q_k) - d_k)$ for all $k \neq j \in J$.

Recall Karp's algorithm for finding a minimum mean cycle in a digraph. The algorithm uses a dynamic program to compute values $F_k(v)$ for each vertex v and each $0 \leqslant k \leqslant n$, where $F_k(v)$ denotes the minimum weight of an edge progression of length k from some arbitrarily chosen vertex s to v. Using these values the algorithm computes the minimum mean cycle. We slightly adjust the dynamic program in Karp's algorithm: Upon evaluating the computed values $F_k(v)$, discard any of these edge progressions which do not admit Eqs. (3.6) to (3.9), ensuring that these conditions hold for the minimum mean cycle returned by the algorithm.

Observe that the minimum mean cycle returned by Karp's algorithm constitutes a feasible augmented schedule to the problem.

Lemma 3.27. *Let S be a schedule for* LSP(D,n) *and let $\varepsilon > 0$. There exists an augmented schedule S' such that $\bar{c}(S') \leqslant (1+\varepsilon)\bar{c}(S)$.*

Proof. Let S be a schedule for LSP(D,n) and let $q_j^t(\delta) > q_j^t(S)$ be the nearest power of $(1+\delta)$ greater than $q_j^t(S)$. Note that by Lemma 3.12, each product i in both the augmented and the non-augmented solution must have at least one point where its stock level is zero. Denote this point as time-unit

0_i with $q_i^{0_i}(S') = q_i^{0_i}(S) = 0$. For each product $i \in J$, starting at zero stock level $q_i^{0_i}(S)$ successively change rates $r_i^t(S)$ and d_i to $r_i^t(S')$ and $d_i^t(S')$ for each time-unit t as follows.

- If $q_i^t(\delta) = q_i^t(S)$ then let $r_i^t(S')$ remain the same as in S.
- Otherwise, if product i is produced, let the production rate be $r_i^t(S') \leftarrow r_i^t(S) + q_i^t(\delta) - q_i^t(S)$. However, since we want to bound the increase of the costs of the augmented schedule, we bound the stock level throughout the augmented schedule. For every product, for each of its production periods, we ensure that the cumulative amount of stock up to that point is no more than $(1+\delta)$ times the corresponding original stock. Thus, if $\sum_{k=0}^{t-1} \left(r_i^k(S') - d_i\right) + \left(q_i^t(\delta) - q_i^t(S)\right) \geqslant t(p_i - d_i)(1+\delta)$, then choose $r_i^t(S')$ such that $q_i^t(S')$ becomes the largest power of $(1+\delta)$ such that $q_i^t(S') < q_i^t(\delta)$.
- If i is not produced, choose the smallest $d_i^t(S') \geqslant d_i$ such that $q_i^t(S')$ is a power of $(1+\delta)$.

Observe that every stock level in S is a power of $(1+\delta)$ and the schedule is feasible. Since every stock level increased by at most $(1+\delta)$, the total costs for the schedule are increased by at most $c(S)\delta$. Choosing $\varepsilon = \delta$ proves the lemma. ■

Applying the above lemma to an optimal schedule and bounding the running time of AugAlg yields the following result.

Theorem 3.28. *Let S^* be an optimal schedule for* LSP(D,n) *and let $\varepsilon > 0$. AugAlg finds an* augmented *schedule S_D for* LSP(D,n) *such that $c(S_D) \leqslant (1+\varepsilon)c(S^*)$ in time $O\left(\left(\log_{1+\delta}(Q)\right)^n n^2\right)$.*

Proof. Consider AugAlg and observe that the algorithm finds a schedule S_D such that $\bar{c}(S_D) \leqslant (1+\varepsilon)\bar{c}(S^*)$, where S^* is the optimal schedule, as in Lemma 3.27.

Note that there are $O((\log_{1+\delta}(Q))^n n)$ states in $\mathcal{S}$, and thus there are at most $O((\log_{1+\delta}(Q))^n n^2)$ edges. Karp's algorithm runs in $O(m+n)$ time, where m is the number of edges in the graph, proving the theorem. ■

Remark 3.29. Although formally the running time is not polynomial, recall that n is typically a small constant in instances of this problem. Therefore, for all practical purposes, the running time can nevertheless be considered to be polynomial in the input size.

Using AugAlg, we construct a polynomial time approximation algorithm for the continuous problem. To this end, we need to bound the costs of an optimal discrete schedule in terms of an optimal continuous schedule.

Lemma 3.30 (Pseudopolynomial ratio)**.** *Given an instance of the lot sizing problem, let* S_D *and* S_C *be the optimal schedules for* LSP(D,n) *and* LSP(C,n) *respectively. Then, there is a polynomial* $\xi([p_i]_J, [d_i]_J, [h_i]_J, [s_{ij}]_{J\times J})$*, such that* $\bar{c}(S_D) \leqslant \xi \cdot \bar{c}(S_C)$.

Proof. From Lemma 3.17 and Lemma 3.19 we know that $\bar{c}(S_C) \geqslant \varphi_1$ and $\bar{c}(S_D) \leqslant \varphi_2$ for given polynomials φ_1 and φ_2. Hence $\frac{\bar{c}(S_D)}{\bar{c}(S_C)} \leqslant \frac{\varphi_2}{\varphi_1}$, which is bounded by a polynomial ξ in $[p_i]_J$, $[d_i]_J$, $[h_i]_J$, and $[s_{ij}]_{J\times J}$. ■

We can now formulate the following theorem.

Theorem 3.31. *Let* S^* *be an optimal schedule for* LSP(C,n) *and let* $\varepsilon > 0$*. AugAlg finds a feasible schedule* S_C *for* LSP(C,n) *of polynomial length such that* $c(S_C) \leqslant (1+\varepsilon)\xi c(S^*)$ *in time* $O\left(\left(\log_{1+\delta}(Q)\right)^n n^2\right)$.

Proof. Ensure that $\sum_{i\in J} \frac{d_i}{p_i} \leqslant 1$, otherwise there exists no feasible schedule. Run AugAlg to get an augmented schedule S_D for the corresponding instance of LSP(D,n). Observe that the schedule is a feasible augmented schedule for LSP(C,n).

First, lower the demand and production rates to feasible values, i.e., let all demands $d_i^t \leftarrow d_i$ and decrease production rates such that $r_i^t \leqslant p_i$. Next, for each product i for which total production does not cover total demand, i.e., for which $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$, uniformly increase production rates $r_i^t < p_i$ until demand is satisfied, i.e., $\sum_{t=0}^{\ell-1} r_i^t = \ell d_i$, or until $r_i^t = p_i$ for all t. Denote the schedule obtained after this transformation by S'_D. Since demand and production rates are decreased by at most a factor of $(1+\delta)$, overproduction in S'_D is no more than $(1+\delta)\ell(p_i - d_i)$, therefore the costs are bounded as $c(S'_D) \leqslant (1+\delta)c(S_D)$.

Note that if $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$, we cannot satisfy total demand for product i and the lengths of production periods will need to be adjusted. Let x^t denote the length of time slot t. Clearly, there are ℓ time slots. For each product $i \in J$ such that $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$, we will increase all production lengths x^t where $r_i^t > 0$ to meet the demand of product i. To retain feasibility for all products $j \neq i \in J$, we increase production rates and shorten production periods where possible, whilst keeping the schedule length constant. For each product $j \in J$ such that $\sum_{t=0}^{\ell-1} r_j^t \geqslant \ell d_j$, we consider the following three numbered categories:

1 For all t where $0 < r_j^t < p_j$, we will increase r_j^t and decrease x^t such that total production in x^t remains unchanged, at most up to the point where $r_j^t = p_j$.

2 If $\sum_{t=0}^{\ell-1} r_j^t > \ell d_j$ and $r_j^t \in \{0, p_j\}$ for all t, we will decrease lengths of production periods x^t with $r_j^t > 0$, at most up to the point where $\sum_{t=0}^{\ell-1} r_j^t = \ell d_j$.

3 If $\sum_{t=0}^{\ell-1} r_j^t = \ell d_j$ and $r_j^t \in \{0, p_j\}$ for all t, the schedule is tight for this product.

For each $i \in J$ such that $\sum_{t=0}^{\ell-1} r_i^t < \ell d_i$, simultaneously increase all x^t where $r_i^t > 0$, increase r_j^t and decrease x^t for all products j as in Category 1, and decrease x^t for all products j as in Category 2, while keeping the schedule length ℓ constant. Note that the category number of a production period can only be increased by applying the transformation. Hence, since $\sum_{i \in J} \frac{d_i}{p_i} \leqslant 1$, and each production period can be categorised as above, this transformation terminates successfully. Finally, for any product which is produced more than the total demand throughout the cycle, we uniformly decrease production rates for this product - without altering the length of the production period - until demand is met exactly. We denote the resulting schedule by S_C. For the remainder of this proof we assess the quality of S_C.

First look at a single increment of length $\alpha \leqslant \delta$ for a time-unit t where i is produced and $\sum_{t=0}^{\ell-1} r_i^t(S'_D) < \ell d_i$. Let $c_j^t(S)$ denote the costs for a product $j \in J$ during time slot t in a schedule S. Since the production rate is increased in the transformation by a factor $(1+\alpha)$, the costs for product i at time slot t are bounded by $(1+\alpha)c_i^t(S'_D)$. Similarly, the cost for each product $j \neq i \in J$ is increased to at most $(1+\alpha)c_j^t(S'_D)$. At the end of every production period, stock levels in S_C are not increased compared to stock levels in S_D.

Secondly, look at a single decrement of α for a time-unit t where i is produced. Clearly, the costs $c_i^t(S_C)$ do not increase. Furthermore, the costs $c_j^t(S)$ for each product $j \neq i \in J$ are neither increased. Since the production period is shortened, the stock level for each product $j \neq i$ at the end of the production period is increased. In a worst case scenario, this extra stock needs to be carried throughout the entire schedule. Hence, for each decrement of α, for each product $j \neq i$, total costs for the product in the entire schedule can be increased by at most $\alpha d_j h_j \ell$. Observe that $\alpha d_j h_j \ell \leqslant \delta c_j(S)$.

Recall that the maximum increment for a single time-unit is at most δ. Each product for which time units are increased, increases total costs for all products by at most $\delta c(S'_D)$. Furthermore, for each product for which time-units are decreased, costs increase by at most $\delta c(S'_D)$ in total. Thus AugAlg produces a feasible schedule S for LSP(C,n) of costs at most $(1+n\delta)c(S'_D)$. From Lemma 3.30 we know that $\bar{c}(S_D) \leqslant \xi\bar{c}(S^*)$.

Hence, $c(S_C) \leqslant (1+n\delta)c(S'_D) \leqslant (1+n\delta)(1+\delta)c(S_D) \leqslant (1+n\delta)(1+\delta)^2\xi c(S^*)$. Choosing ε such that $(1+\varepsilon) = (1+n\delta)(1+\delta)^2$ proves the theorem. ■

Like in Remark 3.29, we remark that although the running time is not polynomial in the input size, for all practical purposes it can be considered to be polynomial nonetheless.

3.6 CONCLUDING REMARKS

This chapter combines the hardness of high multiplicity encoding with sequence-dependent setup costs, both of which are natural properties of real-life problems. Not only does this introduce hardness akin to the TRAVELLING SALESPERSON PROBLEM, but due to the compact encoding it is not clear whether or not a polynomially sized certificate can be constructed, even for very restricted cases. We discussed the complexity of the problem and presented structural properties largely characterizing optimal schedules, which can be used for future algorithms and computational experiments. We presented a polynomial time augmented approximation scheme, which finds $(1+\varepsilon)$-approximate augmented solutions for the discrete variant of the problem, and ξ-approximate solutions for the continuous case. In contrast to the known complexity of the problem, the algorithm runs in polynomial time and yields schedules of polynomial length.

It is unclear whether it can be guaranteed that an optimal schedule exists at all. Consider the case of LSP(C,2), where the optimal schedule is already irrational even under rational input values. Is it possible that due to the irrationality of the cost-balance, the optimal schedule for LSP(C,3) has infinite length? Can it nevertheless be approximated with a finite schedule? Considering instances with 2 products, can we characterize the optimal solutions for the discrete and the fixed case? We conjecture this is possible to achieve using techniques similar to the ones used in this chapter.

Alternatively, consider the settings where we explicitly make assumptions concerning the input instances. For instance, if the sequence is given,

e.g. using a TSP-oracle, is it possible to find an (approximately) optimal solution for all cases in polynomial time? Or if the sequencing costs have a lexicographical ordering (e.g. when the products only differ in colour and setting up the machine when switching between two similar colours costs less), can we obtain stronger results?

Regarding the complexity of the problem, we conjecture that this problem is contained in a higher complexity class than NP: Already for LSP(F,1) and LSP(C,2), the optimal schedule can be of non-polynomial length. Although the schedule for these cases can still be represented in polynomial time, it is uncertain if this can be done for arbitrary numbers of products. Furthermore, consider the following decision problem: Does there exist an optimal cyclic schedule of average costs k? It is unclear whether this decision problem is contained in NP, and how an adequate polynomial certificate for a NO-instance can be constructed.

4

VECTOR SCHEDULING

4.1 INTRODUCTION

We consider the VECTOR SCHEDULING problem defined as follows. The input consists of a collection J of n jobs $\mathbf{p_1}, \ldots, \mathbf{p_n}$, viewed as d-dimensional vectors from $[0,1]^d$, and m identical machines. The goal is to find an assignment of the jobs to the machines that minimizes $\left\|\sum_{\mathbf{p}\in P_i} \mathbf{p}\right\|_\infty$ for each machine $i \in \{1, 2, \ldots, m\}$, where P_i is the set of jobs assigned to machine i. By a scaling argument we can assume without loss of generality that $\left\|\sum_{\mathbf{p}\in P_i} \mathbf{p}\right\|_\infty \leqslant 1$, that is, the maximum load on any machine in any coordinate is at most 1.

VECTOR SCHEDULING is the natural multi-dimensional generalization of the classic MULTIPROCESSOR SCHEDULING problem (also known as makespan minimization, $P||C_{max}$, or load balancing). In the latter problem, the goal is to assign n jobs with arbitrary processing times to m machines in order to minimize the maximum sum of processing times (load) over all the machines. However, for many applications, the jobs may use different resources and the load of a job cannot be described by a single aggregate measure. For example, if jobs have both CPU and memory requirements, their processing requirement is best modelled as a two-dimensional vector, where the value in each coordinate corresponds to each of the requirements. Note that the assumption that the maximum load of a machine in any coordinate is 1 is without loss of generality, as the different coordinates can be scaled independently.

In this chapter we are again concerned with approximation algorithms. In this context, we say that an algorithm is an α-approximation for some $\alpha > 1$ if it finds an assignment with maximum load at most α times the maximum load of an assignment with minimum maximum load. Again by a scaling argument, we can assume this is the case if the algorithm finds an assignment with maximum load at most α whenever there exists a feasible schedule with maximum load 1.

4.1.1 *Previous work*

MULTIPROCESSOR SCHEDULING and the related BIN PACKING problem are two of the most fundamental problems in combinatorial optimization with a long and rich history. We only describe the work on MULTIPROCESSOR SCHEDULING in the setting where the number of machines m is part of the input. It is well-known that MULTIPROCESSOR SCHEDULING is strongly NP-hard [43].

The first polynomial time approximation scheme (PTAS, cf. Definition 1.5) was obtained by Hochbaum and Shmoys [61]. The running time of their algorithm is $O(n^{O(1/\varepsilon^2)})$. Note that by the strong NP-Hardness of the problem one cannot hope to have a running time with polynomial dependence in ε (i.e. an FPTAS), unless P=NP.

An efficient polynomial time approximation scheme (EPTAS) was implicit in [61] by replacing the dynamic program by an integer linear program and using fast integer programming algorithms in fixed dimensions. Alon et al. [4] developed a more general framework to obtain EPTASes for parallel machine scheduling that runs in $f(\varepsilon) + O(n)$ time, where $f(\varepsilon)$ is a double exponential function in $1/\varepsilon$.

Recently, this running time was substantially improved by Jansen [65] to $O(2^{\tilde{O}(1/\varepsilon^2)} + n^{O(1)})$. His main idea is to use fast integer programming in fixed dimensions, together with an elegant result of Eisenbrand and Shmonin [31] about the existence of optimum integer solutions with small support. Most of these results also extend to the setting of uniform machines, i.e. a setting where the machine speeds differs (see e.g. [62, 65]).

Fewer results are known for the case when the number of dimensions exceeds one. Chekuri and Khanna [23] gave the first polynomial-time approximation scheme for a fixed number of dimensions. They gave an algorithm with running time $n^{g(\varepsilon,d)}$, where $g(\varepsilon, d) = (1/\varepsilon)^{d \log\log d + o(d)}$ and hence the running time is $n^{(1/\varepsilon)^{\tilde{O}(d)}}$. This seems to be the currently best known running time for this problem. PTASes for several other generalizations are also known [15, 32, 33].

When d is part of the input, Chekuri and Khanna [23] gave a polynomial time $O(\ln^2 d)$-approximation and proved that it is NP-hard to approximate the problem within any constant factor. This approximation factor has been recently improved to $O(\log d)$ by Meyerson et al. [81]. The latter result even holds in the online setting.

4.1.2 *Our contribution*

A natural question is whether there exists an approximation scheme for VECTOR SCHEDULING with a single exponential running time in $1/\varepsilon$ and d, e.g. $\exp(\text{poly}(1/\varepsilon, d))$. We rule out this possibility by showing the following strong lower bound.

Theorem 4.1. *For any $\varepsilon > 0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{o\left((1/\varepsilon)^{d/3}\right)}(nd)^{\mathcal{O}(1)}\right)$ for* VECTOR SCHEDULING *in $d \geqslant d(\varepsilon)$ dimensions, unless the Exponential Time Hypothesis (ETH) fails.*

This follows from a relatively simple reduction from the 3-DIMENSIONAL MATCHING problem. The same reduction also implies the following hardness under a more standard complexity assumption.

Theorem 4.2. *For any $\varepsilon > 0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{(1/\varepsilon)^{o(d)}}(nd)^{\mathcal{O}(1)}\right)$ for* VECTOR SCHEDULING *in $d \geqslant d(\varepsilon)$ dimensions, unless NP has subexponential time algorithms, i.e. $NP \subseteq \cap_{\varepsilon>0} DTIME(2^{n^\varepsilon})$.*

One may wonder whether these lower bounds are robust or whether they crucially exploit the fact that no additional machines are allowed. It is instructive to consider the case of $d = 1$ (i.e. MULTIPROCESSOR SCHEDULING). Recall that no FPTAS is possible for the problem. However, if one allows some extra machines (say $\lceil \varepsilon m \rceil$ of them), then the running time dependence on ε reduces dramatically and in particular, an FPTAS is possible. In fact, the known FPTASes for BIN PACKING imply that even very few extra machines (poly-logarithmic in m) suffice [67, 88], and in fact one does not even need to violate the capacity of any machine.

Somewhat surprisingly, we show that extra machines do not help for VECTOR SCHEDULING, provided that the desired approximation ratio is sufficiently small.

Theorem 4.3. *For any $\varepsilon < \varepsilon_0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{(1/\varepsilon)^{o(d)}}(nd)^{\mathcal{O}(1)}\right)$ for* VECTOR SCHEDULING *in $d \geqslant d(\varepsilon)$ dimensions, even with $\lceil \varepsilon m \rceil$ extra machines, unless NP has subexponential time algorithms, i.e. $NP \subseteq \cap_{\varepsilon>0} DTIME(2^{n^\varepsilon})$, where $\varepsilon_0 < 1$ is a universal constant. Assuming the ETH, no such algorithm can run in time $\mathcal{O}\left(2^{\left(o(1/\varepsilon)^{d/6}\right)}(nd)^{\mathcal{O}(1)}\right)$.*

To complement the lower bounds above, we show the following algorithmic result.

Theorem 4.4. *For any $\varepsilon > 0$ and $d \geqslant 1$, there is a deterministic $(1+\varepsilon)$-approximation algorithm for d-dimensional* VECTOR SCHEDULING *that runs in time $\mathcal{O}\left(2^{(1/\varepsilon)^{\mathcal{O}(d \log\log d)}} + nd\right)$.*

By the lower bounds above, the running time is essentially the best possible (modulo the $\mathcal{O}(\log\log d)$ factor in the exponent), and the nd term is simply the time required to read the input. Theorem 4.4 gives the first EPTAS for VECTOR SCHEDULING.

Techniques

At a high level, the algorithm is similar to that of [65], and relies on integer programming in fixed dimensions and the existence of optimum integer solutions with small support. However, there are some important differences between $d = 1$ and $d > 1$. In particular, for $d = 1$ the small jobs (with size $\leqslant \varepsilon$) do not cause any problems and can later be assigned greedily in the remaining space, after solving the problem for just big jobs. However, for $d \geqslant 2$, the big and small jobs (by small we mean jobs that are small in *every* dimension) interact in more complex ways and must be considered together. The following example illustrates this difficulty.

Example 4.5. Consider the following instance in $d = 2$ dimensions, with $m = 2$ machines and the following jobs: $\mathbf{p_1} = \left(\frac{1}{2}, 0\right)$, $\mathbf{p_2} = \left(\frac{1}{2}, 0\right)$ and $\mathbf{p_i} = \left(\frac{\varepsilon}{2}, \varepsilon\right)$ for $3 \leqslant i \leqslant 2/\varepsilon$. Clearly, these jobs can be scheduled on two machines by assigning the first two jobs to separate machines and splitting the small jobs evenly, see Fig. 4.1. However, if the two large jobs are assigned to the same machine, there is no assignment of the small jobs such that the maximum load of the machines is exceeded by a constant factor dependent on ε. The two large jobs have total load $(1, 0)$. As the small jobs have total load $(1, 2)$, no matter how these are assigned to the two machines, one machine will have load at least $\min\{\max\{1+x, 2x\}, \max\{1-x, 2(1-x)\}\}$, which is $4/3$ (attained for $x = 1/3$).

Chekuri and Khanna [23] overcame this problem by 'guessing' the division between small and large jobs for each machine. This allows them to decouple the assignment of small and big vectors. However, as there are roughly $m^{(1/\varepsilon)^d}$ different possible divisions, with ε precision, this is not useful to obtain an efficient polynomial time approximation scheme.

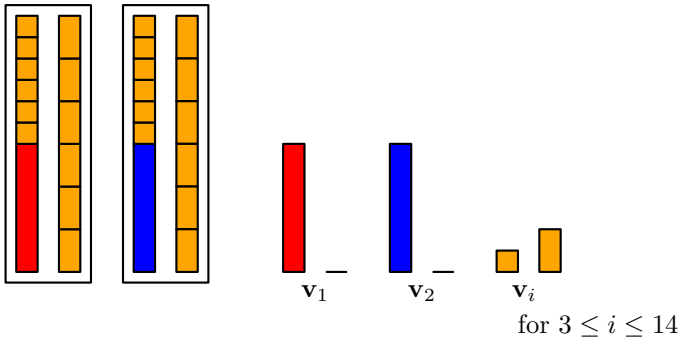

Figure 4.1: A feasible schedule for the instance of Example 4.5.

To get around this, we incorporate both large and small vectors in our mixed integer linear program (MILP), but ensure that it has only few constraints by tracking only some coarse-grained information for the small jobs. We find an optimum solution to this MILP, which gives an integral assignment of large jobs, but small jobs might be assigned fractionally. We then show how to assign the small jobs to machines without overloading them. To do this, we first assign the jobs greedily guided by a potential function, which guarantees that the aggregate amount of overload on machines is small. This load is small enough to ensure that the jobs on overloaded machines can be redistributed in a round-robin manner. A naive implementation of the greedy assignment requires $\mathcal{O}(mn)$ time (as for each job, we need to determine which machine causes the least increase in potential), so we also present some additional ideas to show how everything can be done in linear time.

Organization

In Section 4.2 we state our notation and the hypotheses on which our lower bounds are based, and we describe the relevant background on integer programming. In Section 4.3 we prove our lower bounds for Vector Scheduling and we present our algorithm in Section 4.4.

4.2 PRELIMINARIES

Let $[n]$ denote the set of positive integers 1 to n, i.e. $[n] := \{1, \dots, n\}$. Let $\mathbf{1}$ be the all-ones vector. For a d-dimensional vector $\mathbf{v} = (v_1, \dots, v_d)$, let v_j denote its j-th coordinate. For two vectors $\mathbf{a}, \mathbf{b}$ we say that $\mathbf{a} \leqslant \mathbf{b}$ if $a_i \leqslant b_i$

for all i. Throughout the chapter the logarithm log is taken with base 2 and we let $\exp(x)$ denote 2^x. We say that a function $f(n)$ is sub-exponential if $f(n) \in \mathcal{O}(2^{\mathcal{O}(n^{o(1)})})$. Without loss of generality we assume that the number of machines is less than the number of jobs (otherwise assign one job per machine or conclude infeasibility).

In the 3-CNF-SAT problem, we are given a Boolean expression in conjunctive normal form, consisting of N variables and M clauses that each consist of 3 literals. The question is whether or not there exists an assignment of logical values to the variables such that the expression evaluates to TRUE. Impagliazzo, Paturi, and Zane [64] formulated the Exponential Time Hypothesis, which in combination with the sparsification lemma [19] can be stated as follows.

Hypothesis 4.6 (Exponential Time Hypothesis (ETH) [64])**.** *There is a positive real s such that* 3-CNF-SAT *with N variables and M clauses cannot be solved in time $\mathcal{O}(2^{sM}(N+M)^{\mathcal{O}(1)})$.*

We will use the following well-known result for fast integer linear programs with few integer variables.

Theorem 4.7 (Lenstra [75], Kannan [66], Frank and Tardos [39])**.** *Consider a mixed-integer linear program $\min\{\mathbf{c}^T\mathbf{x} \mid A\mathbf{x} \geqslant \mathbf{b}$ and $\forall i \in \mathcal{I} : x_i \in \mathbb{Z}\}$ with n variables and m constraints, and where $\mathcal{I} \subseteq [n]$ denotes the set of indices of integer variables. Let s denote the binary encoding length of the input. There is an algorithm that finds a feasible solution or decides that there is no feasible solution in $\mathcal{O}(n^{2.5n+o(n)} \cdot s)$ arithmetic operations.*

Relatively recently, based on an elegant pigeonhole argument, Eisenbrand and Shmonin [31] showed that every feasible integer linear program has an optimum solution with small support.

Theorem 4.8 (Eisenbrand and Shmonin [31])**.** *Let $\min\{\mathbf{c}^T\mathbf{y} | A\mathbf{y} = \mathbf{b}, \mathbf{y} \geqslant 0, \mathbf{y} \in \mathbb{Z}^n\}$ be an integer program, where $A \in \mathbb{Z}^{m\times n}$ and $\mathbf{c} \in \mathbb{Z}^n$. If this integer program has a finite optimum, then there exists an optimal solution $\mathbf{y}^* \in \mathbb{Z}^n_{\geqslant 0}$ in which the number of non-zero components is at most $2(m+1)(\log(m+1)+s+2)$, where s is the largest size in binary representation of any coefficient of A and $\mathbf{c}$.*

4.3 LOWER BOUNDS ON THE RUNNING TIME

We prove our lower bounds by a reduction from 3-DIMENSIONAL MATCHING (3-DM) to VECTOR SCHEDULING. In Section 4.3.1 we prove Theorem 4.1 by

describing the reduction and proving that an approximate solution to the VECTOR SCHEDULING instance implies an exact solution for 3-DM and hence 3-CNF-SAT. In Section 4.3.2 we outline how the same reduction implies Theorem 4.2. Finally, in Section 4.3.3 we give the proof for Theorem 4.3 concerning resource augmentation.

Before we give our reduction, we first define the 3-DIMENSIONAL MATCHING problem. An instance of 3-DM consists of three disjoint sets X, Y, and Z, satisfying $|X| = |Y| = |Z| := q$, and a set $T \subset X \times Y \times Z$ of triples. The goal is to find a subset of triples $T' \subset T$ such that each element of X, Y, and Z occurs in exactly one triple of T'.

In [43], a reduction from 3-CNF-SAT to 3-DM is given, that transforms instances of 3-CNF-SAT with N variables and M clauses into instances for 3-DM with $q = 6M$ and $|T| = 17M$. Therefore, the ETH (Hypothesis 4.6) implies there is no $\mathcal{O}(2^{o(q)}|T|^{\mathcal{O}(1)})$ time algorithm for 3-DM.

4.3.1 *Lower bound assuming the ETH*

The construction

The main idea of the reduction is the following construction of a VECTOR SCHEDULING instance from 3-DM. For each triple in T we construct a job (that we call a *triple-job*), and for each element in X, Y or Z we construct as many jobs as the number of times this element occurs in the triples (we call such jobs *element-jobs*). We explicitly refer to *X-jobs, Y-jobs and Z-jobs* if we want to distinguish the element-jobs of the three sets. For each element i, we designate exactly one of its jobs as the *real* element-job corresponding to i, and refer to the other element-jobs of i as *dummy* jobs. The number of machines is equal to the number of triples. We will assign sizes to these jobs such that to obtain a schedule where the maximum load in any coordinate is at most 1, we need to schedule each triple together with its corresponding three element-jobs, and moreover these element-jobs are either all *real* or all dummy element-jobs.

Let $\varepsilon > 0$ be such that $1/\varepsilon$ is integer. Let $b = 1/\varepsilon - 1$ and let $\mathbf{b}$ denote the vector that has b in every coordinate. By $\langle i \rangle$ we denote the $(b+1)$-ary encoding of the integer i and by $\overline{\langle i \rangle}$ we denote its complement, that is, $\overline{\langle i \rangle} := \mathbf{b} - \langle i \rangle$. Let $\langle i \rangle_j$ denote the j-th digit from the right of $\langle i \rangle$. For ease of notation, we scale the jobs by a factor b. That is, all jobs are vectors in $[0, b]^d$ and we want to know whether we can schedule the jobs such that the maximum load in each coordinate is at most b. To make the proofs easier

to read, we rename the elements in the sets X, Y and Z by assuming that $X = Y = Z = \{1, \ldots, q\}$.

The formal reduction

Given an instance $(X, Y, Z; T)$ of 3-DM, let $n_X(i)$ denote the number of triples (x, y, z) for which $x = i$; in a similar way, we define $n_Y(i)$ and $n_Z(i)$. For each element $i \in X$, we create $n_X(i)$ jobs, one *real* X-job i and $n_X(i) - 1$ *dummy* X-jobs. In a similar way, we create $n_Y(j)$ Y-jobs for each element $j \in Y$ and $n_Z(k)$ Z-jobs for each element $k \in Z$. Finally, we have $|T|$ triple-jobs, one for each triple $l \in T$. The number of machines is equal to $m := |T|$. Note that the total number of jobs is $\sum_{i \in X} n_X(i) + \sum_{j \in Y} n_Y(j) + \sum_{k \in Z} n_Z(k) + |T| = 4|T|$.

Recall that $|X| = |Y| = |Z| = q$, and let $\ell := \lceil \log_{(1/\varepsilon)} q \rceil$. We associate a vector to each of the jobs as in Table 4.1. These vectors are d-dimensional, where $d := 7 + 3\ell$. In particular, the first four coordinates of a job indicate whether the job corresponds to an element in X, Y, Z or to a triple in T. The following three coordinates encode for each X, Y, or Z-job whether it is a real job or a dummy job. The last part of each job encodes the element to which the job corresponds.

Proof of the reduction

We now show that the reduction has the desired properties.

Lemma 4.9. *(Completeness) If the* 3-DM *instance has a solution, then there exists an assignment of the jobs to the* m *machines such that the load on every machine in each coordinate is at most* b.

Proof. Consider the collection T' of disjoint triples that cover X, Y and Z. For each triple $(i, j, k) \in T'$ we assign the corresponding triple-job and the real element-jobs corresponding to i, j and k to a single machine. Clearly, every coordinate on every such machine has load at most b. We place each of the remaining triples (i, j, k) on a machine with a dummy job for i, for j and for k. It is easily verified that this is a feasible assignment. ■

Lemma 4.10. *If the* VECTOR SCHEDULING *instance has a solution with load at most* $(1 + \varepsilon)b$, *then there is a solution to the corresponding* 3-DM *instance.*

Proof. Consider any solution with load at most $(1 + \varepsilon)b$. We begin with various properties of such a solution.

Job name	Values of the coordinates				
	T/X/Y/Z	Real/dummy	Encoding of element(s)		
real X-job i:	$0, b, 0, 0$	$b, 0, 0$	$\langle i\rangle_1, \ldots, \langle i\rangle_\ell$	$0, \ldots, 0$	$0, \ldots, 0$
dummy X-job i:	$0, b, 0, 0$	$0, b, 0$	$\langle i\rangle_1, \ldots, \langle i\rangle_\ell$	$0, \ldots, 0$	$0, \ldots, 0$
real Y-job j:	$0, 0, b, 0$	$0, b, 0$	$0, \ldots, 0$	$\langle j\rangle_1, \ldots, \langle j\rangle_\ell$	$0, \ldots, 0$
dummy Y-job j:	$0, 0, b, 0$	$0, 0, b$	$0, \ldots, 0$	$\langle j\rangle_1, \ldots, \langle j\rangle_\ell$	$0, \ldots, 0$
real Z-job k:	$0, 0, 0, b$	$0, 0, b$	$0, \ldots, 0$	$0, \ldots, 0$	$\langle k\rangle_1, \ldots, \langle k\rangle_\ell$
dummy Z-job k:	$0, 0, 0, b$	$b, 0, 0$	$0, \ldots, 0$	$0, \ldots, 0$	$\langle k\rangle_1, \ldots, \langle k\rangle_\ell$
triple (i, j, k):	$b, 0, 0, 0$	$0, 0, 0$	$\overline{\langle i\rangle}_1, \ldots, \overline{\langle i\rangle}_\ell$	$\overline{\langle j\rangle}_1, \ldots, \overline{\langle j\rangle}_\ell$	$\overline{\langle k\rangle}_1, \ldots, \overline{\langle k\rangle}_\ell$

Table 4.1: Construction of the jobs from elements and triples of the 3-DM problem.

Property 4.11. The load is exactly b in each coordinate on each machine.

Proof. The load of each machine is at most $(1+\varepsilon)b = b + b/(b+1) < b+1$. As all jobs have integer coordinates, the load of each machine is at most b.

Moreover, since $\sum_{i\in X} n_X(i) = \sum_{j\in Y} n_Y(j) = \sum_{k\in Z} n_Z(k) = |T| = m$, observe that the total amount of work in the i-th coordinate summed over all jobs is mb. As all jobs are scheduled and the load is at most b, it is *exactly* b. ■

Property 4.12. Each machine processes exactly one triple-job, one X-job, one Y-job, and one Z-job.

Proof. This follows immediately from the values in the first four coordinates and the previous property. ■

Property 4.13. Element-jobs assigned to the same machine are either all *real* jobs or all *dummy* jobs.

Proof. From Property 4.11 and the values in the fifth, sixth and seventh coordinate we see that the following three statements are simultaneously true:

1. There is exactly one real X-job or dummy Z-job (coordinate 5);
2. There is exactly one real Y-job or dummy X-job (coordinate 6);
3. There is exactly one real Z-job or dummy Y-job (coordinate 7).

The claim now follows by combining this with the fact that by Property 4.12 there is exactly one (real or dummy) job of each of the types X, Y and Z. ■

Property 4.14. If a machine processes the triple-job (i, j, k) and a (real or dummy) element-job a, then a is equal to i, j or k, depending on whether a is an X, Y or Z-job.

Proof. We only consider the case that a is an X-element; the other cases are similar. By Property 4.11 and Property 4.12, we know that $\overline{\langle i \rangle} + \langle a \rangle = \mathbf{b}$. Therefore, $\langle a \rangle = \mathbf{b} - \overline{\langle i \rangle} = \mathbf{b} - (\mathbf{b} - \langle i \rangle) = \langle i \rangle$ and thus $a = i$. ■

If a machine processes three real element-jobs, then by the last property the corresponding three elements form a triple in the 3-DM instance. Let T' consist of all triples corresponding to the triple-jobs that are scheduled together with real elements. Then, the triples in T' have no overlap as there is only one real element-job corresponding to an element. Moreover, T' covers all elements, because all jobs, and therefore also all real element-jobs, need to be scheduled. ■

Therefore we have the following lemma.

Lemma 4.15. *Given an instance of* 3-Dimensional Matching *with* $|X| = |Y| = |Z| = q$, $T \subseteq X \times Y \times Z$, $b \in \mathbb{N}_+$, $b \geqslant 2$ *and* $\varepsilon = 1/(b+1)$. *Then, there is a polynomial time reduction to an instance of* Vector Scheduling *with* $4|T|$ *vectors in dimension* $d := 3\left\lceil \log_{(1/\varepsilon)} q \right\rceil + 7$. *Moreover, a* $(1+\varepsilon)$*-approximate solution to the* Vector Scheduling *instance defines a solution to the* 3-DM *problem.*

Thus, Lemma 4.15 in combination with the ETH and the reduction from 3-CNF-Sat to 3-Dimensional Matching yields the following theorem.

Theorem 4.1. *For any $\varepsilon > 0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{o\left((1/\varepsilon)^{d/3}\right)}(nd)^{\mathcal{O}(1)}\right)$ for* VECTOR SCHEDULING *in $d \geqslant d(\varepsilon)$ dimensions, unless the Exponential Time Hypothesis (ETH) fails.*

Proof. Suppose that there exists a $(1+\varepsilon)$-approximation for VECTOR SCHEDULING that runs in time $\mathcal{O}\left(\exp\left(o((1/\varepsilon)^{d/3})\right)(nd)^{\mathcal{O}(1)}\right)$. By Lemma 4.15 we get an $\mathcal{O}\left(2^{o(q)}|T|^{\mathcal{O}(1)}\right)$ time algorithm for 3-DM, which in turn implies an $\mathcal{O}\left(2^{o(M)}M^{\mathcal{O}(1)}\right)$ time algorithm for 3-CNF-SAT, which contradicts the ETH. ■

4.3.2 *Lower bound assuming NP has no subexponential time algorithms*

Lemma 4.15 also implies the following.

Theorem 4.2. *For any $\varepsilon > 0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{(1/\varepsilon)^{o(d)}}(nd)^{\mathcal{O}(1)}\right)$ for* VECTOR SCHEDULING *in $d \geqslant d(\varepsilon)$ dimensions, unless NP has subexponential time algorithms, i.e. NP $\subseteq \cap_{\varepsilon>0} DTIME(2^{n^{\varepsilon}})$.*

Proof. Any problem in NP of size n can be reduced to an NP-complete problem of size $n^{\mathcal{O}(1)}$ in polynomial time. In particular, any problem $\mathcal{P}$ in NP can be formulated as a 3-DM problem with at most n^c elements and triples, for some constant c.

Suppose, by contradiction, that there exists a $(1+\varepsilon)$-approximation for VECTOR SCHEDULING that runs in time $\mathcal{O}\left(\exp\left((1/\varepsilon)^{o(d)}\right)(nd)^{\mathcal{O}(1)}\right)$. Then setting d equal to $\log_{(1/\varepsilon)}(n^c) + 7 \leqslant c\log_{(1/\varepsilon)} n + 7$ gives an algorithm for $\mathcal{P}$ with running time $\mathcal{O}\left(\exp\left((1/\varepsilon)^{o(\log_{(1/\varepsilon)}(n))}\right)n^{\mathcal{O}(1)}\right) = \mathcal{O}\left(\exp\left(n^{o(1)}\right)\right)$, which is subexponential. ■

4.3.3 *Lower bound with resource augmentation*

In this subsection we show a lower bound on the running time of $(1+\varepsilon)$-approximation algorithms for VECTOR SCHEDULING that are allowed *resource augmentation*, i.e. besides exceeding the optimal load by a factor $(1+\varepsilon)$, it is also allowed to use εm extra machines.

To show this, we reduce from a stricter version of 3-DIMENSIONAL MATCHING, namely 3-DIMENSIONAL MATCHING-B, abbreviated as 3-DM-B. In this problem we are given a set of triples $T \subseteq X \times Y \times Z$, where X, Y and Z are

Job name	Values of the coordinates				
	T/X/Y/Z	Real/dummy	Encoding of element(s)		
real X-job i:	$0, b, 0, 0$	$b, 0, 0$	$\langle i\rangle_1, \ldots, \langle i\rangle_\ell, \overline{\langle i\rangle}_1, \ldots, \overline{\langle i\rangle}_\ell$	$0, \ldots, 0$	$0, \ldots, 0$
dummy X-job i:	$0, b, 0, 0$	$0, b, 0$	$\langle i\rangle_1, \ldots, \langle i\rangle_\ell, \overline{\langle i\rangle}_1, \ldots, \overline{\langle i\rangle}_\ell$	$0, \ldots, 0$	$0, \ldots, 0$
triple (i, j, k):	$b, 0, 0, 0$	$0, 0, 0$	$\overline{\langle i\rangle}_1, \ldots, \overline{\langle i\rangle}_\ell, \langle i\rangle_1, \ldots, \langle i\rangle_\ell$	$\overline{\langle j\rangle}$	$\overline{\langle k\rangle}$

Table 4.2: New construction of the X-jobs and triple-jobs of the 3-DM-3 problem.

disjoint finite sets and each element in $X \cup Y \cup Z$ appears at most B times in the triples of T. The goal is to find a subset of triples T' that maximizes the number of elements in $X \cup Y \cup Z$ that appear exactly once in T'.

Theorem 4.16 (Petrank, [85])**.** *For* 3-Dimensional Matching-3 *it is NP-hard to distinguish between instances where all elements can be covered by disjoint triples and those instances where at most a* $(1 - \varepsilon_{3\text{-DM}})$ *fraction of the elements can be covered by disjoint triples, where* $\varepsilon_{3\text{-DM}} < 1$ *is some universal constant.*

Using this result we prove the following lemma.

Lemma 4.17. *For* Vector Scheduling *in* $d \geqslant d_0$ *dimensions it is NP-hard to distinguish between instances where all jobs can be scheduled on* m *machines with maximum load* 1 *and those instances where all jobs can be scheduled on* $(1 + \varepsilon_0)m$ *machines with maximum load* $1 + \varepsilon_0$*, where* $0 < \varepsilon_0 < 1$ *and* $d_0 \geqslant 1$ *are some universal constants and* $1/\varepsilon_0$ *is integer.*

Proof. Construct a Vector Scheduling instance from the 3-DM-3 problem in almost the same way as for 3-DM. The only difference is that for every (real or dummy) X-job i and triple (i, j, k), instead of only encoding $\langle i\rangle$ respectively $\overline{\langle i\rangle}$, we append this by encoding $\overline{\langle i\rangle}$ respectively $\langle i\rangle$ (all other jobs get extra zero-entries). See Table 4.2. Consequently, if a triple (i, j, k) is scheduled on a machine where also an X-job x is scheduled, then $i = x$. Previously we established this through the fact that the load in each coordinate is exactly b. However, here we do not have this property because of the extra machines.

There are at most $3q$ triples, where $q = |X| = |Y| = |Z|$. One direction is clear, if all $3q$ elements can be covered by disjoint triples then there is a

schedule of height at most b on $m = 3q$ machines. For the other direction, suppose we found a $(1+\varepsilon)$-approximate solution with $\varepsilon 3q$ extra machines. Using the same reasoning as before, we now have the following properties:

- The maximum load is b;
- On each machine there is at most one triple, one X-, one Y-, and one Z-job;
- On each machine, if there are three element-jobs, then all three are *real* jobs or all three are *dummy* jobs;
- If a triple (i, j, k) and an X-job x, Y-job y and Z-job z are scheduled on the same machine, then $i = x$, $j = y$ and $k = z$.

Therefore, every machine on which a triple and three real elements are scheduled, corresponds to a triple in the solution to the 3-DM-3 problem.

We will now show that there is a universal constant such that it is hard to distinguish between instances where everything fits on m machines with maximum load 1 and instances where everything fits on $(1+\varepsilon)m$ machines with maximum load $1+\varepsilon$. Consider the $\varepsilon 3q$ machines without a triple. These $\varepsilon 3q$ machines contain at most $\varepsilon 9q$ element-jobs. Considering that there are $3q$ machines on which $9q - \varepsilon 9q$ element-jobs must be scheduled together with triples, there are at most $\varepsilon 9q$ machines *with* a triple but with at most two elements. Hence, there are at most $\varepsilon 9q + 2(\varepsilon 9q)$ real elements that are scheduled on either a machine without a triple, or with a triple but with only one other element. Therefore, at least $3q - \varepsilon 27q$ real elements are scheduled together with triples, which corresponds to $q - 9\varepsilon q$ disjoint triples that cover $3q - 27\varepsilon q$ elements. If $27\varepsilon < \varepsilon_{3\text{-DM}}$, we found a solution where more than a $(1 - \varepsilon_{3\text{-DM}})$ fraction of the elements are covered in the 3-DM-3 instance, which is NP-hard. ■

Following the proof of Theorem 4.2, this immediately implies the following.

Theorem 4.3. *For any $\varepsilon < \varepsilon_0$ with $1/\varepsilon \in \mathbb{N}$, there is a $d(\varepsilon)$ such that there is no $(1+\varepsilon)$-approximation algorithm with running time $\mathcal{O}\left(2^{(1/\varepsilon)^{o(d)}}(nd)^{\mathcal{O}(1)}\right)$ for* Vector Scheduling *in $d \geqslant d(\varepsilon)$ dimensions, even with $\lceil \varepsilon m \rceil$ extra machines, unless NP has subexponential time algorithms, i.e. NP $\subseteq \cap_{\varepsilon>0} DTIME(2^{n^\varepsilon})$, where $\varepsilon_0 < 1$ is a universal constant. Assuming the ETH, no such algorithm can run in time $\mathcal{O}\left(2^{\left(o(1/\varepsilon)^{d/6}\right)}(nd)^{\mathcal{O}(1)}\right)$.*

4.4 LINEAR TIME APPROXIMATION ALGORITHM

In this section we describe our linear time algorithm, see Algorithm 4.1. Roughly, it works as follows. First, we preprocess the instance such that there are relatively few different types of large jobs at the cost of a small factor in the approximation guarantee. Next, we formulate and solve a mixed-integer linear program that assigns large jobs integrally to machines and small jobs fractionally. In the randomized algorithm, we assign the small jobs according to the probabilities obtained from the MILP and redistribute the small jobs on the overloaded machines over the other machines in such a way that no machine is overloaded. In the deterministic algorithm, we derandomize this step by assigning the small jobs integrally to machines in a greedy manner guided by a potential function that tracks the aggregate overload on the machines. Finally, we distribute this overload evenly over all machines ensuring the final loads of all machines is at most $1 + \varepsilon$.

Algorithm 4.1: Linear time algorithm for VECTOR SCHEDULING.

1 Preprocess the instance ;
2 Solve the (MILP), and assign big jobs according to this solution ;
3 Assign small jobs to machines randomly according to the probabilities obtained from the (MILP) solution ;
4 Remove small jobs from the overloaded machines and evenly distribute them over all machines ;

4.4.1 *Preprocessing*

The preprocessing uses the same ideas used before in the design of approximation schemes. Typically, it is much easier to work with a few distinct jobs as we will see in the the formulation of our mixed-integer linear program. See Algorithm 4.2.

Algorithm 4.2: Preprocessing for Algorithm 4.1.

1 Round each coordinate of every job down to the nearest power of $(1 + \varepsilon)$ times ε^4/d^2 (Lemma 4.18) ;
2 Set coordinates of jobs that are small in comparison to the biggest coordinate to zero (Lemma 4.19) ;

The first step is to round all coordinates of each job down to the nearest power of $(1+\varepsilon)$ times a small polynomial in ε and $1/d$.

Lemma 4.18 ([23]). *Given a set V of jobs and $\varepsilon > 0$, let W be a modified set of V where we replace each job $\mathbf{v}$ in V with a job $\mathbf{w}$ as follows:*

$$w_j := \begin{cases} \varepsilon^4/d^2 \cdot (1+\varepsilon)^k & \text{if } \exists k \in \mathbb{N}: \\ & \varepsilon^4/d^2 \cdot (1+\varepsilon)^k \leqslant v_j < \varepsilon^4/d^2 \cdot (1+\varepsilon)^{k+1}, \\ 0 & \text{otherwise.} \end{cases}$$

Then, for any subset of jobs $V' \subseteq V$ with corresponding subset $W' \subseteq W$, we have $\sum_{\mathbf{v}\in V'} \mathbf{v} \leqslant (1+\varepsilon) \sum_{\mathbf{w}\in W'} \mathbf{w}$.

Next, we ensure that the non-zero values of coordinates of a job are not too small compared to the largest coordinate of a job.

Lemma 4.19 ([23]). *Given a set V of jobs and $\eta > 0$, let W be a modified set of V where we replace each job $\mathbf{v}$ in V with a job $\mathbf{w}$ as follows:*

$$w_j := \begin{cases} 0 & \text{if } v_j < \eta \|\mathbf{v}\|_\infty, \\ v_j & \text{otherwise.} \end{cases}$$

Then, for any subset of jobs $V' \subseteq V$ with corresponding subset $W' \subseteq W$, we have $\sum_{\mathbf{v}\in V'} \mathbf{v} \leqslant \sum_{\mathbf{w}\in W'} \mathbf{w} + \left(\eta \sum_{\mathbf{w}\in W'} \|\mathbf{w}\|_\infty\right) \mathbf{1}$.

The following lemma states that the error due to the preprocessing of any schedule is small, and follows from the previous lemmata, setting $\eta := \varepsilon/d$.

Lemma 4.20. *Let V be the original set of jobs and W be the preprocessed set of jobs and $\varepsilon > 0$. Then for any $\mathbf{w} \in W$ and coordinate $i \in [d]$,*

- *if $w_i \neq 0$ then there exists a $k \in \mathbb{N}$ such that $w_i = \varepsilon^4/d^2 \cdot (1+\varepsilon)^k$,*
- *if $w_i \neq 0$ then $w_i/\|\mathbf{w}\|_\infty \geqslant \varepsilon/d$.*

Moreover, for any subset of jobs $V' \subset V$ such that $\sum_{\mathbf{v}\in V'} \mathbf{v} \leqslant \mathbf{1}$ with corresponding modified subset $W' \subseteq W$, we have $\sum_{\mathbf{w}\in W'} \mathbf{w} \leqslant \sum_{\mathbf{v}\in V'} \mathbf{v} \leqslant (1+\varepsilon) \sum_{\mathbf{w}\in W'} \mathbf{w} + \varepsilon\mathbf{1}$.

From now on, by job we mean the preprocessed, rounded job.

4.4.2 *The mixed-integer linear program*

In this subsection we describe our mixed-integer linear program and how to solve it fast. We distinguish between *small* and *big* jobs and treat them differently. A job $\mathbf{p}$ is *small* if $\|\mathbf{p}\|_\infty < \varepsilon^3/d$ and otherwise the job is *big*.

As all non-zero coordinates are at most a factor d/ε apart by Lemma 4.20, the smallest possible coordinate of any big job is ε^4/d^2. Let $\mathcal{T}_{\text{big}}$ be the set of all *types* of big jobs, $\mathcal{T}_{\text{big}} := \{0, \varepsilon^4/d^2, (1+\varepsilon)\varepsilon^4/d^2, (1+\varepsilon)^2\varepsilon^4/d^2, \dots, 1\}^d$. A big job $\mathbf{p}$ has type $\mathbf{t} \in \mathcal{T}_{\text{big}}$ if and only if $\mathbf{p} = \mathbf{t}$. Every big job has a corresponding type, since the rounding procedure rounded these jobs to exactly these values.

Similarly, we define a set $\mathcal{T}_{\text{small}}$ of all types of small jobs. We define the type of a small job based on its relative size in each coordinate, that is, a small job $\mathbf{p}$ has type $\mathbf{t} = (t_1, \dots, t_d) \in \mathcal{T}_{\text{small}}$ if and only if $p_j/\|\mathbf{p}\|_\infty = t_j$ for all coordinates $j \in [d]$. As the smallest non-zero coordinate in $\mathbf{p}/\|\mathbf{p}\|_\infty$ is at least ε/d, we define $\mathcal{T}_{\text{small}} := \{0, (1+\varepsilon)^{-\ell}, (1+\varepsilon)^{-\ell+1}, \dots, (1+\varepsilon)^{-1}, 1\}^d$, where $\ell := \left\lceil \log_{(1+\varepsilon)}(d/\varepsilon) \right\rceil$ is such that $(1+\varepsilon)^{-\ell}$ is the smallest power of $1+\varepsilon$ that is at least ε/d. Note that each small job has exactly one type in $\mathcal{T}_{\text{small}}$ and that there are at most $T := \left\lceil 4\log_{(1+\varepsilon)}(d/\varepsilon) + 2 \right\rceil^d$ types of big and small jobs.

The mixed-integer linear programming has a variable for every configuration, which is a collection of big jobs together with available space for small jobs. We will call the (rounded) space for small jobs a *profile*, which is a vector from $\mathcal{F} := \{0, \varepsilon, (1+\varepsilon)\varepsilon, (1+\varepsilon)^2\varepsilon, \dots, 1\}^d$. A *configuration* C is a tuple $C = (B, \mathbf{f})$, where B is a multiset of rounded processing times of big jobs and $\mathbf{f}$ is a profile for small jobs such that the big jobs and the profile fit together on one machine, exceeding the maximum load by only a little, i.e. $(\sum_{\mathbf{p}\in B} p_j) + f_j \leqslant (1+\varepsilon)$ for all coordinates j. As each big job has a coordinate of at least ε^3/d, there can be no more than d^2/ε^3 big jobs on a machine. As there at most T types of big jobs, this implies that there are at most $N \leqslant T^{\lceil d^2/\varepsilon^3 \rceil} \cdot T$ different configurations.

We now describe our mixed-integer linear program. Let $\mathcal{C}$ be the set of all configurations and let x_C denote the number of machines that have jobs assigned to them according to configuration $C \in \mathcal{C}$. Let $n(C, \mathbf{t})$ denote the number of big jobs of type $\mathbf{t}$ in configuration C, and let $n(\mathbf{t})$ denote the total number of big jobs of type $\mathbf{t}$ in the instance. Denote the set of small jobs of type $\mathbf{t}$ assigned to configurations having profile $\mathbf{f}$ by $J(\mathbf{f}, \mathbf{t})$, and define the variables $y_{\mathbf{f},\mathbf{t}} = \sum_{\mathbf{p}\in J(\mathbf{f},\mathbf{t})} \|\mathbf{p}\|_\infty$ as the sum of their largest coordinates,

their *amount*. Let $a(\mathbf{t}) := \sum_{\mathbf{p}:\mathbf{p} \text{ is of small type } \mathbf{t}} \|\mathbf{p}\|_\infty$ denote the total amount of small jobs of type $\mathbf{t}$ in the instance. Consider the following program.

$$\begin{aligned}
\min \quad & \sum_{C \in \mathcal{C}} x_C && && \text{(MILP)}\\
\text{s.t.} \quad & \sum_{C \in \mathcal{C}} x_C \cdot n(C,\mathbf{t}) \geqslant n(\mathbf{t}) && \forall \mathbf{t} \in \mathcal{T}_{\text{big}} && \text{(C1)}\\
& \sum_{\mathbf{f} \in \mathcal{F}} y_{\mathbf{f},\mathbf{t}} \geqslant a(\mathbf{t}) && \forall \mathbf{t} \in \mathcal{T}_{\text{small}} && \text{(C2)}\\
& \sum_{\mathbf{t} \in \mathcal{T}_{\text{small}}} y_{\mathbf{f},\mathbf{t}} \cdot t_i / \|\mathbf{t}\|_\infty \leqslant f_i \cdot \sum_{C : C = (B,\mathbf{f})} x_C && \forall i \in [d], \mathbf{f} \in \mathcal{F} && \text{(C3)}\\
& \mathbf{x} \in \mathbb{Z}^{\mathcal{C}}\\
& \mathbf{y}, \mathbf{x} \geqslant 0
\end{aligned}$$

The first and second constraint ensure that the big and the small jobs are covered integrally respectively fractionally. The third constraint ensures that small jobs fit in the machine profiles, as it requires that for each profile $\mathbf{f}$, the cumulative amount of small jobs of type $\mathbf{t}$ that are assigned to $\mathbf{f}$ is at most the total amount of $\mathbf{f}$. These are valid constraints for any feasible solution.

Lemma 4.21. *An optimal solution to* (MILP) *can be found in time* $\mathcal{O}\left(\exp\left((1/\varepsilon)^{\mathcal{O}(d \log\log d)}\right) \cdot \log(nd)\right)$.

Proof. First, we bound the number of choices for non-zero integer variables. To do that, suppose that there is a finite solution and suppose that the continuous variables $y_{\mathbf{f},\mathbf{t}}$ are fixed: this allows us to disregard constraints (C2), only containing continuous variables. Then introduce slack variables such that all constraints are equality constraints and the (MILP) matches the form of Theorem 4.8. For the application of this theorem we can disregard the non-negativity constraints [31]. Thus, we are left with at most $|\mathcal{T}_{\text{big}}| + d|\mathcal{F}| \leqslant (d+1)T$ constraints. The largest size of the coefficients are the constants $n(C,\mathbf{t})$, t_i, $\|\mathbf{t}\|_\infty$ and f_i, all of which require at most d^2/ε^3 bits to describe. By Theorem 4.8 there is an optimal solution such that there are at most $2((d+1)T+1)\left(\log((d+1)T+1) + d^2/\varepsilon^3 + 2\right)$ non-zero integer variables. As

$$\begin{aligned}
\log((d+1)T+1) &= \log\left((d+1)\left\lceil 4\log_{(1+\varepsilon)}(d/\varepsilon) + 2\right\rceil^d + 1\right)\\
&\leqslant d\log\left(4\log_{(1+\varepsilon)}(d/\varepsilon)\right) \leqslant d^2/\varepsilon^3,
\end{aligned}$$

the number of non-zero integer variables is at most

$$2\left((d+1)T+1\right)\left(2d^2/\varepsilon^3+2\right) = 4\left(\left((d^3+d^2)/\varepsilon^3+d+1\right)T+d^2/\varepsilon^3+1\right)$$
$$\leqslant 8\left(\left(d^3/\varepsilon^3+d\right)T+d^2/\varepsilon^3\right) \leqslant 16Td^3/\varepsilon^3\,.$$

Therefore, we can bound the number of choices for non-zero variables by

$$N^{16Td^3/\varepsilon^3} \leqslant \left(T^{d^2/\varepsilon^3+2}\right)^{16Td^3/\varepsilon^3} = 2^{(d^2/\varepsilon^3+2)16T\log T d^3/\varepsilon^3}\,.$$

Using that $T\log T \leqslant T^2$ and plugging in the definition of T, we bound this by

$$2^{(16d^6/\varepsilon^6)T^2} = \exp\left(16d^6/\varepsilon^6\left\lceil 3\log_{(1+\varepsilon)}(d/\varepsilon)+2\right\rceil^{2d}\right)\,.$$

As the first part is $(1/\varepsilon)^{\mathcal{O}(\log(d))}$ and the second part is $(1/\varepsilon)^{\mathcal{O}(d\log\log d)}$, the number of choices for non-zero variables is at most $\exp\left((1/\varepsilon)^{\mathcal{O}(d\log\log d)}\right)$.

Now we 'guess' the non-zero variables by enumerating all their possible choices, and solve the (MILP) for only those variables. Since there are at most $16Td^3/\varepsilon^3$ variables, by Theorem 4.7 solving the (MILP) takes time $\mathcal{O}\left((16Td^3/\varepsilon^3)^{40Td^3/\varepsilon^3}\cdot s\right)$, where s denotes the maximum length of the binary encoding of the mixed-integer linear program. Using the same rewriting as above, we can rewrite this to $\mathcal{O}\left(\exp\left((1/\varepsilon)^{\mathcal{O}(d\log\log d)}\right)\cdot s\right)$. As the mixed-integer linear program can be described using $s = TN\log(nd)$ bits, the proof is complete. ■

4.4.3 *Randomized algorithm*

In this subsection we sketch step 3 and 4 of Algorithm 4.1, the integral assignment of small jobs to machines using the solution to (MILP).

For step 3, recall that $y_{\mathbf{f},\mathbf{t}}$ is the amount of small jobs of type $\mathbf{t}$ that are assigned to profile $\mathbf{f}$. For each small job type $\mathbf{t}$, let $\beta(\mathbf{f},\mathbf{t})$ denote the fraction of type $\mathbf{t}$ assigned to profile $\mathbf{f}$:

$$\beta(\mathbf{f},\mathbf{t}) := \frac{y_{\mathbf{f},\mathbf{t}}}{\sum_{\mathbf{g}\in\mathcal{F}} y_{\mathbf{g},\mathbf{t}}}\,.$$

For each small job $\mathbf{p}$ of type $\mathbf{t}$, pick a profile $\mathbf{f}$ randomly with probability $\beta_{\mathbf{f},\mathbf{t}}$ and then pick a machine uniformly at random among the ones with profile $\mathbf{f}$. Assign job $\mathbf{p}$ to this machine.

For step 4, we call a machine with profile $\mathbf{f}$ *overloaded* if the load of small jobs exceeds $\mathbf{f} + \varepsilon$ in some coordinate. We take all the small jobs on overloaded machines and distribute them among all machines using a linear time simple sequential assignment. We will prove that the probability that the load on a machine in a coordinate exceeds the profile by more than ε is exponentially small. This implies that the expected overload on each machine is small, hence, the total overload over all the machines is small.

For the following proofs we fix a machine. Define for each small job $\mathbf{p}$ and coordinate j the random variables $X_{\mathbf{p}}^j$ with $\mu_{\mathbf{p}}^j$ as its mean, which is the contribution of job $\mathbf{p}$ to the j-th coordinate of the machine:

$$X_{\mathbf{p}}^j = \begin{cases} p_j, & \text{if job } \mathbf{p} \text{ is assigned to the machine,} \\ 0, & \text{otherwise.} \end{cases}$$

Let X_j denote the load of small jobs coordinate j on the machine i.e. $X_j := \sum_{\text{small job } \mathbf{p}} X_{\mathbf{p}}^j$. We need the following Bernstein's inequality.

Theorem 4.22. *Let $X_1, \ldots, X_n$ be independent random variables with $\mathbb{E}[X_i] = \mu_i$ and $|X_i - \mu_i| \leqslant M$ for all i. Let $\mu = \sum_i \mathbb{E}[X_i]$ and $\sigma_i^2 = \mathbb{E}\left[(X_i - \mu_i)^2\right]$. Then for any $t > 0$, it holds that*

$$\mathbb{P}\left(\sum_i X_i > \mu + t\right) \leqslant \exp\left(-\frac{t^2/2}{(\sum_i \sigma_i^2) + Mt/3}\right).$$

We first show that the probability that the load of small jobs exceeds the profile in a coordinate is small.

Lemma 4.23. *For any machine with profile $\mathbf{f}$ and $0 < \varepsilon < 1$ we have*

$$\mathbb{P}\left(X_j \geqslant f_j + \varepsilon\right) \leqslant e^{-\varepsilon^2/4\delta} \text{ for all coordinates } j \in [d],$$

where δ is the maximum coordinate of any small job.

Proof. Let $m(\mathbf{f})$ denote the number of machines with profile $\mathbf{f}$. Since a job $\mathbf{p}$ of type $\mathbf{t}$ is assigned to this machine with probability $\beta(\mathbf{f}, \mathbf{t})/m(\mathbf{f})$, the expected load on coordinate j on that machine is

$$\sum_{\mathbf{t} \in \mathcal{T}_{\text{small}}} \sum_{\mathbf{p}:\mathbf{p} \text{ of type } \mathbf{t}} p_j \frac{\beta(\mathbf{f}, \mathbf{t})}{m(\mathbf{f})} = \sum_{\mathbf{t} \in \mathcal{T}_{\text{small}}} \sum_{\mathbf{p}:\mathbf{p} \text{ of type } \mathbf{t}} p_j \frac{y_{\mathbf{f},\mathbf{t}}}{m(\mathbf{f})(\sum_{\mathbf{g} \in \mathcal{F}} y_{\mathbf{g},\mathbf{t}})}.$$

Recall that f_j is the space available for small jobs in coordinate j of profile $\mathbf{f}$ and $a(\mathbf{t})$ is the amount of small jobs of type $\mathbf{t}$. Therefore, $\sum_{\mathbf{p}:\mathbf{p} \text{ of type } \mathbf{t}} p_j = a(\mathbf{t})t_j/\|\mathbf{t}\|_\infty$, and hence the expected load is at most

$$\sum_{\mathbf{t}\in\mathcal{T}_{\text{small}}} \frac{a(\mathbf{t})y_{\mathbf{f},\mathbf{t}}t_j}{\|\mathbf{t}\|_\infty m(\mathbf{f}) \sum_{\mathbf{g}\in\mathcal{F}} y_{\mathbf{g},\mathbf{t}}} \leqslant \sum_{\mathbf{t}\in\mathcal{T}_{\text{small}}} \frac{t_j y_{\mathbf{f},\mathbf{t}}}{\|\mathbf{t}\|_\infty m(\mathbf{f})} \leqslant f_j\,.$$

Both inequalities follow from the (MILP) constraints: the first follows as $\sum_{\mathbf{f}\in\mathcal{F}} y_{\mathbf{f},\mathbf{t}} \geqslant a(\mathbf{t})$ and the second follows as $\sum_{\mathbf{t}\in\mathcal{T}_{\text{small}}} y_{\mathbf{f},\mathbf{t}}t_j/\|\mathbf{t}\|_\infty \leqslant m(\mathbf{f})f_j$.

We now apply Bernstein's inequality to our setting. We have that

$$\begin{aligned}(\sigma_{\mathbf{p}}^j)^2 &:= \mathbb{E}\left[\left(X_{\mathbf{p}}^j - \mu_{\mathbf{p}}^j\right)^2\right] \\ &= \left(p_j - p_j\frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})}\right)^2 \frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})} + \left(0 - p_j\frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})}\right)^2 \left(1 - \frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})}\right) \\ &= p_j^2\left(1 - \frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})}\right)\frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})} \leqslant (p_j)^2\frac{\beta(\mathbf{f},\mathbf{t})}{m_{\mathbf{f}}}\,.\end{aligned}$$

Thus,

$$\sum_{\text{small job } \mathbf{p}} (\sigma_{\mathbf{p}}^j)^2 \leqslant \left(\max_{\text{small job } \mathbf{p}} p_j\right) \sum_{\text{small job } \mathbf{p}} \frac{\beta(\mathbf{f},\mathbf{t})}{m(\mathbf{f})} p_j \leqslant \delta f_j \leqslant \delta\,.$$

Moreover $|X_{\mathbf{p}}^j - \mu_{\mathbf{p}}^j| \leqslant \delta$, so choose $M = \delta$. Then

$$\mathbb{P}\left(X_j > f_j + x\right) \leqslant \exp\left(\frac{-x^2/2}{\delta + \delta x/3}\right)\,.$$

For $x \leqslant 3$, we bound this by $\exp(-x^2/(4\delta))$. For $x \geqslant 3$, we bound this by

$$\exp\left(-\frac{x^2/2}{2\delta x/3}\right) = \exp(-3x/4\delta)\,.$$

So, for any $\varepsilon \leqslant 1$, $\mathbb{P}\left(X_j \geqslant f_i + \varepsilon\right) \leqslant e^{-\varepsilon^2/4\delta}$. ■

We now bound $\mathbb{E}\left[X_j | X_j \geqslant f_i + \varepsilon\right]\mathbb{P}\left(X_j \geqslant f_i + \varepsilon\right)$, i.e. the average load on overloaded machines.

Lemma 4.24. *For any machine with profile* $\mathbf{f}$ *and* $0 < \varepsilon < 1/5$ *we have*

$$\mathbb{E}\left[X_j | X_j \geqslant f_j + \varepsilon\right]\mathbb{P}\left(X_j \geqslant f_j + \varepsilon\right) \leqslant 2\varepsilon^3/d^3 \text{ for all coordinates } j\,.$$

Proof. Recall that for any non-negative random variable Y with finite mean,

$$\mathbb{E}[Y] = \int_0^\infty y\mathbb{P}(Y = y)\,dy = \int_0^\infty \mathbb{P}(Y \geqslant y)\,dy\,.$$

where the last equality follows from integration by parts. This implies that

$$\mathbb{E}[Y|(Y > t)] = t + \frac{1}{\mathbb{P}(Y \geqslant t)} \int_{y=0}^\infty \mathbb{P}(Y \geqslant t + y)\,dy\,.$$

Applying this to our setting, we get

$$\mathbb{E}\left[X_j | X_j \geqslant (f_j + \varepsilon)\right] \mathbb{P}\left(X_j \geqslant f_j + \varepsilon\right) \leqslant (f_j + \varepsilon)\mathbb{P}\left(X_j \geqslant f_j + \varepsilon\right) + \int_{x=0}^\infty \mathbb{P}\left(X_j > f_i + \varepsilon + x\right) dx.$$

As $f_i \leqslant 1$ and by the proof of Lemma 4.23, this is at most

$$(1 + \varepsilon)\mathbb{P}\left(X_j \geqslant f_i + \varepsilon\right) + \int_\varepsilon^3 \exp(-x^2/4\delta)\,dx + \int_3^\infty \exp(-3x/4\delta)\,dx\,, \quad (4.1)$$

where $\delta := \max_{\mathbf{p}:\mathbf{p} \text{ small job}} \|\mathbf{p}\|_\infty$ is the maximum coordinate of any small job. The last term is $(4\delta/3)\exp(-9/4\delta)$. The second term can be upper bounded by $\int_\varepsilon^\infty \exp(-x^2/4\delta)\,dx$. Let $f(x) = \frac{1}{\sqrt{2\pi}}\exp(-x^2/2)$ denote the pdf of the standard gaussian $N(0,1)$. Let $\overline{\Phi}(x) = \int_x^\infty f(x)\,dx$. Using that $\overline{\Phi}(x) \leqslant f(x)/x$ for any $x > 0$, it follows that

$$\int_\varepsilon^\infty \exp(-x^2/4\delta)\,dx = \sqrt{2\delta}\int_{\varepsilon/\sqrt{2\delta}}^\infty e^{-y^2/2}\,dy \leqslant (2\delta\sqrt{2\pi}/\varepsilon)\exp(-\varepsilon^2/4\delta)\,.$$

We plug this in Eq. (4.1), bounding δ by $\varepsilon^2/(4\ln(d/\varepsilon))$, which is larger than ε^3/d if $d/\varepsilon \geqslant 9$. As $d \geqslant 2$ and $\varepsilon < 1/5$ this is a valid upper bound for all small jobs. This yields that the total expected load on overloaded machines is at most

$$\left(1 + \varepsilon + 2\delta\sqrt{2\pi}/\varepsilon\right)\exp(-\varepsilon^2/4\delta) + (4\delta/3)\exp(-9/4\delta) = \left(1 + \varepsilon + \frac{\varepsilon\sqrt{2\pi}}{2\ln(d/\varepsilon)}\right)\varepsilon/d + \frac{\varepsilon^2}{3\ln(d/\varepsilon)}(\varepsilon/d)^{9/\varepsilon^2}\,.$$

This is at most $2\varepsilon/d$. ∎

We can now prove the following theorem.

Theorem 4.25. *There is an algorithm that runs in* $\mathcal{O}\left(2^{(1/\varepsilon)^{\mathcal{O}(d \log\log d)}} + nd\right)$ *time and finds a schedule such that the load on each machine is at most* $1+\varepsilon$ *with high probability.*

Proof. Let $\varepsilon' := \varepsilon/9$. First we prove the approximation ratio. For an overloaded machine k, let L_k be the sum of the ℓ_1-norm of all small jobs assigned to k, and let L be the sum of the ℓ_1-norm of all small jobs on all overloaded machines. By Lemma 4.24 we know that $\mathbb{E}[L_k] \leqslant 2\varepsilon'$ for all machines k and thus, by linearity of expectation, $\mathbb{E}[L] \leqslant 2m\varepsilon'$. Therefore $\mathbb{P}(L > 4m\varepsilon') < 1/2$ by Markov's inequality. Remove all small jobs assigned from the overloaded machines and order them arbitrarily. Greedily group them together until the ℓ_1-norm exceeds $4\varepsilon'$ and then start a new group. Every group has size at most $4\varepsilon' + \delta$. Now assign every group to a non-overloaded machine. The small jobs on the overloaded machines have now been redistributed such that the extra load on every machine is in expectation at most the average plus the largest small job size, i.e. $4\varepsilon' + \delta \leqslant 4\varepsilon' + \varepsilon'^3/d \leqslant 5\varepsilon'$. All other machines exceeded their profile in each coordinate by at most ε'. Additionally, from the mixed-integer linear program we lost another ε' since we only required that the big jobs and the profile add up to at most $1+\varepsilon'$. This gives a total of $7\varepsilon'$ on the preprocessed instance and factoring in the preprocessing we get $(1+\varepsilon')(7\varepsilon') + \varepsilon' \leqslant 9\varepsilon' = \varepsilon$.

The preprocessing and randomized rounding steps can be implemented in $\mathcal{O}(nd)$ time. To bound the time of solving (MILP), we use the fact that $ab \leqslant a^2 + b^2$. Choosing $a = 2^{(1/\varepsilon')^{\mathcal{O}(d \log\log d)}}$ and $b = \log(nd)$, we get $\mathcal{O}\left(2^{(1/\varepsilon')^{\mathcal{O}(d \log\log d)}} \log(nd)\right) \leqslant \mathcal{O}\left(2^{2(1/\varepsilon')^{\mathcal{O}(d \log\log d)}} + \log^2(nd)\right)$, so the total running time is at most $\mathcal{O}\left(2^{(1/\varepsilon')^{\mathcal{O}(d \log\log d)}} + nd\right)$. ∎

By simply repeating the rounding and grouping step until a solution is found, we get an $\mathcal{O}(nd)$ time algorithm for assigning small jobs that returns a $(1+\varepsilon)$-approximation with high probability.

4.4.4 *Deterministic algorithm*

Recall that the (MILP) only gives an assignment of small job types to profiles, while we need an assignment of individual jobs to machines for a deterministic algorithm. This can be done in three steps using standard techniques. First, small job types are assigned integrally to profiles. Then,

using a pessimistic estimator, small jobs are integrally assigned to machines having a fixed profile. Finally, a direct calculation shows that the total load on overloaded machines is at most $\mathcal{O}(\varepsilon m/d)$, so the small jobs from these machines can be redistributed over all machines in a round-robin fashion without increasing the loads too much. We will now describe every step in more detail and prove its correctness.

(STEP 1) We first assign small job types to profiles. To this end, observe that as there are few constraints involving y variables, by standard polyhedral arguments there are few non-zero y variables, that are distributed over many profiles. By greedily assigning small jobs to profiles as long as they fit with respect to $y_{f,t}$, only few small jobs remain, which can then be evenly divided over the profiles only causing an $\mathcal{O}(\varepsilon)$ increase in the loads.

Lemma 4.26 (Step 1)**.** *Given the LP solution, we can integrally assign small jobs to profiles such that the total load of each machine on each coordinate is increased by at most* $\mathcal{O}(\varepsilon)$*.*

Proof. Without loss of generality assume that for every profile there is a configuration that uses that profile and that there is at least one small job of each type, i.e. $\forall \mathbf{t} \in \mathcal{T}_{\text{small}} : a(\mathbf{t}) > 0$. Furthermore, let $\mathcal{T}'_{\text{small}}$ denote the set of types where the jobs are assigned to multiple machines.

There are $|\mathcal{F}|d + |\mathcal{T}_{\text{small}}|$ constraints involving variables y. By fixing the integer variables x_C, we note that at most $d|\mathcal{F}| + |\mathcal{T}_{\text{small}}|$ variables y are non-zero by classical polyhedral theory and further that the number of variables y that are between zero and $a(\mathbf{t})$ is bounded by $d|\mathcal{F}| + |\mathcal{T}'_{\text{small}}|$.

Let $q(\mathbf{t})$ be the number of variables $y_{f,\mathbf{t}}$ between zero and the maximum $a(\mathbf{t})$, i.e., $q(\mathbf{t}) := |\{y_{f,\mathbf{t}} : 0 < y_{f,\mathbf{t}} < a(\mathbf{t})\}|$. Since at least each constraint (C2) has two non-zero y variables we have

$$2|\mathcal{T}'_{\text{small}}| \leqslant \sum_{\mathbf{t} \in \mathcal{T}'_{\text{small}}} q(\mathbf{t}) \leqslant d|\mathcal{F}| + |\mathcal{T}'_{\text{small}}| \leqslant 2d|\mathcal{F}|\,.$$

Let $\mathbf{t}$ be a type, for convenient notation let $1 < q = q(\mathbf{t})$ and let $\mathbf{f_1}, \ldots, \mathbf{f_q}$ be profiles such that $0 < y_{\mathbf{f_k},\mathbf{t}}$ for all $k \in [q]$. Let S be the set of all small vectors of type $\mathbf{t}$.

The goal is to assign jobs S to the q profiles and have a small left-over set of jobs L. Let T_k be the set of small jobs that we assign to profile $\mathbf{f_k}$ in this procedure. Initially set $L := S$ and $T_1 = T_2 = \ldots = T_q = \emptyset$. We move a job $\mathbf{a} \in L$ to a set T_k if that job fits together with the other jobs in T_k, i.e., if

$\|\mathbf{a}\|_\infty + \sum_{\mathbf{p}\in T_k} \|\mathbf{p}\|_\infty \leqslant y_{f_k,t}$. When this procedure terminates we claim that $|L| \leqslant q-1$.

Claim 4.27. $|L| \leqslant q-1$.

Proof. Suppose by contradiction that $|L| \geqslant q$, then for all $k \in [q]$ we have that $y_{f_k,t} - \sum_{\mathbf{p}\in T_k} \|\mathbf{p}\|_\infty < \min_{\mathbf{p}\in L} \|\mathbf{p}\|_\infty$. By summing over this inequality we have that

$$\sum_{k\in[q]} \left(y_{F_k,t} - \sum_{\mathbf{p}\in T_k} \|\mathbf{p}\|_\infty \right) < q \min_{\mathbf{p}\in L} \|\mathbf{p}\|_\infty .$$

By rearranging and noticing that $\sum_{k\in[q]} \left(y_{f_k,t}\right) \geqslant a(t)$ by constraints (C2) and that the sum over the infinity norm of all small jobs in S equals $a(\mathbf{t})$, we reach a contradiction since

$$\begin{aligned} a(t) = \sum_{k\in[q]} \left(y_{f_k,t}\right) &< q \min_{\mathbf{p}\in L} \|\mathbf{p}\|_\infty + \sum_{k\in[q]} \sum_{\mathbf{p}\in P_k} \|\mathbf{p}\|_\infty \\ &\leqslant \sum_{\mathbf{p}\in L} \|\mathbf{p}\|_\infty + \sum_{\mathbf{p}\in S\setminus L} \|\mathbf{p}\|_\infty = a(t) . \end{aligned}$$

■

After repeating this procedure for every type of small jobs, there are at most $2|\mathcal{F}|d$ small jobs in the left-over set and the rest is integrally assigned to profiles. Since there are at least $|\mathcal{F}|$ profiles for which a configurations is chosen (by the guessing phase in Step 1) we can assign $2d$ small vectors to each profile which leads to an increase of at most $2d \cdot \varepsilon'/d = 2\varepsilon'$. ■

(STEP 2) By step 1, small jobs are integrally assigned to profiles and thus we can restrict our attention to machines using a fixed profile. The small jobs are now assigned to the machine minimizing a potential function ϕ. Let $\mathbf{p_1}, \ldots, \mathbf{p_n}$ be small jobs assigned to the same profile and let $\lambda > 0$ be some parameter. Define

$$\phi^t := \sum_{k\in[d]} \sum_{i\in[m]} e^{\lambda(L^t_{i,k} - \mu^t_k)} ,$$

where $L^{t,i}_k$ is the load of small jobs $\mathbf{p_1}, \ldots, \mathbf{p_t}$ assigned to machine i on coordinate k, and μ^t_k is the expected load of the first t jobs in coordinate k, which is at most the free space reserved for small jobs by the solution for (MILP).

The idea is that the function ϕ is essentially a pessimistic estimator of the load exceeding the expectation, summed over the machines and coordinates. Regardless of how $\mathbf{p_1}, \ldots, \mathbf{p_{t-1}}$ are assigned, there is always an assignment for $\mathbf{p_t}$ that increases the potential function by a small multiplicative factor that is at most $\exp(\lambda^2(\|p_t\|_\infty)^2/m)$ (cf. Lemma 4.30). By assigning at each step the job to the machine that minimizes the potential, we have an integral assignment and

$$\phi^n \leqslant md\exp(2\lambda^2 dp_{max}),$$

where $p_{max} = \max_j \|\mathbf{p_j}\|_\infty$ is the maximum value over all small jobs and all coordinates.

Lemma 4.28 (Step 2). *For a profile* $\mathbf{f}$*, we can find an integral assignment of small jobs such that* $\phi^n \leqslant md\exp(2\lambda^2 dp_{max})$*, where* $p_{max} = \max_j \|\mathbf{p_j}\|_\infty$.

We need the following technical lemma before we prove step 2.

Lemma 4.29. *For all* $x \in [0,1]$ *and* $m \geqslant 1$,

$$(1/m)e^{(1-1/m)x} + (1-1/m)e^{-x/m} \leqslant e^{2x^2/m}. \tag{4.2}$$

Proof. Any real function $f(x)$ that is infinitely differentiable at a real number a can be expressed as a power series plus a remainder using Taylor series. For any real a, $f(x) = f(a) + f'(a)(x-a) + R_2$ where the Lagrangian remainder is $R_2 = (x-a)^2/2!f''(\alpha)$ for some $\alpha \in [a,x]$.

Applying this to the first term of the left-hand side of (4.2), we obtain

$$1/m + x(1/m)(1-1/m) + x^2/2\left(\left((1-1/m)^2/m\right)e^{(1-1/m)\alpha}\right),$$

and for the second term of the left-hand side we get

$$(1-1/m) - x(1-1/m)(1/m) + x^2/2\left(\left((1-1/m)/m^2\right)e^{-\beta/m}\right).$$

Notice that $x \in [0,1]$, and therefore $\alpha, \beta \in [0,1]$ as well. Hence, $e^{(1-1/m)\alpha} \leqslant e$ and $e^{-\beta/m} \leqslant 1$, and therefore summing both functions and maximizing over m, α and β yields

$$\begin{aligned}
&1 + x^2/2\left(\left((1-1/m)^2/m\right)e^{(1-1/m)\alpha}\right) \\
&\qquad + x^2/2\left(\left((1-1/m)/m^2\right)e^{-\beta/m}\right) \\
&\leqslant 1 + x^2/2\left((e+1)/m\right) \leqslant e^{2x^2/m}.
\end{aligned}$$

The last step is by the fact that $1 + y \leqslant e^y$ for all y. ∎

Recall that we have a fixed profile, and that every small job is assigned with probability $1/m$ to each individual machine in the randomized algorithm. The first part is to show that the expected potential slowly increases over the course of assignment of the small jobs. This implies that for every job there always exists a deterministic choice that increases the potential by at most this factor.

In particular, the next lemma shows that regardless of previous choices, the expected value of ϕ^t is at most a small factor times ϕ^{t-1}.

Lemma 4.30. *For any $\lambda > 0$, $t \geqslant 2$ and $\phi(t-1)$,*

$$\mathbb{E}\left[\phi^t\right] \leqslant \exp(\lambda^2(\|\mathbf{p}^t\|_\infty)^2/m)\phi^{t-1}\,.$$

Proof. For job $\mathbf{p}^t$ there are two choices per machine: with probability $1/m$ it is assigned to the machine and with probability $1-1/m$ it is not. Summing over the two possibilities per machine and coordinate and noticing that $f_k^t = f_k^{t-1} + p_k^t/m$ we have

$$\begin{aligned}
\mathbb{E}\left[\phi(t)\right] &= \sum_{k\in[d]}\sum_{i\in[m]}\left[\frac{1}{m}\exp\left(\lambda\left(L_k^{t-1,i} + p_k^t - (f_k^{t-1} + \frac{p_k^t}{m})\right)\right)\right. \\
&\qquad \left. +(1-\frac{1}{m})\exp\left(\lambda\left(L_k^{t-1,i} - (f_k^{t-1} + \frac{p_k^t}{m})\right)\right)\right] \\
&= \sum_{k\in[d]}\sum_{i\in[m]}\left[\frac{1}{m}\exp\left(\lambda(1-1/m)p_k^t\right)\exp\left(\lambda\left(L_k^{t-1,i} - f_k^{t-1}\right)\right)\right. \\
&\qquad \left. +(1-\frac{1}{m})\exp\left(-\lambda p_k^t/m\right)\exp\left(\lambda\left(L_k^{t-1,i} - f_k^{t-1}\right)\right)\right] \\
&\leqslant \sum_{k\in[d]}\sum_{i\in[m]}\exp\left(2\lambda^2(p_k^t)^2/m\right)\exp\left(\lambda\left(L_k^{t-1,i} - f_k^{t-1}\right)\right) \\
&\leqslant \sum_{k\in[d]}\sum_{i\in[m]}\exp\left(2\lambda^2(\|\mathbf{p}^t\|_\infty)^2/m\right)\exp\left(\lambda\left(L_k^{t-1,i} - f_k^{t-1}\right)\right)\,.
\end{aligned}$$

The third line is obtained by applying Lemma 4.29 with $x := \lambda p_k^t$ and the fourth line by replacing p_k^t by $\|\mathbf{p}^t\|_\infty$. Then by moving $\exp\left(2\lambda^2(\|\mathbf{p}^t\|_\infty)^2/m\right)$ outside the summations we obtain

$$\mathbb{E}\left[\phi(t)\right] \leqslant \exp\left(2\lambda^2(\|\mathbf{p}^t\|_\infty)^2/m\right)\phi(t-1)\,. \qquad \blacksquare$$

Proof of Lemma 4.28. Observe that $\phi^0 = md$. Then Lemma 4.30 yields

$$\phi^n \leqslant \exp\left((\lambda^2/m)\sum_{j\in[n]}\left(\|\mathbf{p}^j\|_\infty\right)^2\right) md\,.$$

Using $\sum_{j\in[n]}(\|\mathbf{p}^j\|_\infty)^2 \leqslant p_{max}\sum_{j\in[n]}(\|\mathbf{p}^j\|_\infty) \leqslant p_{max}md$, the result follows. ■

(STEP 3) Finally, we can prove that after steps 1 and 2, removing the small jobs from overloaded machines and reassigning them in a round-robin fashion over all machines, increases the load of each machine by at most $\mathcal{O}(\varepsilon)$.

Lemma 4.31 (Step 3)**.** *For a profile* $\mathbf{f}$ *and integral assignment of small jobs such that* $\phi^n \leqslant md\exp(2\lambda^2 dp_{max})$*, we can find an integral assignment of small jobs such that the total load of each machine on each coordinate is increased by at most* $\mathcal{O}(\varepsilon)$*.*

Proof. For a given assignment of jobs such that $\phi^n \leqslant md\exp(2\lambda^2 dp_{max})$, let $m(x)$ be the number of machines with load exceeding $1+\varepsilon+x$ in some coordinate and let L be the set of small jobs on machines with load at least $1+\varepsilon$ in any coordinate.

If $m(x)$ machines have such load, the potential is at least $m(x)\cdot e^{\lambda(\varepsilon+x)}$. By rewriting and setting $\lambda := 1/\varepsilon\log(d^3/\varepsilon^3)$ we obtain that

$$m(x) \leqslant \frac{md\exp(2\lambda^2 dp_{max})}{e^{\lambda(\varepsilon+x)}} = \frac{md\exp(\mathcal{O}(1))}{e^{\lambda(\varepsilon+x)}} \leqslant \frac{\mathcal{O}(md)}{(d^3/\varepsilon^3)^{1+x/\varepsilon}}.$$

Notice that we can safely pick $\delta := \varepsilon^4/d^2$, so $p_{max} \leqslant \varepsilon^4/d^2$.

The total load on machine with load at least $1+\varepsilon$ is $(1+\varepsilon)dm(0) + \int_{x=0}^{\infty} m(x)d \leqslant (1+\varepsilon)dm(0) + \sum_{i=0}^{\infty} m(\varepsilon i)d\varepsilon$. This summation is a geometric series that is shrinking very fast, i.e. $m(\varepsilon i) \geqslant (d^3/\varepsilon^3)m(\varepsilon(i+1))$. Therefore, we have that $d\varepsilon m(0) \geqslant \int_{x=0}^{\infty} m(x)d\varepsilon$, and therefore that $\|\sum_{\mathbf{p}\in L}\mathbf{p}\|_\infty$ is at most $\mathcal{O}(\varepsilon m/d)$.

By removing L from their assigned machines and reassigning at most one to each machine we increase the loads of machines by at most the maximum size of a small job and the lemma follows. ■

Implementation in linear time

While step 1 and 3 can be implemented directly in linear time, step 2 is more complicated. The trivial implementation of trying each machine for each job and evaluating the resulting value of ϕ takes $\mathcal{O}(md)$ time per job, which is super-linear.

To get around this, we do the following. Let δ be the maximum value of any coordinate of any small jobs. Greedily glue small jobs of the same type

together, as long as the resulting small job is still small. This results in jobs B that have their coordinates in $\{0\} \cup [\varepsilon\delta/2d, \delta]$ and at most $|\mathcal{T}_{\text{small}}|$ jobs S with coordinates in $[0, \delta/2]$. Next, round the coordinates of jobs from B down to a multiple of $\gamma = \varepsilon(\varepsilon\delta/2d)$, at the cost of an extra factor of $(1+\varepsilon)$ in the loads. This rounding procedure ensures that during the assignment of jobs from B, the machines have only few distinct loads, namely $T := (\lceil 1/\gamma \rceil + 1)^d$. This in turn establishes that for each job the best machine can be found in a greedy fashion. First order the machine loads according to their increase in the potential function, and then for each job go through the ordered loads and check whether there is a machine having that load. In total, at most T loads are checked and updating the machine load takes $O(d)$ time. A similar strategy for S would fail, but notice that $|S|$ is relatively small. If $|S| \leqslant m$, then assign one job to each machine, increasing the loads by at most $O(\delta)$. If $|S| > m$ then the number of machines is at most $|\mathcal{T}_{\text{small}}|$ and the trivial algorithm takes $O(|\mathcal{T}_{\text{small}}|^2) = O\left((\lceil \delta/\gamma \rceil + 1)^{2d}\right)$ time. This results in the following lemma.

Lemma 4.32. *Step 2 can be implemented in* $O(nd + (\lceil 2d/\varepsilon^2 \rceil + 1)^{2d})$ *time.*

From this and Theorem 4.25, we have our main theorem.

Theorem 4.4. *For any* $\varepsilon > 0$ *and* $d \geqslant 1$*, there is a deterministic* $(1+\varepsilon)$*-approximation algorithm for* d*-dimensional* VECTOR SCHEDULING *that runs in time* $O\left(2^{(1/\varepsilon)^{O(d \log\log d)}} + nd\right)$.

4.5 CONCLUDING REMARKS

This chapter (nearly) closes the gap between the hardness and the approximability of the VECTOR SCHEDULING problem. It exhibits lower bounds on the running time of approximation schemes under two complexity assumptions, and shows that even using resource augmentation, similar lower bounds hold. It complements these lower bounds by presenting an approximation scheme whose running time nearly matches the lower bounds. The running time of our algorithm is essentially the best possible, modulo the $O(\log\log d)$ factor in the exponent. A natural theoretical question to ask is whether this small gap can be closed completely, although the added value for real life application seems to be small.

Perhaps a more interesting direction for further research is to search for approximation algorithms with a constant approximation guarantee in sin-

gle exponential time. Is it possible to obtain e.g. a 2-approximation with running time $\mathcal{O}(2^d)$?

Another research direction that is still open, is the VECTOR SCHEDULING problem on non-identical machines. In the problem this chapter considers, the load on every coordinate of every machine needs to be at most 1. One could investigate the setting where these upper bounds on the loads can vary between machines. Obviously the lower bounds presented in this chapter immediately transfer to this setting, but algorithmic results on the positive side would be interesting to see.

5

OPTIMAL STOPPING AND POSTED PRICES

5.1 INTRODUCTION

5.1.1 *Optimal stopping theory*

Optimal stopping theory has been extensively studied in many different areas and concerns itself with maximizing the expected reward in problems in which a time needs to be chosen to take a particular action. The famous *prophet inequalities*, introduced in the sixties by Gilbert and Mosteller [45], are a key example of an optimal stopping problem. Here, a gambler faces a finite sequence of non-negative independent random variables that arrive in some order, with known distributions from which iteratively a prize is drawn. After seeing a prize, the gambler can either accept the prize and end the game, or reject the prize and await the next prize. The classical result of Krengel and Sucheston [73, 74], also attributed to Garling, states that the gambler can obtain at least half of the expected reward that a prophet can make who knows the realizations of the prizes beforehand. That is, $\sup\{\mathbb{E}[X_\tau] : \tau \text{ stopping rule}\} \geqslant \frac{1}{2}\mathbb{E}\{\sup_{1\leqslant i\leqslant n} X_i\}$. Moreover, Krengel and Sucheston also showed that this bound is best possible. Samuel-Cahn [91] showed that the bound of 2 can be obtained by a good threshold rule, which stops as soon as a prize is above a fixed threshold. In [92], Samuel-Cahn considers the case in which the random variables have a negative dependence. In this setting, she proves a slightly better bound and also shows that this bound is obtainable by the best threshold rule. Hill [57] studies the situation in which the order in which the random variables are presented can be chosen by the gambler. Kennedy [70] as well as Assaf, Goldstein and Samuel-Chan [7] considered the setting in which the gambler can select k different prizes. Whereas Kennedy looked at the situation in which the sum of the prizes is compared to that of an all-knowing prophet, Assaf, et al. studied the situation in which the maximum of the prizes is given as a reward. For more results on prophet inequalities, we refer to the survey of Hill and Kertz [59].

In this chapter we consider the setting in which the random variables arrive *uniformly at random*. We examine two stopping rules, namely a *non-*

adaptive one and an *adaptive* one. In the first, the gambler accepts a prize whenever it exceeds a threshold whose value is set beforehand and only depends on the distribution of the random variable. While in the latter, the value of the threshold is also allowed to depend on the history of the previously rejected prizes. The non-adaptivity of the first stopping rule is different from the classical prophet inequality setting. For this setting, we provide a tight bound of the expected reward compared to the expected maximum (cf. Theorem 5.3).

On the other hand, the adaptive stopping rule closely resembles the setting above, the secretary problem and the recently introduced prophet secretary problem [34]. In the adaptive setting we consider independent and identically distributed (i.i.d.) random variables, in which case the fraction of a half can be improved. The result for this setting can be seen as a follow up on a result by Hill and Kertz [58]. Their main result of is a recursive characterization of α_n, the best possible factor when faced with n i.i.d. random variables. More precisely, they prove that if $X_1, \dots, X_n$ are i.i.d. non-negative random variables and T_n denotes the set of stopping rules for $X_1, \dots, X_n$ then

$$\mathbb{E}(\max\{X_1, \dots, X_n\}) \leqslant \alpha_n \sup\{\mathbb{E}(X_t) : t \in T_n\}.$$

Furthermore, Hill and Kertz find instances in which it is not possible to beat the factor α_n. They also prove that $\alpha_n \leqslant e/(e-1)$, conjecture that the sequence is monotone, and leave open the existence and computation of its limit. The monotonicity together with the limit calculation would readily give a universal bound (valid for all n) on the performance of the best stopping rule. Shortly after, Samuel-Cahn [91] reports that Kertz proves existence of the limit α of the α_n sequence and conjectures that it equals 1.342 (obtained as the solution to $\int_0^1 (y - y\ln(y)) + \alpha - 1)^{-1}\, dy = 1$). Finally, Kertz [71, Lemma 6.2] proves the latter conjecture (for which Saint-Mont [90] derives a simpler proof). However, he is unable to prove that the sequence is monotone and therefore the best upper bound on the whole α_n sequence still stood at $e/(e-1) \approx 1.582$ [71, Lemma 3.4]. Very recently, and independently of our work, Abolhassani et al. [1] improved this upper bound to $1/0.738 \approx 1.355$. Our result (cf. Theorem 5.6) closes this gap and implies that for all n, $\alpha_n \leqslant \alpha \approx 1.3415$, and by the tight examples of Hill and Kertz [58] it turns out that this constant is best possible.

We extend these results in Section 5.4 to the setting of *posted price mechanisms*. In this setting, a seller sells one item by deciding on a, potentially

different, non-negotiable price for every interested customer. Both the non-adaptive and the adaptive results and algorithms described above yield corresponding corollaries for this setting.

5.1.2 *Problem description*

In the optimal stopping problem we consider, a gambler faces a sequence of n non-negative independent random variables X_i with known distributions F_i arriving in a random order, for $i \in N = \{1, \dots, n\}$. In every stage, a prize $\pi_i \sim F_i$ is drawn and the gambler needs to decide whether to accept and keep that prize, or to reject it and wait for the next realization. The goal is to maximize his expected reward. We consider a non-adaptive and an adaptive scenario.

Problem 5.1 (Non-adaptive optimal stopping problem)**.** The gambler sets thresholds $\tau_i \geqslant 0$ for all $i \in N$, with the goal of maximizing his expected reward defined as

$$\sum_{i \in N} \pi_i^\tau \mathbb{P}_{\sigma,F} \left[i = \operatorname*{argmin}_{j \in N} \{\sigma(j) \,|\, \pi_j \geqslant \tau_j\} \right],$$

where the probability is taken over the arrival permutation σ and the distributions of the random variables F. Furthermore, π_i^τ denotes the random variable $(\pi_i \,|\, \pi_i \geqslant \tau_i)$.

Problem 5.2 (Adaptive optimal stopping problem)**.** The gambler sets thresholds upon arrival of every random variable. So, the gambler sets functions $\tau_i : 2^N \to \mathbb{R}$ for each random variable X_i, such that, if S is the set of random variables that did not exceed their threshold before, $\tau_i(S)$ is the threshold for random variable X_i if this is the next random variable to arrive. For an arrival permutation σ, we denote $\pi_i(\sigma) = \pi_i(\{\sigma^{-1}(1), \dots, \sigma^{-1}(\sigma(i) - 1)\})$ and $\pi_i^\tau(\sigma) = (\pi_i(\sigma) \,|\, \pi_i(\sigma) \geqslant \tau_i)$, and therefore we can write the gambler's expected revenue as

$$\mathbb{E}_\sigma \left[\sum_{i \in N} \pi_i^\tau(\sigma) \mathbb{P}_F \left[i = \operatorname*{argmin}_{j \in N} \{\sigma(j) \,|\, \pi_j \geqslant \tau_j(\sigma) \right] \right],$$

where the expectation is taken over the arrival permutation σ, and the probability is taken over the distributions of the random variables F.

5.1.3 *Our results*

We present two stopping rules: A non-adaptive stopping rule that guarantees an expected reward within a factor $1 - 1/e$ of the expected value of the maximum, and an adaptive stopping rule for i.i.d. prize distributions that has a guaranteed expected reward of 0.745 of the expected value of the maximum.

Theorem 5.3. *Given n independent non-negative random variables $X_1, \ldots, X_n$ with $X_i \sim F_i$. There exist values $\tau_1, \ldots, \tau_n$ such that*

$$\mathbb{E}\left[\frac{\sum_{i=1}^{n} X_i Y_i}{\sum_{i=1}^{n} Y_i}\right] \geqslant \left(1 - \frac{1}{e}\right) \mathbb{E}\left[\max_{i=1,\ldots,n} X_i\right],$$

where Y_i is a Bernoulli random variable that has value 1 if $X_i > \tau_i$.

Note that the quantity on the left exactly corresponds to the expected value of the first X_i above τ_i, when the X_i's are ordered uniformly at random.

In the case of continuous distributions, the algorithm that achieves this result becomes remarkably simple, see Algorithm 5.1. The algorithm, while randomized, can be derandomized using standard techniques.

Algorithm 5.1: Algorithm for the non-adaptive stopping rule.

1 Compute $q_i = \mathbb{P}\left(X_i = \max_{j \in N} X_j\right)$;

2 Set threshold $\tau_i = \begin{cases} F_i^{-1}(1 - q_i) & \text{w.p. } \frac{2}{2+(e-2)q_i} \\ \infty & \text{otherwise.} \end{cases}$;

3 Accept the first random variable having $X_i > \tau_i$;

Algorithm 5.1 may seem counterintuitive since, in step 2, the higher the probability is that a random variable is the maximum, the higher the probability is that the algorithm rejects it a priori (by setting a threshold of ∞), though the probability of rejecting any random variable is at most $1 - \frac{2}{e}$. The following example gives some intuition on why random variables need to be rejected.

Example 5.4. Consider just two random variables: X_1 who has deterministic value equal to 1, and X_2 who has a value of 100 with probability 1/10 and 0 with probability 9/10. In this situation, the optimal stopping rule chooses X_2 with probability 9/10 and the total expected reward is $10 + 9/10$. Now, if

a non-adaptive algorithm sets finite thresholds for both random variables, the expected reward is $(1/10)(50.5) + (9/10)(1) = 5.95$, which is not within the claimed ratio of the optimal mechanism.

Another somewhat surprising element of Algorithm 5.1 is that the probability of not accepting any random variable can be computed as $\prod_{i\in N}(1 - 2q_i/(2 + (e-2)q_i)) \geqslant 2/e$. Again, the previous example provides intuition to the fact that, if we shoot for an algorithm that accepts too frequently, we risk settling for too low a prize. This intuition does not hold in the adaptive case.

The cornerstone of our analysis is a basic result about Bernoulli random variables which may be of independent interest. The result states that if we are given a set of non-homogeneous independent Bernoulli random variables with associated prizes, then there is a subset of variables so that the expected average prize of the successes is at least a factor $1 - 1/e$ of the expectation of the maximum prize over all random variables.

Lemma 5.5 (Bernoulli Selection Lemma)**.** *Given a set* $N = \{1, \ldots, n\}$ *of independent Bernoulli random variables* $X_1, \ldots, X_n$*, where* $X_i = 1$ *with probability* q_i *and* 0 *otherwise, and associated prizes* $b_1, \ldots, b_n$*. The following inequalities hold:*

$$\frac{e}{e-1} \max_{S\subseteq N} \mathbb{E}\left[\frac{\sum_{i\in S} b_i X_i}{\sum_{i\in S} X_i}\right] \geqslant \max_{z_i \leqslant q_i} \left\{ \sum_{i\in N} b_i z_i \;\middle|\; \sum z_i \leqslant 1 \right\} \geqslant \mathbb{E}[\max_{i\in N}\{b_i X_i\}]\,.$$

Here, when evaluating the leftmost term, we define $0/0 = 0$*.*

To prove the lemma, we consider a continuous relaxation of the maximization problem, and then guess a solution in which each random variable is included in S with some instance-dependent probability. Then, we look for the worst possible instance by applying the first order optimality conditions of a non-linear problem. These conditions reveal some structural insight on what a worst case instance looks like. Using this, we obtain the desired bound. Theorem 5.3 follows from Lemma 5.5 with fairly little extra work.

To complement our results, we provide instances that show that the bounds in Lemma 5.5 and Theorem 5.3 are tight. In particular, we show that even with *independent identically distributed* (*i.i.d.*) random variables the bound of Theorem 5.3 cannot be beaten. Therefore, to go beyond $1 - 1/e$, even with i.i.d. distributions, a different setting needs to be considered. We examine the adaptive setting, for which we show the following bound of $1/\beta^* \approx 0.745$. As stated before, the tight examples of Hill and Kertz [58] show that this constant is best possible.

Theorem 5.6. *Given non-negative i.i.d. random variables* $X_1, \dots, X_n$*, there exist thresholds* $\tau_1, \dots, \tau_n$*, such that*

$$\mathbb{E}(\max\{X_1, \dots, X_n\}) \leqslant \beta^* \mathbb{E}(X_t),$$

where $t := \min\{i \in \{1, \dots, n\} : X_i \geqslant \tau_i\}$ *and* $\beta^* \approx 1.3415$ *is the unique value such that*

$$\int_0^1 \frac{1}{y(1 - \ln(y)) + (\beta - 1)} \, dy = 1 . \tag{5.1}$$

To achieve this result we use a quite natural idea: as less random variables are left, the thresholds should decrease. Besides this the key ingredient of our algorithm is to use random thresholds drawn from a well chosen distribution that mimics an expression we obtain for the expected maximum value. See Algorithm 5.2 for our algorithm in the case of continuous distributions. Like Algorithm 5.1, also Algorithm 5.2 can be derandomized using standard techniques.

Algorithm 5.2: Algorithm for the adaptive stopping rule.

1 Partition the interval $[0, 1]$ into intervals $A_i = [a_{i-1}, a_i]$, s.t. $a_0 = 0$, $a_n = 1$;
2 Sample q_i from A_i with an appropriately chosen distribution ;
3 When the i—th random variable comes, set threshold $\tau_i = F^{-1}(1 - q_i)$;

5.1.4 *Posted price mechanisms*

In Section 5.4 we proceed by highlighting the connection between optimal stopping theory and posted price mechanisms, which constitute an attractive and widely applicable way of selling items to strategic consumers. In this context, consumers are faced with take-it-or-leave-it offers, and therefore strategic behaviour simply vanishes. This type of mechanism has been vastly studied, particularly in the marketing community [18]. In recent years, there has been a significant effort to understand the expected revenue of the outcome generated by different posted price mechanisms when compared to that of the optimal auction [3, 13, 21, 22, 51, 101]. In addition, several companies have started to apply *personalized pricing* to sell their products. Under this policy, companies set different prices for different consumers based on purchase history or other factors that may affect

their willingness to pay. For example, the online data provider Lexis-Nexis sells to virtually every user at a different price [94]. In 2012, Orbitz online travel agency found that people who use Mac computers spent as much as 30% more on hotels, so it started to show them different, and sometimes costlier, travel options than those shown to Windows visitors [79]. Similarly, retailers and supermarket chains such as Safeway are using data culled from billions of purchases to offer deals tailored to specific shoppers [72]. Choudhary et al. [24] further investigated this issue, providing more examples and developing a theoretical framework to analyze equilibria between firms that apply personalized pricing and those who do not.

In its simplest form, the problem we consider in this setting is described as follows. A monopolist sells a single item to a set of n known potential buyers. The seller places no value on the item, while each buyer i has a private value v_i for the item, independent of the values of the other buyers. The seller does not know exactly how much everyone values the item, but he has some beliefs, e.g. based on history. Therefore he assumes that every value v_i is drawn from a known distribution F_i, which might be different for each buyer. The main question is to design a mechanism maximizing the revenue of the seller.

This question has been answered in 1981 by Myerson [83]. He uses the virtual valuation function of each customer i, which is defined as $c_i(v) = v - \frac{1-F_i(v)}{f_i(v)}$, where f_i is the density of F_i. If all virtual valuation functions are monotone, the auction and its analysis become much simpler and this is referred to as the *regular* case. In the general case, these virtual valuations need to be ironed in the following sense. Let $H_i(q) = \int_0^q c_i(F_i^{-1}(\theta))\,d\theta$ be the negative revenue curve as a function of the acceptance probability q. Then define $G(q) = \text{conv}(H(q))$ as the convexification of $H(q)$, that is, $G(q) = \min\{\gamma H(q_1) + (1-\gamma)H(q_2) : \gamma q_1 + (1-\gamma)q_2 = q,\ \gamma, q_1, q_2 \in [0,1]\}$. The ironed virtual valuation function is then defined as $\bar{c}(v) = G'(F_i(q))$. Myerson's auction now sells the item to the buyer with the highest virtual valuation, provided this is positive.

Though Myerson's auction is optimal, in the general case it is involved and therefore not as widely used in practice as one might expect from the optimal auction format. Researchers therefore started studying formats that might be suboptimal but are easy and simple to understand, like *posted price mechanisms*. Here, the buyers arrive sequentially and the seller offers each arriving buyer i a single take-it-or-leave-it price p_i. If she accepts the price, the seller makes a revenue of p_i. If she refuses the offered price, she leaves the system and never comes back, and the seller can set a potentially

different price for the next arriving customer. The natural question to ask is what prices the seller needs to set to maximize his expected revenue. A common example of this practice is that of *direct mail campaigns*, in which the seller contacts its potential buyers directly and offers each one a certain price for the item. The item is then sold to the first consumer who accepts the offer [18, 27].

First Hajiaghayi, Kleinberg, and Sandholm [51] and then Chawla et al. [22] noted a close connection between online mechanisms in general and posted price ones in particular and prophet inequalities. Specifically, Chawla et al. [22] implicitly show that the problem of designing posted price mechanisms can be reduced to that of finding stopping rules of a related prophet inequality. This connection opened the way for new approaches and results and constitutes the starting point of this paper. We refer to the recent survey of Lucier [77] for further details. Theorem 5.21 highlights this connection by showing a new reduction.

In the non-adaptive setting, all offers have to be made simultaneously, and customers respond in random order, akin to direct mail campaigns. In the adaptive posted price mechanism, the seller may base the price on the customer he is offering to, as well as the customers who already rejected earlier offers. The reduction allows us to prove bounds for both settings, yielding Corollary 5.22 and Corollary 5.24 as the counterparts of Theorem 5.3 and Theorem 5.6 above. Like above, we also present algorithms achieving these bounds.

5.2 THE BERNOULLI SELECTION LEMMA

In this section we prove Lemma 5.5. Actually, we prove a slightly stronger version which will become clear at the end of the proof. Moreover, we provide an instance showing that the bound is tight and discuss some generalizations.

5.2.1 *The proof*

The second inequality of Lemma 5.5 is trivial, as the expectation of the maximum is a sum over all values b_i weighed by the probability with which that value is the maximum. Since these probabilities sum to at most one, the inequality follows.

The proof for the first inequality has two main ingredients. First, we reformulate the left hand side in an appropriate way, and lower bound it by another function using KKT-conditions. Then, we show that this function is bounded from below by $1 - 1/e$.

A simplified proof

As a warm up, we first show how to get a weaker result, that only gives us a factor of $\sqrt{e}$ instead of $\frac{e}{e-1}$, with more straightforward arguments.

Proof. We start the proof by rewriting the optimization problem:

$$\max_{S \subseteq N} \left\{ \mathbb{E}\left[\frac{\sum_{i \in S} b_i X_i}{\sum_{i \in S} X_i} \right] \right\} . \tag{P}$$

Instead of choosing a subset of N, we set for each $i \in N$ a value $\chi_i \in [0,1]$, which represents the probability with which we actually choose i. Now, let $\pi_i = \chi_i q_i$ denote the probability of i being picked and having $X_i = 1$. So we can consider the following maximization problem, with decision variables π, as a relaxation of (P):

$$\max_{0 \leqslant \pi_i \leqslant q_i} \sum_{S \subseteq N} \left(\frac{b(S)}{|S|} \left(\prod_{i \in S} \pi_i \right) \left(\prod_{i \notin S} (1 - \pi_i) \right) \right) ,$$

where $b(S) = \sum_{i \in S} b_i$. Note that the previous objective is linear in each variable so that there is an extreme optimal solution [27]. Thus, the previous problem is in fact equivalent to (P). Now, by changing the order of the summations, we obtain

$$\max_{0 \leqslant \pi_i \leqslant q_i} \sum_{i \in N} b_i \pi_i \sum_{S \subseteq N \setminus \{i\}} \frac{1}{1 + |S|} \prod_{j \in S} \pi_j \prod_{j \in N \setminus (S \cup \{i\})} (1 - \pi_j) . \tag{5.2}$$

With (P) in this equivalent form, we now proceed to guess a feasible solution. To this end, consider an optimal solution z^* to

$$\max \left\{ \sum_{i \in N} b_i z_i \,\middle|\, \sum_{i \in N} z_i \leqslant 1, z_i \leqslant q_i \text{ for all } i \in N \right\} ,$$

and set $\pi_i = 2z_i^*/(2 + z_i^*)$. Note that $\pi_i \leqslant q_i$, so that, substituting this in the objective of (5.2), we get

$$\sum_{i \in N} b_i z_i^* \prod_{j \in N} \frac{1}{1 + \frac{z_j^*}{2}} \sum_{S \subseteq N \setminus \{i\}} \frac{2^{|S|}}{1 + |S|} \prod_{j \in S} \frac{z_j^*}{2} \prod_{j \in N \setminus (S \cup \{i\})} \left(1 - \frac{z_j^*}{2} \right) .$$

It is easy to see that

$$\sum_{S\subseteq N\setminus\{i\}} \frac{2^{|S|}}{1+|S|}\prod_{j\in S}\frac{z_j^*}{2}\prod_{j\in N\setminus(S\cup\{i\})}\left(1-\frac{z_j^*}{2}\right)\geqslant 1\,,$$

since the left hand side corresponds to $\mathbb{E}[f(S)]$ over all $S\subseteq N\setminus\{i\}$ under probabilities $z_j^*/2$ for every element i and $f(S)=2^{|S|}/(|S|+1)\geqslant 1$. While for any values z_i such that $\sum_i z_i\leqslant 1$, we have

$$\prod_{j=1}^{n}\frac{1}{1+\frac{z_j}{2}}\geqslant e^{-\sum_{j=1}^{n}\frac{z_j}{2}}\geqslant\frac{1}{\sqrt{e}}\,,$$

where the first inequality follows from $1+x\leqslant e^x$, concluding the proof. ■

Obtaining a bound of $1-1/e$

To obtain the improved factor $1-1/e$ we do the same as in the simplified proof, but make a subtle modification in the choice of π_i. We choose $\pi_i=\frac{2z_i^*}{2+(e-2)z_i^*}$, such that $1-\pi_i=\frac{2-(4-e)z_i^*}{2+(e-2)z_i^*}$. Note that this is a feasible choice of π_i for all $i\in N$, since for this choice $\pi_i\leqslant z_i^*\leqslant q_i$. Because of the choice of π_i, we actually prove the slightly stronger bound where we maximize over $z_i\leqslant\frac{2q_i}{2-(e-2)q_i}$. The choice of π_i suggests that the random variables are not picked deterministically, but with probability less than 1, since $\pi_i<z_i^*$ if $z_i^*>0$. However, as noted in the beginning of the proof, because of linearity of the objective in each variable, there is always an extreme optimal solution where the random variables are picked deterministically.

We plug this back into (5.2), and obtain that (P) is lower bounded by

$$\sum_{i\in N}2b_iz_i^*\left(\prod_{j\in N}\frac{1}{2+(e-2)z_j^*}\right)\cdot \sum_{S\subseteq N\setminus\{i\}}\frac{2^{|S|}}{1+|S|}\prod_{j\in S}z_j^*\prod_{j\in N\setminus(S\cup\{i\})}(2-(4-e)z_j^*)\,. \tag{5.3}$$

We proceed to lower bound this quantity, where we use the following technical result.

Proposition 5.7. *Consider the problem* $\min_{x\in\mathbb{R}_+^M}\{f_M(x):\sum_{i\in M}x_i\leqslant a\}$, *where* $a\leqslant 1$ *and*

$$g_M(x)=\left(\prod_{j\in M}\frac{1}{2+(e-2)x_j}\right)\sum_{S\subseteq M}\frac{2^{|S|}}{1+|S|}\prod_{j\in S}x_j\prod_{j\in M\setminus S}(2-(4-e)x_j)\,.$$

In an optimal solution, all non-zero variables are equal and $\sum_{i\in M} x_i = a$.

Proof. Consider an optimal solution x^*, and assume its support is $M' \subseteq M$. Let y^* be the restriction of x^* to M'. Then, $g_M(x^*) = g_{M'}(y^*)$ and y^* minimizes $g_{M'}$. Consider the function $g(y_1, y_2)$ as the function $g_{M'}$ restricted to the first two variables, while the others are fixed to y_i^*. Clearly, y_1^*, y_2^* minimize $g(y_1, y_2)$ subject to the constraints that $y_1, y_2 > 0$, and $y_1 + y_2 \leqslant a - \sum_{i\in M'\setminus\{1,2\}} y_i^*$. Now $g(y_1, y_2)$ can be written as

$$g(y_1, y_2) = \frac{A + By_1 + By_2 + Cy_1y_2}{(2 + (e-2)y_1)(2 + (e-2)y_2)},$$

where

$$A = 4\left(\prod_{j\in N'} \frac{1}{2 + (e-2)y_j^*}\right) \sum_{S\subseteq N'} \frac{2^{|S|}}{1 + |S|} \prod_{j\in S} y_j^* \prod_{j\in N'\setminus S} (2 - (4-e)y_j^*),$$

$$B = \frac{e-4}{2}A + 2\left(\prod_{j\in N'} \frac{1}{2 + (e-2)y_j^*}\right) \sum_{S\subseteq N'} \frac{2^{|S|+1}}{2 + |S|} \prod_{j\in S} y_j^* \prod_{j\in N'\setminus S} (2 - (4-e)y_j^*),$$

$$C = \frac{e-4}{2}B + \left(\prod_{j\in N'} \frac{1}{2 + (e-2)y_j^*}\right) \sum_{S\subseteq N'} \frac{2^{|S|+2}}{3 + |S|} \prod_{j\in S} y_j^* \prod_{j\in N'\setminus S} (2 - (4-e)y_j^*),$$

with $N' = M' \setminus \{1, 2\}$. Since the constraint $y_1 + y_2 \leqslant a - \sum_{i\in N'} y_i^*$ is the only active constraint and it is symmetric with respect to y_1 and y_2, the KKT conditions dictate that a minimum of $g(y_1, y_2)$ satisfies

$$\frac{\partial g(z_1^*, z_2^*)}{\partial z_1^*} = \frac{\partial g(z_1^*, z_2^*)}{\partial z_2^*}. \tag{5.4}$$

Taking the derivatives

$$\frac{\partial g(y_1, y_2)}{\partial y_1} = \frac{2B + 2y_2C - (e-2)A - (e-2)y_2B}{(2 + (e-2)y_1)^2(2 + (e-2)y_2)},$$
$$\frac{\partial g(y_1, y_2)}{\partial y_2} = \frac{2B + 2y_1C - (e-2)A - (e-2)y_1B}{(2 + (e-2)y_1)(2 + (e-2)y_2)^2},$$

we see that (5.4) holds if and only if

$$\left((4C-(e-2)^2A)+(2(e-2)C-(e-2)^2B)(y_2+y_1)\right)(y_2-y_1)=0\,.$$

So either $y_1=y_2$, or at least one is strictly positive and

$$y_1+y_2=\frac{(e-2)^2A-4C}{2(e-2)C-(e-2)^2B}\,.$$

We evaluate the right-hand side of the latter, using the formulae for A, B, and C. Note first that $A\geqslant 0$, and observe that

$$\begin{aligned}
B&=\frac{e-4}{2}A+\\
&\quad 2\left(\prod_{j\in N'}\frac{1}{2+(e-2)y_j^*}\right)\sum_{S\subseteq N'}\frac{2^{|S|+1}}{2+|S|}\prod_{j\in S}y_j^*\prod_{j\in N'\setminus S}(2-(4-e)y_j^*)\\
&\leqslant\frac{e-4}{2}A+\\
&\quad 4\left(\prod_{j\in N'}\frac{1}{2+(e-2)y_j^*}\right)\sum_{S\subseteq N'}\frac{2^{|S|}}{2+|S|}\prod_{j\in S}y_j^*\prod_{j\in N'\setminus S}(2-(4-e)y_j^*)\\
&\leqslant\frac{e-4}{2}A+A=\frac{e-2}{2}A\,.
\end{aligned}$$

Now we use that $\dfrac{2^{|S|+i+2}}{i+2+|S|}\geqslant\dfrac{2^{|S|+i}}{i+|S|}$ for $i\in\mathbb{R}_+$ and all $|S|$. We can write

$$\begin{aligned}
C&=\frac{e-4}{2}B+\\
&\qquad\left(\prod_{j\in N'}\frac{1}{2+(e-2)y_j^*}\right)\sum_{S\subseteq N'}\frac{2^{|S|+2}}{3+|S|}\prod_{j\in S}y_j^*\prod_{j\in N'\setminus S}(2-(4-e)y_j^*)\\
&\geqslant\frac{e-4}{2}B+\\
&\qquad\left(\prod_{j\in N'}\frac{1}{2+(e-2)y_j^*}\right)\sum_{S\subseteq N'}\frac{2^{|S|+1}}{2+|S|}\prod_{j\in S}y_j^*\prod_{j\in N'\setminus S}(2-(4-e)y_j^*)\\
&\geqslant\frac{e-4}{2}B+\frac{1}{2}B-\frac{(e-4)}{4}A\geqslant\tfrac{e-3}{2}B-\tfrac{(e-4)}{4}A\\
&\geqslant\tfrac{(e-3)(e-2)}{4}A-\tfrac{(e-4)}{4}A=\tfrac{e^2-6e+10}{4}A\,.
\end{aligned}$$

Now we bound

$$y_1^* + y_2^* = \frac{(e-2)^2A - 4C}{2(e-2)C - (e-2)^2B} \leqslant \frac{(e-2)^2A - (e^2-6e+10)A}{\frac{(e-2)(e^2-6e+10)}{2}A - \frac{(e-2)^3}{2}A}$$
$$= \frac{(2e-6)A}{\frac{(e-2)((e^2-6e+10)-(e-2)^2)}{2}A} = \frac{2(2e-6)}{(e-2)(6-2e)},$$

which is negative. This contradicts the constraint $y_1, y_2 > 0$. As the choice of the index $\{1,2\}$ is arbitrary, we conclude that all coordinates of y^* have to be equal.

To finish the proof, we still need to show that in an optimal solution, the constraint $\sum_{i\in M} x_i^* = \sum_{i\in M'} y_i^* \leqslant a$ is tight. As all coordinates of y^* are equal from the above, we know that $y_i^* = \bar{y}$, for some $\bar{y}$. Let k denote the number of non-zero variables in x^*, i.e., $k = |M'|$. Abusing notation, let

$$g_{M'}(\bar{y}) = \left(\prod_{j\in M'} \frac{1}{2+(e-2)\bar{y}}\right) \sum_{S\subseteq M'} \frac{2^{|S|}}{1+|S|} \prod_{j\in S} \bar{y} \prod_{j\in M'\setminus S} (2-(4-e)\bar{y})$$
$$= (2+(e-2)\bar{y})^{-k} \sum_{\ell=0}^{k} \binom{k}{\ell} \frac{2^\ell}{1+\ell} \bar{y}^\ell (2-(4-e)\bar{y})^{k-\ell}$$
$$= \frac{(2+(e-2)\bar{y})^{k+1} - (2-(4-e)\bar{y})^{k+1}}{(2+(e-2)\bar{y})^k \cdot 2\bar{y}(k+1)}.$$

We claim that $g_{M'}(\bar{y}+\varepsilon) \leqslant g_{M'}(\bar{y})$, for small $\varepsilon > 0$ and $\bar{y} < \frac{a}{k}$. In order to prove this, we take the derivative of $g_{M'}(\bar{y})$ with respect to $\bar{y}$, and show that this is non-positive for $\bar{y} \geqslant 0$, from which the claim follows. Note that

$$\frac{\partial g_{M'}(\bar{y})}{\partial \bar{y}} = \frac{(2+(e-2)\bar{y})^{-(k+1)}}{(k+1)\bar{y}^2} \cdot$$
$$\left((2-(4-e)\bar{y})^k(2+(e-2)\bar{y}+2k\bar{y}) - (2+(e-2)\bar{y})^{k+1}\right).$$

As $\bar{y} \geqslant 0$, it is easy to see that the sign of the derivative is equal to the sign of the function $(2-(4-e)\bar{y})^k(2+(e-2)\bar{y}+2k\bar{y}) - (2+(e-2)\bar{y})^{k+1}$. Therefore, to show that the derivative is non-positive for $\bar{y} \geqslant 0$, it is sufficient to show that

$$(2+(e-2)\bar{y})^{k+1} \geqslant (2-(4-e)\bar{y})^k(2+(e-2)\bar{y}+2k\bar{y}). \tag{5.5}$$

We will prove this inequality is true by induction on the value of k. For $k = 1$, we have

$$(2+(e-2)\bar{y})^2 = (2-(4-e)\bar{y})(2+(e-2)\bar{y}+2\bar{y}) + 4\bar{y}^2$$
$$\geqslant (2-(4-e)\bar{y})(2+(e-2)\bar{y}+2\bar{y}).$$

Assume that (5.5) is true for given k. Then,

$$\begin{aligned}
&(2+(e-2)\bar{y})^{k+2} \\
&\quad \geqslant (2-(4-e)\bar{y})^{k}(2+(e-2)\bar{y}+2k\bar{y})(2-(4-e)\bar{y}+2\bar{y}) \\
&\quad = (2-(4-e)\bar{y})^{k+1}(2+(e-2)\bar{y}+2k\bar{y}) \\
&\qquad + (2-(4-e)\bar{y})^{k}(2-(4-e)\bar{y}+2(k+1)\bar{y})2\bar{y} \\
&\quad \geqslant (2-(4-e)\bar{y})^{k+1}(2+(e-2)\bar{y}+2k\bar{y}) + (2-(4-e)\bar{y})^{k+1}2\bar{y} \\
&\quad = (2-(4-e)\bar{y})^{k+1}(2+(e-2)\bar{y}+2(k+1)\bar{y}),
\end{aligned}$$

where the first inequality is due to the induction hypothesis. Hence, (5.5) is true. For each $k \geqslant 1$ and $\bar{y} \geqslant 0$, the derivative is non-positive, and $g_{M'}(\bar{y})$ is minimized for $\bar{y}$ as large as possible, that is, $\sum_{i\in M'} \bar{y} = a$. ■

Using Proposition 5.7, we lower bound (5.3) as follows. Consider the term

$$\left(\prod_{j\in N\setminus\{i\}} \frac{1}{2+(e-2)z_j^*}\right) \sum_{S\subseteq N\setminus\{i\}} \frac{2^{|S|}}{1+|S|} \prod_{j\in S} z_j^* \prod_{j\in N\setminus(S\cup\{i\})} (2-(4-e)z_j^*).$$

Note that this is equal to $g_{N\setminus\{i\}}(z^*_{-i})$, where x_{-i} denotes the vector x with coordinate i eliminated. So, Proposition 5.7 can be applied with $a = 1-z_i^*$. Thus,

$$g_{N\setminus\{i\}}(z^*_{-i}) \geqslant g_{N\setminus\{i\}}(x^*),$$

with x^* the optimal solution to $\min_{x\in\mathbb{R}_+^{N\setminus\{i\}}}\{g_{N\setminus\{i\}}(x) : \sum_{j\in N\setminus\{i\}} x_j \leqslant a\}$.

Proposition 5.7 states that $x_j^* = (1-z_i^*)/k$, where $k \leqslant n-1$ is the number of non-zero variables in x^*. Conditioning on the cardinality of the set S, and using the Binomial Theorem, a straightforward but tedious calculation shows that

$$g_{N\setminus\{i\}}(x^*) = \frac{2k+(e-2)(1-z_i^*)}{2(k+1)(1-z_i^*)} \left(1-\left(1-\frac{2(1-z_i^*)}{2k+(e-2)(1-z_i^*)}\right)^{k+1}\right).$$

As this quantity only depends on k and z_i^*, we may define

$$\varphi_k(z_i^*) = \frac{2}{(2+(e-2)z_i^*)} g_{N\setminus\{i\}}(x^*),$$

to conclude that expression (5.3) (and in turn (P)) is lower bounded by

$$\sum_{i\in N} b_i z_i^* \varphi_{k(i)}(z_i^*).$$

where the index $k(i)$, denoting the number of non-zero variables in x^*, is always at least 1, yet may vary, depending on i.

Bounding the function by $1 - 1/e$

The remainder of the proof establishes that $\varphi_{k(i)}(z_i^*) \geqslant 1 - \frac{1}{e}$. Our quantity of interest is

$$\varphi_n(y) = \frac{2}{2+(e-2)y} \frac{2n+(e-2)(1-y)}{2(n+1)(1-y)} \cdot \left(1 - \left(1 - \frac{2(1-y)}{2n+(e-2)(1-y)}\right)^{n+1}\right).$$

We prove that $\varphi_n(y) \geqslant 1 - \frac{1}{e}$ for all $y \in [0,1]$ and $n \geqslant 2$. We start with a lemma that rephrases this claim. Define

$$f_n(x) := \frac{1}{n+1} - \frac{(1-x)^{n+1}}{n+1} - \frac{e-1}{2}x + \frac{(e-1)(e-2)n}{e(2-(e-2)x)}x^2.$$

Lemma 5.8. *$\varphi_n(y) \geqslant 1 - \frac{1}{e}$ for all $y \in [0,1]$ and all $n \geqslant 2$ if and only if $f_n(x) \geqslant 0$ for all $n \geqslant 1$ and $x \in [0, \bar{x}]$, where $\bar{x} = 1/(n-1+e/2)$.*

Proof. Consider the variable change

$$x = \frac{2(1-y)}{2(n-1)+(e-2)(1-y)},$$

so that

$$y = \frac{2-(2(n-1)+e-2)x}{2-(e-2)x}.$$

As y ranges from 0 to 1, x ranges from 0 to $\frac{1}{n-2+e/2}$. Note that

$$\frac{2}{2+(e-2)y} = \frac{2(2-(e-2)x)}{e(2-(e-2)x)-2(e-2)(n-1)x}.$$

Substituting this, we see that $\varphi_n(y) \geqslant \frac{e-1}{e}$ holds for all $y \in [0,1]$ and $n \geqslant 2$ if and only if

$$\frac{1}{n}(1-(1-x)^n) \geqslant \frac{e-1}{e}x \frac{e(2-(e-2)x)-2(e-2)(n-1)x}{2(2-(e-2)x)},$$

for all $x \in [0, \frac{1}{n-2+e/2}]$ and $n \geqslant 2$. Moving the index of n by 1 and rewriting the inequality above gives the result. ∎

Using Lemma 5.8, the new goal is to prove $f_n(x) \geqslant 0$ for all $n \geqslant 1$ and $x \in [0, \bar{x}]$. We use the following result.

Proposition 5.9. *Let* $c \in [0, \frac{1}{2}]$. *Then* $h_1(x) = \left(1 - \frac{1}{x+c}\right)^x$ *is a non-decreasing function of* x *in* $(1, \infty)$.

Proof. Define $h_2(x) = \ln(h_1(x)) = x \ln\left(1 - \frac{1}{x+c}\right)$. We prove $h_1(x)$ is non-decreasing by proving that $h_2'(x) \geqslant 0$. Note that

$$h_2'(x) = \ln\left(\frac{x+c-1}{x+c}\right) + \frac{x}{(x+c-1)(x+c)},$$

which is non-negative if and only if

$$\frac{x}{(x+c-1)(x+c)} \geqslant \ln\left(1 + \frac{1}{x+c-1}\right).$$

We substitute $\frac{1}{x+c-1} = z$. Then, the right-hand side is equal to $\ln(1+z)$, while the left-hand side is equal to

$$\frac{x}{(x+c-1)(x+c)} = \frac{\frac{1}{z} - c + 1}{\frac{1}{z} + 1} z = \frac{1 + z - cz}{1+z} z = \frac{z}{1+z}(1 + (1-c)z).$$

When we expand $\ln(1+z) = z - \frac{z^2}{2} + \frac{z^3}{3} - \frac{z^4}{4} \pm \ldots$, we see that it is sufficient to prove

$$\frac{z}{1+z}(1 + (1-c)z) \geqslant z - \frac{z^2}{2} + \frac{z^3}{3} - \frac{z^4}{4} \pm \ldots.$$

We multiply both sides by $\frac{1+z}{z}$ to retrieve

$$1 + (1-c)z \geqslant 1 + \frac{z}{2} - \frac{z^2}{6} + \frac{z^3}{12} - \frac{z^4}{20} \pm \ldots.$$

As $c \leqslant \frac{1}{2}$, it suffices to prove $-\frac{z^2}{6} + \frac{z^3}{12} - \frac{z^4}{20} \pm \ldots \leqslant 0$, i.e.,

$$\sum_{i=2}^{\infty} \frac{(-1)^i z^i}{i(i+1)} \geqslant 0.$$

We rewrite this as

$$\sum_{i=2}^{\infty} \frac{1}{z} \frac{(-1)^i z^{i+1}}{i(i+1)} = \sum_{i=2}^{\infty} \frac{1}{z} \int_0^z \frac{(-1)^i t^i}{i} \, dt = \frac{1}{z} \int_0^z \sum_{i=2}^{\infty} \frac{(-1)^i t^i}{i} \, dt.$$

The result now follows, since

$$\sum_{i=2}^{\infty} \frac{(-1)^i t^i}{i} = -\sum_{i=1}^{\infty} \frac{(-1)^{i+1} t^i}{i} + t = -\ln(1+t) + t \geqslant 0,$$

where the last inequality follows from $t \geqslant \ln(1+t)$ for $t \geqslant 0$. ■

Now we show that $f_n(x) \geqslant 0$ using Proposition 5.9.

Lemma 5.10. *$f_n(x) \geqslant 0$ for all $n \geqslant 1$ and $x \in [0, \bar{x}]$.*

Proof. We split the proof into the following parts which together imply the result. All derivatives are with respect to x.

(i) $f_n(0) = 0$ for all $n \geqslant 1$,

(ii) $f_n(\bar{x}) \geqslant 0$ for all $n \geqslant 1$,

(ii) $f_n'(0) > 0$ for all $n \geqslant 1$,

(iv) $f_n'(\bar{x}) < 0$ for all $n \geqslant 1$,

(v) $f_n'''(x) > 0$ for all $x \in [0, \bar{x}]$ and $n \geqslant 1$.

First we show how the lemma follows from these parts, see Fig. 5.1 for an illustration.

Assume (i)-(v) hold. We prove $f_n(x) \geqslant 0$ by contradiction, so assume that for some n there exists an $x_1 \in [0, \bar{x}]$ such that $f_n(x_1) < 0$. Note that, by (i) and (ii), the function decreases in a part of the interval $(0, x_1)$ and increases in a part of the interval $(x_1, \bar{x})$. Hence, as $f_n(x)$ is differentiable, there exist in particular an x_1 such that not only $f_n(x_1) < 0$ but moreover $f_n'(x_1) = 0$. Since the function increases from a negative value in x_1 to a non-negative value in $\bar{x}$, there exists some $x_2 \in (x_1, \bar{x})$ such that $f_n'(x_2) > 0$. However, as $f_n'(x_1) = 0$, $f_n'(x_2) > 0$ and $f_n'(\bar{x}) < 0$ from (iv), there exists an $x_3 \in (x_1, \bar{x})$ with $f_n''(x_3) = 0$.

Similarly, the function decreases from 0 in 0 to a negative value in x_1, so there exists some $x_4 \in (0, x_1)$ such that $f_n'(x_4) < 0$. Again, as $f_n'(0) > 0$ from (iii), $f_n'(x_4) < 0$ and $f_n'(x_1) = 0$, there also exists an $x_5 \in (0, x_1)$ with $f_n''(x_5) = 0$. However, from (v) we know that f'' is strictly increasing in x, so there cannot be two distinct values x_3 and x_5 such that $f_n''(x_3) = f_n''(x_5) = 0$. Contradiction.

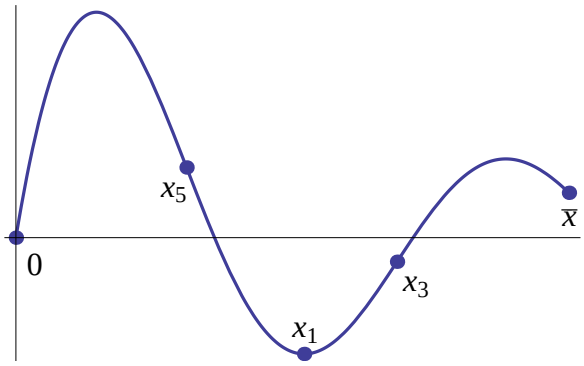

Figure 5.1: An illustration to clarify the proof of Lemma 5.10.

To prove the required statements, we compute the first three derivatives of f with respect to x.

$$f_n'(x) = (1-x)^n - \frac{e-1}{2} + \frac{n(e-1)(e-2)}{e}\left(\frac{4x-(e-2)x^2}{(2-(e-2)x)^2}\right),$$
$$f_n''(x) = n\left(-(1-x)^{n-1} + \frac{8(e-1)(e-2)}{e(2-(e-2)x)^3}\right),$$
$$f_n'''(x) = n\left((n-1)(1-x)^{n-2} + \frac{24(e-1)(e-2)^2}{(2-(e-2)x)^4}\right).$$

We finish the proof by proving the five statements.

(i) $f_n(0) = 0$ for all $n \geqslant 1$ by a direct calculation.

(ii) By the proof of Lemma 5.8, $f_n(\bar{x}) \geqslant 0$ for all $n \geqslant 1$ is equivalent to $\varphi_n(z_i^*) \geqslant 1 - 1/e$ for $z_i^* = 0$ and all $n \geqslant 2$. By direct evaluation, we see the latter is true for $n = 2, 3, 4$. Thus, it remains to show for all $n \geqslant 5$ that

$$\frac{n-1+\frac{e-2}{2}}{n}\left(1-\left(1-\frac{1}{n-1+\frac{e-2}{2}}\right)^n\right) \geqslant 1-\frac{1}{e},$$

or equivalently, for all $n \geqslant 4$ that

$$\frac{n-1+\frac{e}{2}}{n+1}\left(1-\left(1-\frac{1}{n-1+\frac{e}{2}}\right)^{n+1}\right) \geqslant 1-\frac{1}{e}.$$

We write this as

$$\left(1-\frac{1}{n-1+\frac{e}{2}}\right)^{n+1} \leqslant 1-\frac{(n+1)(1-\frac{1}{e})}{n-1+\frac{e}{2}} = \frac{\frac{n+1}{e}+\frac{e}{2}-2}{n-1+\frac{e}{2}},$$

and multiplying both sides by $(1-1/(n-1+e/2))^{(e-5)/2}$ yields

$$\begin{aligned}\left(1-\frac{1}{n-1+\frac{e}{2}}\right)^{n+\frac{e}{2}-\frac{3}{2}} &\leqslant \frac{\frac{n+1}{e}+\frac{e}{2}-2}{n-1+\frac{e}{2}}\left(1-\frac{1}{n-1+\frac{e}{2}}\right)^{\frac{e}{2}-\frac{5}{2}}\\ &= \frac{\frac{n+1}{e}+\frac{e}{2}-2}{n-1+\frac{e}{2}}\left(\frac{n-2+\frac{e}{2}}{n-1+\frac{e}{2}}\right)^{\frac{e}{2}-\frac{5}{2}}\\ &= \frac{(\frac{n+1}{e}+\frac{e}{2}-2)(n-1+\frac{e}{2})^{\frac{3-e}{2}}}{(n-2+\frac{e}{2})^{\frac{5-e}{2}}}.\end{aligned}$$

By invoking Proposition 5.9 for the left-hand side, we see that the left-hand side is a non-decreasing function in n with limit $1/e$. For the right-hand side, note that the limit for n to infinity is also $1/e$, and the derivative with respect to n of the right-hand side is

$$4\frac{3-e}{e}(1-(3-e)n)\frac{(2n+e-2)^{\frac{1-e}{2}}}{(2n+e-4)^{\frac{7-e}{2}}},$$

which is negative for $n \geqslant 4$. The proof of (ii) is complete.

(iii) $f_n'(0) = 1-(e-1)/2 > 0$.

(iv) For $n = 1$ and $n = 2$, direct evaluation of $f_1'(\bar{x})$ and $f_2'(\bar{x})$ gives negative values. For $n \geqslant 3$, proving that

$$f_n'(\bar{x}) = \left(1-\frac{1}{n+\frac{e}{2}-1}\right)^n + \frac{(e-1)((e-2)^2+2n(e-4))}{4en} < 0,$$

is equivalent to proving that

$$e\left(1-\frac{1}{n+\frac{e}{2}-1}\right)^n + \frac{(e-1)(e-2)^2}{4n} < \frac{(e-1)(4-e)}{2}.$$

Consider the left-hand side. By invoking Proposition 5.9 for the first term and taking the limit of n to infinity, and using $n \geqslant 3$ for the second term, we get

$$e\left(1-\frac{1}{n+\frac{e}{2}-1}\right)^n + \frac{(e-1)(e-2)^2}{4n} < 1 + \frac{(e-1)(e-2)^2}{12},$$

which is indeed smaller than $\frac{(e-1)(4-e)}{2}$.

(v) Since $0 \leqslant x \leqslant \bar{x} < 1$ for all n, $f_n'''(x)$ consists of sums, products, and quotients of only strictly positive terms. ■

From Lemma 5.8 and Lemma 5.10 we conclude that indeed $\varphi_n(z_i^*) \geqslant 1 - \frac{1}{e}$ for all $z_i^* \in [0,1]$ and $n \geqslant 2$. This concludes the proof of the Bernoulli Selection Lemma.

5.2.2 *Tightness*

We now provide a family of instances that shows that the $1-1/e$ bound in the Bernoulli Selection Lemma is actually best possible. Consider n^2 independent identically distributed Bernoulli random variables with parameter $1/n$ and prizes $b_1 = n/(e-2)$ and $b_i = 1$ for $2 \leqslant i \leqslant n^2$. The expectation of the maximum prize is given by

$$\mathbb{E}\left[\max_{i\in N}\{b_i X_i\}\right] = \frac{1}{e-2} + \left(1-\frac{1}{n}\right)\left(1-\left(1-\frac{1}{n}\right)^{n^2-1}\right) \longrightarrow \frac{1}{e-2}+1\,.$$

In this particular setting, where the Bernoulli random variables are i.i.d., the best strategy is to sort by prize and take some subset with those of higher prize. This means to choose the first random variable and a subset of size $k-1$ of the rest for some $1 \leqslant k \leqslant n^2$. This yields an expected prize that is equal to

$$\left(1-\left(1-\frac{1}{n}\right)^{k}\right)\frac{\frac{n}{e-2}+k-1}{k} \leqslant \left(1-\left(1-\frac{1}{n}\right)^{k}\right)\left(\frac{n}{k(e-2)}+1\right).$$

As $n \to \infty$, the above converges to

$$\max_{0\leqslant x\leqslant n}(1-e^{-x})\left(\frac{1}{x(e-2)}+1\right),$$

where $x = \frac{k}{n}$. The following proposition shows that this expression is maximized at $x = 1$. Hence, this strategy yields the value $(1-e^{-1})\left(\frac{1}{e-2}+1\right) = (1-1/e)\mathbb{E}[\max_{i\in N}\{b_i X_i\}]$.

Proposition 5.11. *The function* $f(x) = (1-e^{-x})\left(\frac{1}{x(e-2)}+1\right)$ *has a global maximum in* $x = 1$.

Proof. We compute the first two derivatives and find

$$f'(x) = \frac{-e^x + (e-2)x^2 + x + 1}{(e-2)x^2} e^{-x},$$
$$f''(x) = \frac{2e^x - \left((e-2)x^3 + x^2 + 2x + 2\right)}{(e-2)x^3} e^{-x}.$$

We see that $f'(1) = 0$. To show that $x = 1$ is a global maximum, we prove that $f'(x) > 0$ for $x < 1$ and $f'(x) < 0$ for $x > 1$.

To see this, first note that $f''(x)$ has the same sign as the function

$$g(x) = 2e^x - \left((e-2)x^3 + x^2 + 2x + 2\right).$$

Note further that $g(0) = 0$. Since this is an exponential function with a positive coefficient minus a polynomial with only positive coefficients, $g(x)$ first decreases until some point because of the polynomial, after which it is increasing because of the exponential term that starts to dominate the polynomial. So there exists some $x^* > 0$ such that $g(x) < 0$ for $x < x^*$, $g(x^*) = 0$ and $g(x) > 0$ for $x > x^*$. Since $g(1) = e - 3 < 0$, we know $x^* > 1$. Therefore, $f''(x) < 0$ up to $x^* > 1$ and $f''(x) > 0$ afterwards, and hence, $f'(x)$ is decreasing up to $x^* > 1$ and increasing afterwards.

Since $f'(1) = 0$ and $f'(x)$ is decreasing for $x \leqslant 1$, we know $f'(x) > 0$ for $x < 1$.

Furthermore, $f'(x) < 0$ for $1 < x \leqslant x^*$, since $f'(1) = 0$ and $f'(x)$ is decreasing for $1 < x < x^*$. Since $\lim_{x\to\infty} f'(x) = 0$ and $f'(x)$ is increasing from x^* onwards, we know $f'(x) < 0$ for $x > x^*$, and hence, $f'(x) < 0$ for all $x > 1$.

Therefore, $x = 1$ is a global maximum. ■

5.2.3 *Prophet inequality*

The Bernoulli Selection Lemma can be extended to more general random variables, i.e. to the prophet inequality setting. We are now ready to prove Theorem 5.3.

Theorem 5.3. *Given n independent non-negative random variables $X_1, \ldots, X_n$ with $X_i \sim F_i$. There exist values $\tau_1, \ldots, \tau_n$ such that*

$$\mathbb{E}\left[\frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n Y_i}\right] \geqslant \left(1 - \frac{1}{e}\right) \mathbb{E}\left[\max_{i=1,\ldots,n} X_i\right],$$

where Y_i is a Bernoulli random variable that has value 1 if $X_i > \tau_i$.

Proof. First assume that the F_i are continuous for all i. Let $q_i = \mathbb{P}(X_i \geqslant X_j,\ \forall j = 1,\dots,n)$ be the probability that X_i is the largest and let α_i be a value for which $1 - F_i(\alpha_i) = q_i$. Consider $b_i = \mathbb{E}[X_i \mid X_i > \alpha_i]$ and the Bernoulli random variables $Z_1,\dots,Z_n$ where Z_i has parameter q_i. We apply the Bernoulli Selection Lemma to this instance, and thus let $S \subseteq \{1,\dots,n\}$ be a set for which the lemma holds. Now define $\tau_i = \alpha_i$ for $i \in S$ and $\tau_i = \infty$ otherwise, and note that for $i \notin S$, we have $Y_i = 0$ almost surely, and for $i \in S$, we have $\mathbb{P}(X_i > \alpha_i) = \mathbb{P}(Y_i = 1) = q_i$. It follows that

$$\begin{aligned}
\mathbb{E}\left[\frac{\sum_{i=1}^n X_i Y_i}{\sum_{i=1}^n Y_i}\right] &= \sum_{i\in S} \mathbb{E}\left[\frac{X_i Y_i}{\sum_{j\in S} Y_j}\right] \\
&= \sum_{i\in S} \mathbb{E}[X_i \mid Y_i = 1]\mathbb{E}\left[\left(1 + \sum_{j\in S\setminus\{i\}} Y_j\right)^{-1} \middle| Y_i = 1\right] \mathbb{P}(Y_i = 1) \\
&= \sum_{i\in S} \mathbb{E}[X_i \mid X_i > \alpha_i]\mathbb{E}\left[\frac{Y_i}{\sum_{j\in S} Y_j}\right] \\
&= \mathbb{E}\left[\frac{\sum_{i\in S} \mathbb{E}[X_i \mid X_i > \alpha_i] Z_i}{\sum_{i\in S} Z_i}\right] \\
&\geqslant \frac{e-1}{e} \max_{z_i \leqslant q_i} \left\{ \sum_{i=1}^n \mathbb{E}[X_i \mid X_i > \alpha_i] z_i \,\middle|\, \sum_{i=1}^n z_i \leqslant 1 \right\} \\
&\geqslant \frac{e-1}{e} \sum_{i=1}^n \mathbb{E}[X_i \mid X_i > \alpha_i] q_i \,,
\end{aligned}$$

where the second to last inequality follows from the Bernoulli Selection Lemma, while the last holds since $\sum_{i=1}^n q_i = 1$. Now note that we can write $\mathbb{E}\left[\max_{i=1,\dots,n} X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i \mid X_i \geqslant X_j,\ \forall j = 1,\dots,n] q_i$. To finish the proof, it suffices to show that

$$\mathbb{E}[X_i \mid X_i > \alpha_i] \geqslant \mathbb{E}[X_i \mid X_i \geqslant X_j,\ \forall j = 1,\dots,n] \,.$$

Indeed, if $x > \alpha_i$, we have $\mathbb{P}(X_i > x \mid X_i > \alpha_i) = \int_x^\infty \frac{1}{q_i}\, dF_i(t)$, while, if $x \leqslant \alpha_i$, this equals 1. On the other hand,

$$\mathbb{P}(X_i > x \mid X_i \geqslant X_j\ \forall j = 1,\dots,n) = \int_x^\infty \frac{\prod_{j\neq i} F_j(t)}{q_i}\, dF_i(t) \,.$$

It follows that $\mathbb{P}(X_i > x \mid X_i > \alpha_i) \geqslant \mathbb{P}(X_i > x \mid X_i \geqslant X_j,\ \forall j = 1,\dots,n)$ for all $x \geqslant 0$. Thus we can conclude that $X_i \mid (X_i > \alpha_i)$ stochastically dominates $X_i \mid (X_i \geqslant X_j\ \forall j = 1,\dots,n)$, and the conclusion follows.

When some F_i are not continuous, it could be the case that there is no α_i such that $1 - F_i(\alpha_i) = q_i$ or that $\sum q_i > 1$. If the former happens, the result still holds provided α_i is chosen randomly. The latter case is solved by slightly perturbing the support of the random variables in a way that the probability that two or more are the maximum simultaneously is negligible. ■

5.3 ADAPTIVE THRESHOLD RULE

In the previous section we considered the setting in which the threshold value only depends on the random variable X_i, not on the order. In this section we consider the setting in which the threshold value may depend both on X_i and on the prizes that arrived before but got rejected. For i.i.d. random variables we design an adaptive threshold strategy that achieves an expected revenue of at least a $1/\beta^* \approx 0.745$ fraction of the expected maximum value. In particular we prove the following.

Theorem 5.6. *Given non-negative i.i.d. random variables $X_1, \ldots, X_n$, there exist thresholds $\tau_1, \ldots, \tau_n$, such that*

$$\mathbb{E}(\max\{X_1, \ldots, X_n\}) \leqslant \beta^* \mathbb{E}(X_t),$$

where $t := \min\{i \in \{1, \ldots, n\} : X_i \geqslant \tau_i\}$ and $\beta^* \approx 1.3415$ is the unique value such that

$$\int_0^1 \frac{1}{y(1 - \ln(y)) + (\beta - 1)} \, dy = 1 . \tag{5.1}$$

For $X_1, \ldots, X_n$ non-negative i.i.d. random variables, we take F as their probability distribution function and refer to X as a random variable with the same common distribution. Let $F^{-1}(q) = \inf\{x \geqslant 0 \mid F(x) \geqslant q\}$ be the generalized inverse of F (or quantile function) and let $\tau(q) = F^{-1}(1 - q)$. Therefore, we have that $\mathbb{P}(X \geqslant \tau(q)) \geqslant q \geqslant \mathbb{P}(X > \tau(q)) = 1 - F(\tau(q))$, and this holds with equalities if F is continuous at $\tau(q)$. We start by deriving an expression for the expectation of the maximum of $X_1, \ldots, X_n$, for which we let $R(q) = \int_0^q F^{-1}(1 - \theta) \, d\theta$, use Fubini's Theorem, and integration by parts:

$$\begin{aligned} \mathbb{E}(\max\{X_1, \ldots, X_n\}) &= \int_0^\infty 1 - F^n(t) \, dt = \int_0^1 F^{-1}(\sqrt[n]{z}) \, dz \\ &= n \int_0^1 (1 - q)^{n-1} F^{-1}(1 - q) \, dq \end{aligned} \tag{5.6}$$

$$= n \int_0^1 (n-1)(1-q)^{n-2} \left(\int_0^q F^{-1}(1-\theta)\, d\theta \right) dq$$
$$= n \int_0^1 (n-1)(1-q)^{n-2} R(q)\, dq\,.$$

Rather than directly constructing a threshold rule, our approach is to choose at each step a probability of acceptance, which naturally will be increasing as less random variables are left. It is worth mentioning that in this i.i.d. case one could compute the optimal thresholds at each step, however the analysis of such an optimal strategy becomes difficult. Here we use a less direct algorithm but obtain two big advantages: first our thresholds are explicit, and second the posterior analysis becomes quite simple.

Specifically when faced with a random variable X, we will select a proper acceptance probability q. Now if F is continuous at $\tau(q)$ we stop if $X \geqslant \tau(q)$ (so the acceptance probability is q). Otherwise (there is mass at $\tau(q)$), there may be no value that accomplishes the previous condition, so we stop if $X > \tau(q)$ and if $X = \tau(q)$ we stop with probability $s = [q - \mathbb{P}(X > \tau(q)]/\mathbb{P}(X = \tau(q))$ (so again the acceptance probability is q). The goal behind this seemingly obscure rule is that, if at a given step we are faced with a random variable X and have decided on an acceptance probability q, the expected reward in that step equals $R(q)$. Indeed, the reward can be calculated as:

$$\begin{aligned} R(q) &= \mathbb{P}(X = \tau(q)) s\tau(q) + \mathbb{P}(X > \tau(q))\mathbb{E}[X | X > \tau(q)] \\ &= (q - \mathbb{P}(X > \tau(q)))\tau(q) + \int_0^\infty \mathbb{P}(X > t, X > \tau(q))\, dt \\ &= (q + F(\tau(q)) - 1)\tau(q) + \int_{\tau(q)}^\infty 1 - F(t)\, dt = \int_0^q F^{-1}(1-\theta)\, d\theta. \end{aligned}$$

Quantile stopping rule

Our stopping rule is constructed as follows, see Fig. 5.2 for an illustration. We first partition the interval $A = [0,1]$ into n intervals $A_i = [\varepsilon_{i-1}, \varepsilon_i]$, with $0 = \varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_{n-1} < \varepsilon_n = 1$. For X_i we draw an acceptance probability q_i at random from the interval A_i, according to probability density function $f_i(q) = \frac{\psi(q)}{\gamma_i}$, where $\psi(q) = (n-1)(1-q)^{n-2}$ and γ_i is a normalization parameter equal to $\gamma_i = \int_{q \in A_i} \psi(q)\, dq$. As this q_i is the acceptance probability of variable X_i, the corresponding threshold at step i is $\tau_i = \tau(q_i)$. With all the q_i's at hand we scan $X_1, \ldots, X_n$ and stop at

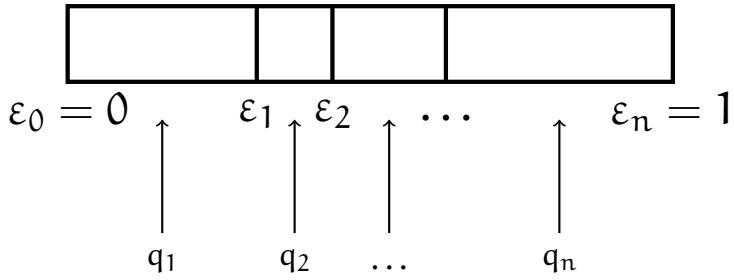

Figure 5.2: A graphical illustration of our quantile stopping rule for the adaptive setting. The intuition is that the acceptance probabilities increase as less customers are remaining, and hence, the thresholds will be decreasing.

step i with probability q_i using the previously described rule (i.e., if F is continuous at $\tau(q_i)$ we stop if $X \geqslant \tau(q_i)$; otherwise we stop for sure if $X > \tau(q_i)$, and if $X = \tau(q_i)$ we stop with probability $s_i = [q_i - \mathbb{P}(X > \tau(q_i))]/\mathbb{P}(X = \tau(q_i))$).

Lemma 5.12. *Let* $\rho_1 = \frac{1}{\gamma_1}$ *and* $\rho_{i+1} = \frac{\rho_i}{\gamma_{i+1}} \int_{\varepsilon_{i-1}}^{\varepsilon_i} \psi(q)(1-q)\,dq$ *for* $i = 1, \ldots, n-1$. *Then the expected value of the random variable at which we stop,* X_r, *satisfies*

$$\mathbb{E}(X_r) = \sum_{i=1}^{n} \rho_i \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} R(q)\,dq\,.$$

Proof. We have already shown that the expected value at step i equals $R(q_i)$. On the other hand, the probability that we get to step i is equal to $\prod_{j=1}^{i-1}(1-q_j)$. Hence, by linearity of expectation and independence of the q_i's, the expected value of X_r is:

$$\begin{aligned}
\mathbb{E}(X_r) &= \sum_{i=1}^{n} \mathbb{E}(R(q_i)) \prod_{j=1}^{i-1} \mathbb{E}(1-q_j) \\
&= \sum_{i=1}^{n} \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} R(q)\,dq \frac{\prod_{j=1}^{i-1} \int_{\varepsilon_{j-1}}^{\varepsilon_j} \psi(q)(1-q)\,dq}{\prod_{j=1}^{i} \gamma_i} \\
&= \sum_{i=1}^{n} \rho_i \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} R(q)\,dq\,,
\end{aligned}$$

and the proof is complete. ∎

Our stopping rule still has freedom in the choice of $\varepsilon_1, \ldots, \varepsilon_{n-1}$. The next lemma says that an appropriate choice leads to express the expected value X_r in terms of that of the expectation of the maximum.

Lemma 5.13. *If we choose $\varepsilon_1, \ldots, \varepsilon_{n-1}$ such that $\rho_1 = \rho_2 = \ldots = \rho_n$, then*

$$\mathbb{E}(\max\{X_1, \ldots, X_n\}) = n\gamma_1 \mathbb{E}(X_r)\,.$$

Proof. If we choose $\varepsilon_1, \ldots, \varepsilon_{n-1}$ such that $\rho_i = \rho_1 = \frac{1}{\gamma_1}$ for all i, then by equation (5.6) and Lemma 5.12 we can express

$$\begin{aligned}
\mathbb{E}(\max\{X_1, \ldots, X_n\}) &= n \int_0^1 (n-1)(1-q)^{n-2} R(q)\, dq \\
&= n\gamma_1 \rho_1 \sum_{i=1}^{n} \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} R(q)\, dq \\
&= n\gamma_1 \sum_{i=1}^{n} \rho_i \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} R(q)\, dq \\
&= n\gamma_1 \mathbb{E}(X_r)\,.
\end{aligned}$$

■

Bounding γ_1

Since $\rho_{i+1} = \frac{\rho_i}{\gamma_{i+1}} \int_{\varepsilon_{i-1}}^{\varepsilon_i} \psi(q)(1-q)\, dq$ for all i, choosing $\varepsilon_1, \ldots, \varepsilon_{n-1}$ such that all ρ_i are equal amounts to choosing them such that $\int_{\varepsilon_i}^{\varepsilon_{i+1}} \psi(q)\, dq = \int_{\varepsilon_{i-1}}^{\varepsilon_i} \psi(q)(1-q)\, dq$ for all i. By the definition of $\psi(q)$, this is equivalent to choosing them such that for all i

$$\frac{n-1}{n}\left((1-\varepsilon_{i-1})^n - (1-\varepsilon_i)^n\right) = (1-\varepsilon_i)^{n-1} - (1-\varepsilon_{i+1})^{n-1}\,.$$

Substituting $x_i = 1 - \varepsilon_i$ we obtain the following equivalent recursion on x_i:

$$\frac{{x_{i-1}}^n}{n} - \frac{{x_i}^n}{n} = \frac{{x_i}^{n-1}}{n-1} - \frac{{x_{i+1}}^{n-1}}{n-1}\,, \tag{5.7}$$

where $x_0 = 1$ and $x_n = 0$. With these boundary constraints, we can write this recursion as follows:

$$x_{i+1} = \left(\frac{n-1}{n}{x_i}^n - \frac{\alpha_n}{n}\right)^{1/(n-1)}\,, \tag{5.8}$$

where $\alpha_n = n - 1 - n{x_1}^{n-1}$ is the recursion from Hill and Kertz [58]. Indeed, for $i = 1$ equation (5.7) gives

$$x_2 = \left({x_1}^{n-1} + \frac{n-1}{n}{x_1}^n - \frac{n-1}{n}\right)^{1/(n-1)}.$$

Now, suppose the claim is true for $i = 1, \dots, j$. From (5.7), we have that

$$\begin{aligned}
{x_{j+1}}^{n-1} &= {x_j}^{n-1} + \frac{n-1}{n}{x_j}^n - \frac{n-1}{n}{x_{j-1}}^n \\
&= \frac{n-1}{n}{x_{j-1}}^n + {x_1}^{n-1} - \frac{n-1}{n} + \frac{n-1}{n}{x_j}^n - \frac{n-1}{n}{x_{j-1}}^n \\
&= \frac{n-1}{n}{x_j}^n + {x_1}^{n-1} - \frac{n-1}{n}.
\end{aligned}$$

Note that our quantity of interest γ_1 equals $\int_0^{\varepsilon_1} \psi(q)\, dq = 1 - {x_1}^{n-1}$. So that if $n(1 - {x_1}^{n-1}) \leqslant \beta^*$, the expected value of the maximum is at most that of our stopping rule times β^*. We remark that the value of x_1 (and all of the recursion) depends on n, but we have omitted this dependency for simplicity.

Observe that $n(1 - {x_1}^{n-1}) \leqslant \beta$ is equivalent to $x_1 \geqslant (1 - \frac{\beta}{n})^{1/(n-1)}$. Thus, if we find the minimum value of β such that $x_1 < (1 - \frac{\beta}{n})^{1/(n-1)}$ implies $x_n < 0$, we know that $x_1 \geqslant (1 - \frac{\beta}{n})^{1/(n-1)}$ for that value of β. Hence, we proceed by showing an upper bound on the value of x_n.

Comparing this to Hill and Kertz [58], they prove that the smallest possible value a_n that satisfies their initial recurrence is equal to $1 + \alpha_n$, and therefore we can write a_n in terms of this new recursion as $a_n = n(1 - {x_1}^{n-1})$. By bounding γ_1, we thus also bound their quantity of interest and prove their conjecture.

Bounding the recursion through a differential equation

In the following, we show that each of the terms x_i in the recursion can be upper bounded by a function $y(t) : [0, 1] \to \mathbb{R}$, defined through the following differential equation. All derivatives of y are with respect to t.

$$\begin{aligned}
y' &= y(\ln(y) - 1) - (\beta - 1), \\
y(0) &= 1.
\end{aligned} \tag{ODE}$$

Furthermore, $y(1) := \lim_{t \uparrow 1} y(t)$ is the continuous extension of $y(t)$.

Later on, we will choose $\beta = \beta^* \approx 1.3415$. For this β, we have $y \in [0,1]$, so we restrict our analysis of (ODE) to this interval. We assume $\beta > 1.25$ and $y \in [0,1]$. We validate these assumptions at the end of our analysis.

Lemma 5.14. *Differential equation* (ODE) *has a unique solution* $y(t)$*, which is a decreasing and strictly convex function on the interval* $[0,1]$*. Furthermore,* $y'''(t) > 0$ *for* $y \in (0,1)$*.*

Proof. Note that $y'(0) = -\beta < 0$ because $y(0) = 1$. For $y \in (0,1]$, we know $\ln(y) \leqslant 0$. Also, as $\beta > 1$, we conclude $y'(t) < 0$. Furthermore, $y(t)$ is convex as for $y \in [0,1)$,

$$y'' = y'(\ln(y) - 1) + y\frac{y'}{y} = y' \ln(y) > 0\,,$$

and $y'' = 0$ for $y = 1$. Finally,

$$\begin{aligned} y''' &= y'' \ln(y) + y'\frac{y'}{y} = y' \ln^2(y) + \frac{(y')^2}{y} \\ &= y' \left(\ln^2(y) + \ln(y) - 1 - \frac{\beta - 1}{y} \right). \end{aligned}$$

We show that $\ln^2(y) + \ln(y) - 1 - \frac{\beta-1}{y} < 0$ for $y \in (0,1)$ or, equivalently, that $g(y) = y\ln^2(y) + y\ln(y) - y - \beta + 1 < 0$ for $y \in (0,1)$. To determine the maximum value of $g(y)$, observe that

$$\begin{aligned} \frac{dg(y)}{dy} &= \ln^2(y) + 2y\ln(y)\frac{1}{y} + \ln(y) + y\frac{1}{y} - 1 \\ &= \ln^2(y) + 3\ln(y) = \ln(y)\,(\ln(y) + 3)\,. \end{aligned}$$

Note that $\frac{dg(y)}{dy} \geqslant 0$ on $y \in (0, e^{-3})$ and $g'(y) < 0$ on $y \in (e^{-3}, 1)$. Hence, since $g(y)$ is continuous, its maximum is attained at $y = e^{-3}$, and $g(e^{-3}) = 5e^{-3} - \beta + 1 < 0$ as $\beta > 1.25$.

Moreover, note that if $y \in (0,1)$, then $|y''|$ is bounded, and hence y' is Lipschitz continuous. Therefore, by the Picard-Lindelöf Theorem [76], $y(t)$ is unique on $(0,1)$. As $y(0)$ is given, and we defined $y(1)$ as the continuous extension of $y(t)$, the solution $y(t)$ is unique on $[0,1]$. ∎

We now proceed to prove that the solution of (ODE) dominates the terms of the recurrence. In this proof, we make use of the following technical result.

Proposition 5.15. *If* $x \in (0,1]$ *and* $n \geqslant 2$*, then*

$$x + \frac{x(\ln(x)-1)}{n} + \frac{\ln(x)(x(\ln(x)-1)-(\beta-1))}{2n^2} \geqslant \frac{n-1}{n} x^{\frac{n}{n-1}}.$$

Proof. Fix a value for n. Since $-(\beta-1)\ln(x)$ is positive, and since $x > 0$, it suffices to prove that

$$f(x) := 1 + \frac{\ln(x)-1}{n} + \frac{\ln(x)(\ln(x)-1)}{2n^2} - \frac{n-1}{n} x^{\frac{1}{n-1}} \geqslant 0.$$

As $f(1) = 0$ for all n, showing that f is non-increasing completes the proof. Noting that $f'(x) = \frac{1}{nx}\left(1 - x^{\frac{1}{n-1}} + \frac{1}{2n}(2\ln(x)-1)\right)$, we have that for $x \in (0,1]$, f' has the same sign as $g(x) := 1 - x^{\frac{1}{n-1}} + \frac{1}{2n}(2\ln(x)-1)$. We prove that g has a maximum $x^* \in (0,1]$ with $g(x^*) \leqslant 0$. This implies that both g and f' are non-positive. Indeed,

$$g'(x) = \frac{1}{nx} - \frac{x^{\frac{1}{n-1}-1}}{n-1}, \quad g''(x) = -\frac{1}{nx^2} + \frac{n-2}{(n-1)^2} x^{\frac{1}{n-1}-2},$$

So $g'(x^*) = 0$ only when $x^* = (\frac{n-1}{n})^{n-1}$. Furthermore, g'' has the same sign as $h(x) := -\frac{1}{n} + \frac{n-2}{(n-1)^2} x^{\frac{1}{n-1}}$, which is an increasing function in x for all $n \geqslant 2$. As $h(1) = -\frac{1}{n(n-1)^2} < 0$, g'' is negative, g is concave and attains its maximum at x^*. Finally,

$$g(x^*) = \frac{1}{2n} + \frac{n-1}{n}\ln\left(1-\frac{1}{n}\right) \leqslant \frac{1}{2n} + \frac{n-1}{n}\left(-\frac{1}{n}\right) = \frac{1}{n^2} - \frac{1}{2n} \leqslant 0,$$

where the last inequality follows from $n \geqslant 2$. This concludes the proof. ∎

Lemma 5.16. *If* $x_1 < (1-\frac{\beta}{n})^{\frac{1}{n-1}}$*, then* ${x_i}^{n-1} < y(\frac{i}{n})$ *for* $i = 1, \ldots, n$*, where* $y(t)$ *is the unique solution of* (ODE).

Proof. First note that $x_0 = y(0) = 1$, by definition. Moreover, we already saw that $y'(0) = -\beta$. As $y(t)$ is strictly convex and, by the assumption of the lemma, we have that $y(1/n) > y(0) - \frac{1}{n}\beta > {x_1}^{n-1}$, proving the statement for $i = 1$. We proceed by induction assuming that ${x_i}^{n-1} < y(\frac{i}{n})$. The Taylor expansion of $y(\frac{i+1}{n})$ around $\frac{i}{n}$ states that there exists $\zeta \in [\frac{i}{n}, \frac{i+1}{n}]$ such that

$$y(\tfrac{i+1}{n}) = y(\tfrac{i}{n}) + \frac{1}{n}y'(\tfrac{i}{n}) + \frac{1}{2n^2}y''(\tfrac{i}{n}) + \frac{1}{6n^6}y'''(\zeta).$$

Thus,

$$\begin{aligned}
y(\tfrac{i+1}{n}) &> y(\tfrac{i}{n}) + \frac{1}{n}y'(\tfrac{i}{n}) + \frac{1}{2n^2}y''(\tfrac{i}{n}) \\
&= y(\tfrac{i}{n}) + \frac{y'(\tfrac{i}{n})}{n}\left(1 + \frac{\ln(y(\tfrac{i}{n}))}{2n}\right) \\
&= y(\tfrac{i}{n}) + \frac{y(\tfrac{i}{n})(\ln(y(\tfrac{i}{n})) - 1) - (\beta - 1)}{n}\left(1 + \frac{\ln(y(\tfrac{i}{n}))}{2n}\right) \\
&\geqslant \frac{n-1}{n}y(\tfrac{i}{n})^{\frac{n}{n-1}} - \frac{\beta-1}{n} > \frac{n-1}{n}{x_i}^n - \frac{\beta-1}{n} > {x_{i+1}}^{n-1}.
\end{aligned}$$

Here, the first inequality follows from $y''' > 0$, the second from Proposition 5.15, the third from the induction hypothesis, and the last from equation (5.8) and the assumption that ${x_1}^{n-1} < 1 - \frac{\beta}{n}$. ■

Now we are ready to prove Theorem 5.6.

Theorem 5.6. *Given non-negative i.i.d. random variables $X_1, \ldots, X_n$, there exist thresholds $\tau_1, \ldots, \tau_n$, such that*

$$\mathbb{E}(\max\{X_1, \ldots, X_n\}) \leqslant \beta^* \mathbb{E}(X_t),$$

where $t := \min\{i \in \{1, \ldots, n\} : X_i \geqslant \tau_i\}$ and $\beta^* \approx 1.3415$ is the unique value such that

$$\int_0^1 \frac{1}{y(1 - \ln(y)) + (\beta - 1)}\, dy = 1. \tag{5.1}$$

Proof. Consider the thresholds of the optimal threshold strategy, which can be easily computed by dynamic programming trough the recurrence:

$$\begin{cases} \tau_n = 0, & V_n = \mathbb{E}(X) \\ \tau_i = V_{i+1}, & V_i = \mathbb{E}(X | X \geqslant \tau_i), \quad i = n-1, \ldots, 1. \end{cases}$$

Note that under these thresholds, it is irrelevant whether to stop when $X \geqslant \tau_i$ or $X > \tau_i$, since they are constructed such that there is indifference between selecting a variable and keeping its value, or to continue for the expected value of the remaining. Therefore, any stopping rule that uses deterministic thresholds obtains an expected reward less or equal than $\mathbb{E}(X_t)$.

We argue that the expected reward obtained by our randomized threshold rule $\mathbb{E}(X_r)$ is upper bounded by the reward of a rule that uses only

deterministic thresholds. Recall that our stopping rule randomizes at step i every time the corresponding threshold $\tau(q_i)$ has mass, choosing between accepting when $X > \tau(q_i)$ or $X \geqslant \tau(q_i)$. If we denote $\bar{q}_i = \mathbb{P}(X \geqslant \tau(q_i))$ and $\underline{q}_i = \mathbb{P}(X > \tau(q_i))$, then $F^{-1}(1-q)$ is constant in $[\underline{q}_i, \bar{q}_i]$. Thus, $R(q) = \int_0^q F^{-1}(1-\theta)\,d\theta$ is linear in that interval, implying that $E(X_r)$ is linear as a function of q_i when $\tau(q_i)$ has mass. This means that either the strategy that stops in step i whenever $X \geqslant \tau(q_i)$, or the strategy that does so when $X > \tau(q_i)$, attains a larger expected reward than $\mathbb{E}(X_r)$, and because both these strategies use only deterministic thresholds, they are in turn upper bounded by $\mathbb{E}(X_t)$. Then, we know by Lemma 5.13 that

$$\mathbb{E}(\max\{X_1, \ldots, X_n\}) \leqslant n\gamma_1 \mathbb{E}(X_r) \leqslant n\gamma_1 \mathbb{E}(X_t)\,,$$

where $\gamma_1 = 1 - x_1{}^{n-1}$.

It remains to show that $n(1 - x_1^{n-1}) \leqslant \beta^*$ for $\beta^* \approx 1.3415$. Lemma 5.16 yields $x_n < y(1)$, so we choose β such that $y(1) = 0$ to reach a contradiction with the fact that $x_n = 0$. Note that this indeed implies $y \in [0,1]$ as we assumed. Hereto, note that $y(t)$ is invertible by Lemma 5.14. Hence, we can consider t as a function of y, for which we know $t(1) = 0$, and we want to choose β such that $t(0) = 1$. In particular, we have that

$$\begin{aligned} t(1) &= t(0) + \int_0^1 \frac{dt}{dy}\,dy = 1 + \int_0^1 \frac{1}{\frac{dy}{dt}}\,dy \\ &= 1 - \int_0^1 \frac{1}{y(1-\ln(y)) + (\beta - 1)}\,dy. \end{aligned}$$

So β^* is the value such that the last integral equals 1. This yields $\beta^* \approx 1.3415$. ∎

Remark 5.17. A routine exercise shows that the sequence a_n defined by Hill and Kertz [58] exactly equals our $n\alpha_1$. Note here that our α_1 does depend on n, though we have omitted this dependency for simplicity of notation. Thus our result implies that $a_n \leqslant \beta^*$ for all $n > 1$, and by the work of Kertz [71] we know that $a_n \to \beta^*$. Let T_n be the set of stopping rules for $X_1, \ldots, X_n$. Recalling that a_n is the smallest possible value for which

$$\mathbb{E}(\max\{X_1, \ldots, X_n\}) \leqslant a_n \sup\{\mathbb{E}(X_t) : t \in T_n\}, \tag{5.9}$$

we know that β^* is the smallest value satisfying (5.9) for all $n > 1$, and hence, it is tight.

5.4 POSTED PRICE MECHANISMS

5.4.1 *Problem description*

In this section we extend our results to the setting of posted price mechanisms. A seller has a single item to sell to a given set of customers $\mathcal{I}$. We assume that the seller has no value for keeping the item. Customers have independent random valuations for the item with customer $i \in \mathcal{I}$ valuing the item at v_i, drawn from distribution $F_i(\cdot)$. The customers arrive in (uniform) random order, and the goal of the seller is to maximize his expected revenue. As is standard in the literature, we say that a distribution F_i is *regular* if the *virtual value function* $c_i(v) = v - (1 - F_i(v))/f_i(v)$ is non-decreasing, where f_i is the density of F_i. Also here we consider a non-adaptive and an adaptive scenario (cf. Problem 5.1 and Problem 5.2).

Problem 5.18 (Non-adaptive posted price mechanism problem)**.** The seller sets prices $p_i \geqslant 0$ for all $i \in \mathcal{I}$, with the goal of maximizing his expected revenue, defined as

$$\sum_{i \in \mathcal{I}} p_i \mathbb{P}_{\sigma,v} \left[i = \operatorname*{argmin}_{j \in \mathcal{I}} \{\sigma(j) \,|\, v_j \geqslant p_j\} \right],$$

where the probability is taken over the arrival permutation σ and the customers' valuations v.

Problem 5.19 (Adaptive posted price mechanism problem)**.** The seller offers each customer a price as she arrives. So, the seller sets functions $p_i : 2^{\mathcal{I}} \to \mathbb{R}$ for each customer i, such that, if S is the set of customers who already arrived and declined the offer, $p_i(S)$ is the price offered to customer i if she is next to arrive. For an arrival permutation σ, we denote $p_i(\sigma) = p_i(\{\sigma^{-1}(1), \ldots, \sigma^{-1}(\sigma(i) - 1)\})$, and therefore we can write the seller's expected revenue as

$$\mathbb{E}_\sigma \left[\sum_{i \in \mathcal{I}} p_i(\sigma) \mathbb{P}_v \left[i = \operatorname*{argmin}_{j \in \mathcal{I}} \{\sigma(j) \,|\, v_j \geqslant p_j(\sigma)\} \right] \right],$$

where the expectation is taken over the arrival permutation σ, and the probability is taken over the customers' valuations v.

In the next section we will prove Theorem 5.21, which gives us a new reduction from the prophet inequality setting to posted price mechanisms

(cf. [22, 51]). Using this reduction, we will derive tight results for both non-adaptive and adaptive posted price mechanisms (cf. Corollary 5.22 and Corollary 5.24). In particular, we will design a non-adaptive posted price mechanism with expected revenue within a factor of $1 - 1/e$ of Myerson's [83] optimal auction and an adaptive posted price mechanism whose expected revenue is within a factor of $1/\beta^* \approx 0.745$ of the expected revenue of the optimal auction.

One may think here that the right benchmark should be the expectation of the maximum valuation. However, this cannot yield useful results. Consider a single customer whose valuation lies in $[1, +\infty)$ distributed according to $F(v) = 1 - 1/v$. Clearly, if we charge price p the acceptance probability is $1/p$, for a total revenue of 1. On the other hand, the expectation of the valuation is actually $+\infty$. This example can easily be turned into one with finite expectation but arbitrarily large ratio between the optimal pricing and the expectation of the random variable.

5.4.2 *Reduction*

In this section we extend the results for the prophet inequality setting to posted price mechanisms by deriving a general reduction principle. Similar reductions have been developed by [9, 22, 50]. We provide an alternative proof showing the applicability of this construction. We start with the following lemma.

Lemma 5.20. *Let X be the positive part of the virtual valuation of a non-negative random variable V with regular distribution F. For any $\tau \geqslant 0$ let $q = \mathbb{P}(X > \tau)$. Then $\mathbb{E}(X \mid X > \tau) = F^{-1}(1 - q)$.*

If the distribution is non-regular, there exist $q_1, q_2, x \in [0, 1]$ such that $xq_1 + (1-x)q_2 = q$ and

$$\mathbb{E}(X \mid X > \tau) = \frac{xq_1 F^{-1}(1 - q_1) + (1-x)\, q_2 F^{-1}(1 - q_2)}{q}.$$

Proof. Let $\bar{c}$ be the ironed virtual value function. Recall that $\bar{c}(v) = G'(F(v))$, where G is the convexification of the negative revenue function $H(\theta) = -(1-\theta)F^{-1}(\theta)$. Then

$$\mathbb{E}(X \mid X > \tau)\, \mathbb{P}(X > \tau) = \int_{\tau}^{\infty} \bar{c}(u)\, dF(u) = \int_{0}^{q} G'(1-\theta)\, d\theta = -G(1-q).$$

In the case of a regular distribution, $G = H$ and the result follows. ∎

Since the events $X_i > \tau_i$ and $V_i \geqslant F_i^{-1}(1-q_i)$ are equivalent, it follows that if the virtual value function φ is increasing, then $\mathbb{E}(X \mid X > \tau) = c^{-1}(\tau)$.

Using this lemma we can prove the following reduction.

Theorem 5.21. *Consider a sequential auction of one item where each customer $i \in N$ has a private valuation $v_i \sim F_i$. Suppose there exists a threshold rule for non-negative random variables $X_i \sim F_i$ that are presented following the same sequence as the customers in the auction, achieving an expected reward of α times the expected maximum. Then there exists a posted price mechanism for the auction that achieves a revenue of α times the optimal auction on N.*

Proof. Let $V_1, \ldots, V_n$ be the valuation functions of the customers. The expected revenue of a posted price mechanism that offers prices $v_1, \ldots, v_n$ equals

$$\sum_{i=1}^{n} v_i \mathbb{P}(V_i \geqslant v_i, i \text{ is the first accepting customer}) .$$

Let $X_1, \ldots, X_n$ be the positive part of the virtual valuation functions of $V_1, \ldots, V_n$. Consider non-negative thresholds $\tau_1, \ldots, \tau_n$ over them and let $q_i = \mathbb{P}(X_i > \tau_i)$. Let r be the index of the first customer whose virtual value is above his threshold. In case of regular distributions, the expected virtual value is

$$\begin{aligned}
\mathbb{E}(X_r) &= \sum_{i=1}^{n} \mathbb{E}(X_i \mid i = r)\, \mathbb{P}(i = r) \\
&= \sum_{i=1}^{n} \mathbb{E}(X_i \mid X_i > \tau_i)\, \mathbb{P}(X_i > \tau_i), i \text{ is the first accepting customer}) \\
&= \sum_{i=1}^{n} F_i^{-1}(1-q_i) \cdot \\
&\qquad \mathbb{P}\left(V_i \geqslant F_i^{-1}(1-q_i), i \text{ is the first accepting customer}\right) .
\end{aligned}$$

Since the expected virtual valuation is equal to the expected revenue (see e.g. [22]), the expected value of the threshold rule over the virtual valuations equals the expected revenue of the posted price mechanism over the valuations. If the distribution of some customer is irregular, randomizing between two prices for him maintains the result.

Since any threshold threshold derived from a threshold rule over the non-negative virtual valuations will be non-negative, all prices induced by these thresholds will be at least the reservation price for every customer. ∎

5.4.3 *Non-adaptive posted price mechanism*

In this section we prove the following corollary for non-adaptive posted price mechanisms:

Corollary 5.22. *For any given set of potential customers $\mathcal{I}$, there exists a non-adaptive posted price mechanism that achieves an expected revenue of at least a $1-1/e$ fraction of that of Myerson's optimal auction on $\mathcal{I}$.*

Besides the Bernoulli Selection Lemma, key to our analysis is the by now classic result of Chawla et al. [22].

Lemma 5.23 ([22, Lemma 4])**.** *If all value distributions are regular, then the expected value of Myerson's optimal auction is bounded from above by*

$$\sum_{i\in\mathcal{I}} F_i^{-1}(1-q_i^M)q_i^M\,,$$

where q_i^M is the probability that the optimal auction assigns the item to i.

Furthermore, for every i (with regular or non-regular value distribution) there exist two prices $\underline{p_i}$ and $\overline{p_i}$, with corresponding probabilities $\underline{q_i}$ and $\overline{q_i}$, and a number $0\leqslant x_i\leqslant 1$, such that $x_i\underline{q_i}+(1-x_i)\overline{q_i}=q_i^M$, and the expected revenue of Myerson's optimal auction is bounded from above by

$$\sum_{i\in\mathcal{I}} x_i\underline{p_i}\,\underline{q_i}+(1-x_i)\overline{p_i}\,\overline{q_i}\,.$$

Proof of Corollary 5.22. We prove the regular case first. Let q_i^M denote the probability with which Myerson's optimal auction assigns the item to customer $i\in\mathcal{I}$, and set $b_i=F_i^{-1}(1-q_i^M)$. The expected revenue of a non-adaptive posted price mechanism, that chooses to sell only to customers in $S\subseteq\mathcal{I}$ while offering prices b_i, is given by

$$\begin{aligned}
&\sum_{i\in S} b_i\mathbb{P}[i=\operatorname*{argmin}_{j\in S}\{\sigma(j)\,|\,v_j\geqslant b_j\}]\\
&\quad=\sum_{i\in S} b_i q_i^M\mathbb{P}[i=\operatorname*{argmin}_{j\in S}\{\sigma(j)\,|\,v_j\geqslant b_j\}\,|\,v_i\geqslant b_i]\\
&\quad=\sum_{i\in S} b_i q_i^M\sum_{R\subseteq S\setminus\{i\}}\frac{1}{1+|R|}\prod_{j\in R}q_j^M\prod_{j\in S\setminus(R\cup\{i\})}(1-q_j^M)\\
&\quad=\sum_{i\in S} b_i q_i^M\mathbb{E}\left[\tfrac{1}{1+\sum_{j\in S\setminus\{i\}}X_j}\right]=\mathbb{E}\left[\tfrac{\sum_{i\in S}b_iX_i}{\sum_{i\in S}X_i}\right],
\end{aligned}$$

where $\{X_i\}_{i\in\mathcal{I}}$ are Bernoulli random variables with $X_i = 1$ with probability q_i^M. By the Bernoulli Selection Lemma we can choose the set $S \subseteq \mathcal{I}$ to be such that the latter is lower bounded by

$$\left(1-\frac{1}{e}\right)\max_{z_i\leqslant q_i^M}\left\{\sum_{i\in\mathcal{I}} b_i z_i \,\middle|\, \sum z_i \leqslant 1\right\} \geqslant \left(1-\frac{1}{e}\right)\sum_{i\in\mathcal{I}} F_i^{-1}(1-q_i^M)q_i^M .$$

Therefore, Lemma 5.23 leads to the desired conclusion.

In the non-regular case, the posted price mechanism runs a lottery between two prices to get the desired bound. This lottery can be derandomized using standard techniques, since each combination of prices offered to the customers is a deterministic mechanism in itself and the random mechanism is simply a lottery over, and thus a convex combination of, those deterministic mechanisms. First, for every bidder with positive probability of winning the optimal auction, set

$$b_i' = \frac{x_i \underline{p_i}\,\underline{q_i} + (1-x_i)\overline{p_i}\,\overline{q_i}}{q_i^M},$$

where the variables are defined as in the lemma. Also consider the same Bernoulli random variables presented in the first part of the proof. The non-adaptive posted price mechanism sells only to a set S' of customers (to be defined). For every $i \in S'$, it offers a random price p_i equal to $\underline{p_i}$ with probability x_i, and $\overline{p}_i$ otherwise. This way, the a priori probability that v_i is above the price offered is exactly $x_i\underline{q_i} + (1-x_i)\overline{q_i} = q_i^M$, while the expected revenue of the mechanism can be evaluated as

$$\begin{aligned}
&\sum_{i\in S'} x_i \underline{p_i}\,\underline{q_i}\,\mathbb{P}[i = \operatorname*{argmin}_{j\in S'}\{\sigma(j) \mid v_j \geqslant p_j\} \mid v_i \geqslant p_i,\ p_i = \underline{p_i}] \\
&\qquad + (1-x_i)\overline{p_i}\,\overline{q_i}\,\mathbb{P}[i = \operatorname*{argmin}_{j\in S'}\{\sigma(j) \mid v_j \geqslant p_j\} \mid v_i \geqslant p_i,\ p_i = \overline{p_i}] \\
&= \sum_{i\in S'} (x_i \underline{p_i}\,\underline{q_i} + (1-x_i)\overline{p_i}\,\overline{q_i})\cdot \\
&\qquad \sum_{R\subseteq S'\setminus\{i\}} \frac{1}{1+|R|}\prod_{j\in R} q_j^M \prod_{j\in S'\setminus(R\cup\{i\})} (1-q_j^M) \\
&= \sum_{i\in S'} b_i' q_i^M \mathbb{E}\left[\tfrac{1}{1+\sum_{j\in S'\setminus\{i\}} X_j}\right] = \mathbb{E}\left[\tfrac{\sum_{i\in S'} b_i' X_i}{\sum_{i\in S'} X_i}\right].
\end{aligned}$$

By the same argument as before, Lemma 5.5 implies that there exists $S' \subseteq \mathcal{I}$ such that the latter is lower bounded by $(1-1/e)\sum_{i\in\mathcal{I}} b_i' q_i^M$. Lemma 5.23 implies the bound over the optimal auction. ■

In the case of monotone virtual valuations,the algorithm that achieves this result becomes remarkably simple, see Algorithm 5.3.

Algorithm 5.3: Algorithm for the non-adaptive posted price mechanism.

1 Compute q_i = probability that optimal auction assigns to i ;
2 Discard customer i with probability $1 - \frac{2}{2+(e-2)q_i}$;
3 Offer non-discarded customers price $F_i^{-1}(1-q_i)$;
4 Item is allocated to a random customer accepting the offer ;

5.4.4 *Tight instance with i.i.d. valuations*

We construct a family of instances for Problem 5.1 with i.i.d. customer valuations, such that, for all $\varepsilon > 0$, there is an instance from this family for which no non-adaptive strategy can achieve an expected revenue within a factor $(1+\varepsilon)(1-1/e)$ of the optimal expected revenue. The idea is to mimic the instance that makes the Bernoulli Selection Lemma tight, but here we achieve this with i.i.d. valuations. Consider n^2 customers whose values are independent identically distributed according to

$$V = \begin{cases} \frac{n}{e-2} & \text{w.p. } \frac{1}{n^3}\,, \\ 1 & \text{w.p. } \frac{1}{n}\,, \\ 0 & \text{w.p. } 1 - \frac{1}{n} - \frac{1}{n^3}\,. \end{cases}$$

Then, it is easy to design an auction that achieves a revenue approaching $(e-1)/(e-2)$ as $n \to \infty$. Indeed, consider the auction that offers the item for price $n/(e-2) - c$ (with c a small value, say $c = 2$) to any bid above that price (and assigns the item at random if more than one such offer is received), and if no such bid is received, then it runs a lottery at price 1 among all the bids above that price. As there are many buyers of value 1, a potential large value customer will prefer to make a revenue of c rather than risking to lose the item in the lottery. Therefore the revenue the auction will generate will approach $1/(e-2)+1$ as $n \to \infty$. Of course, the revenue of the optimal auction is then at least this quantity. On the other hand, the best posted price mechanism offers a price of 1 to, say, customers $1, \ldots, k$ and $n/(e-2)$ to the rest of the customers, for some well chosen value of k which turns out to be roughly n. One can show that in the limit the revenue

approaches $(\frac{1}{e-2}+1)(1-e^{-1})$. The expected revenue of this mechanism can be computed by recursing on the expected revenue of the remaining buyers. Let $V(j)$ be the expected revenue when there are j customers left. Denoting $p = \frac{k}{n^2}$, the total expected revenue is

$$\begin{aligned} V(n^2) &= (1-p)\frac{1}{n^3}\frac{n}{e-2} + p\left(\frac{1}{n}+\frac{1}{n^3}\right) + \left(1-\frac{1}{n^3}-\frac{p}{n}\right)V(n^2-1) \\ &= \left((1-p)\frac{1}{n^2}\frac{1}{e-2} + p\left(\frac{1}{n}+\frac{1}{n^3}\right)\right)\sum_{i=0}^{n^2-1}\left(1-\frac{1}{n^3}-\frac{p}{n}\right)^i \\ &= \left((1-p)\frac{1}{n(e-2)} + p\left(1+\frac{1}{n^2}\right)\right)\frac{1-\left(1-\frac{1}{n^3}-\frac{p}{n}\right)^{n^2}}{\frac{1}{n^2}+p}. \end{aligned}$$

Letting $x(n) = np + 1/n$, we have that

$$V(n^2) \leqslant \left(\frac{1}{e-2} + x(n)\right)\frac{1-\left(1-\frac{1}{n^2}x(n)\right)^{n^2}}{x(n)}.$$

Thus for all $\varepsilon > 0$, there is a large enough n such that

$$V(n^2) \leqslant \varepsilon + \max_{x\geqslant 0}\left(\frac{1}{x(e-2)}+1\right)(1-e^{-x}).$$

Invoking Proposition 5.11 from Section 5.2.2 yields

$$V(n^2) \leqslant \left(\frac{1}{e-2}+1\right)(1-e^{-1}) + \varepsilon = (1-e^{-1})\mathrm{OPT} + \varepsilon'.$$

5.4.5 *Adaptive posted price mechanism*

Given the tight bound above, we turn our attention to *adaptive* posted price mechanisms, in which the seller may also base the price he is offering to a customer on the set of customers who previously rejected the offer. For this setting, Theorem 5.6 and Theorem 5.21 yield the following corollary.

Corollary 5.24. *For any given set of potential customers $\mathcal{I}$ whose values are independent and identically distributed, there exists an adaptive posted price mechanism that achieves an expected revenue of at least a $1/\beta > 0.745$ fraction of that of Myerson's optimal auction on $\mathcal{I}$, where β is the unique value satisfying (5.1).*

However, deriving a simple algorithm from the previous characterization is not straightforward. In this section we will provide the details of the algorithmic construction. For our analysis the bound provided by Lemma 5.23 is not enough, so we derive an exact expression for the expected revenue of the optimal auction for i.i.d. customers.

Expected value of Myerson's optimal auction for i.i.d. customers

We assume that the valuations of the customers are i.i.d. with cumulative distribution function $F(\cdot)$ and probability density function $f(\cdot)$. As before we define the *virtual valuation* as $c(v) = v - \frac{1-F(v)}{f(v)}$ and the *ironed virtual valuation* as $\bar{c}(v) = G'(F(v))$, where $G = \text{conv}(H)$ is the convexification of the negative revenue curve $H(q) = \int_0^q c(F^{-1}(\theta))\,d\theta$ as a function of the acceptance probability q. Let $\mathbb{E}(MY(n, F))$ be the expected revenue of the optimal auction over n customers with values drawn from distribution F.

Lemma 5.25. *For a given set of n i.i.d. potential customers with cumulative distribution function $F(\cdot)$, the expected revenue of Myerson's optimal auction is*

$$\mathbb{E}(MY(n, F)) = n(n-1)\int_0^1 (1-q)^{n-2}\bar{G}(1-q)\,dq\,. \tag{5.10}$$

Proof. The expected profit of the optimal auction equals its expected virtual surplus (see, e.g., [56]), i.e., the sum over all customers of the expected value of the maximum of $\bar{c}$ above zero. Note that $\bar{c}$ is an increasing function, and let v^* be the value at which $\bar{c}(v^*) = 0$ or zero, if no such value exists. Then, the latter can be evaluated as:

$$\mathbb{E}(MY(n, F)) = \int_{v^*}^{\infty} nF(v)^{n-1}\bar{c}(v)f(v)\,dv\,.$$

Performing the change of variables $q = 1 - F(v)$ and $\alpha^* = 1 - F(v^*)$, we obtain

$$\begin{aligned}
\mathbb{E}(MY(n, F)) &= n\int_0^{\alpha^*} (1-q)^{n-1}\bar{c}(F^{-1}(1-q))\,dq \\
&= n\int_0^{\alpha^*} (1-q)^{n-1}G'(1-q)\,dq \\
&= -nG(1-q)(1-q)^{n-1}\Big|_0^{\alpha^*} - \int_0^{\alpha^*} n(n-1)(1-q)^{n-2}G(1-q)\,dq \\
&= nG(1) - nG(F(v^*))F(v^*)^{n-1} - n(n-1)\int_0^{\alpha^*} (1-q)^{n-2}G(1-q)\,dq\,.
\end{aligned}$$

Since $\bar{c}(v^*) = 0$, we know that G attains a minimum at $F(v^*)$ and, therefore, equals $H(F(v^*))$ at that point. Now, observe that

$$H(q) = \int_0^q F^{-1}(\theta) - \frac{1-\theta}{f(F^{-1}(\theta))}\,d\theta = -(1-q)F^{-1}(q)\,.$$

Therefore, we can conclude that

$$\begin{aligned}
&\mathbb{E}(MY(n,F)) \\
&= -nH(F(v^*))F(v^*)^{n-1} - n(n-1)\int_0^{\alpha^*}(1-q)^{n-2}G(1-q)\,dq \\
&= nv^*(1-F(v^*))F(v^*)^{n-1} - n(n-1)\int_0^{\alpha^*}(1-q)^{n-2}G(1-q)\,dq\,.
\end{aligned}$$

Now, let

$$\bar{G}(1-q) = \begin{cases} -G(1-q) & \text{if } 1-q > F(v^*)\,, \\ v^*(1-F(v^*)) & \text{otherwise}\,. \end{cases}$$

Then, we can write the expected revenue of the optimal mechanism as Eq. (5.10) and the proof is complete. ■

We note that expression (5.10), although fairly natural to derive, appears to be new.

The adaptive posted price mechanism

In the adaptive setting, the price offered to every customer also depends on the set of customers that previously declined their offer. However, since the customers are i.i.d., it suffices to know only how many customers arrived before the current customers.

As in Section 5.3 we partition the interval $A = [0,1]$ into n intervals $A_i = [\varepsilon_{i-1}, \varepsilon_i]$ with $0 = \varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_n = 1$. We draw an acceptance probability q_i for the i-th customer who arrives from interval A_i according to probability density function $\psi(q_i) = \frac{(n-1)(1-q_i)^{n-2}}{\gamma_i}$, where γ_i is a normalization factor. We offer the customer a price of $\max\{F^{-1}(1-q_i), v^*\}$, where v^* is the reservation price of the optimal auction.

The expected revenue from selling the item to customer i is $\bar{G}(1-q_i)$. To see this, suppose that $q_i < 1 - F(v^*)$. Then, for monotone virtual valuations, the price offered to customer i is $F^{-1}(1-q_i)$, and thus the expected

revenue is $q_i F^{-1}(1-q_i) = -G(1-q_i) = \bar{G}(1-q_i)$. On the other hand, if $q_i > 1 - F(v^*)$, the price offered to customer i is v^* which is accepted with probability $1 - F(v^*)$. Similar arguments hold when the virtual valuation is not monotone, where it might be the case that $q_i F^{-1}(1-q_i) = -H(1-q_i) < \bar{G}(1-q_i)$, and by offering a price $F^{-1}(1-q_i)$ we might not get the best revenue. To circumvent this problem, we can randomize between two acceptance probabilities q_{i1} and q_{i2} such that $G(1-q_i) = \gamma H(1-q_{i1}) + (1-\gamma)H(1-q_{i2})$ and $q_i = \gamma q_{i1} + (1-\gamma)q_{i2}$.

Following the same reasoning as in Section 5.3, we can bound the expected revenue of this adaptive posted price mechanism by

$$\sum_{i=1}^{n} \rho_i \int_{\varepsilon_{i-1}}^{\varepsilon_i} (n-1)(1-q)^{n-2} \bar{G}(1-q)\, dq\,,$$

where $\rho_1 = \frac{1}{\gamma_1}$ and $\rho_{i+1} = \frac{\rho_i}{\gamma_{i+1}} \int_{\varepsilon_{i-1}}^{\varepsilon_i} \psi(q)(1-q)\, dq$ for $i = 1, \ldots, n-1$. Again, if we choose $\varepsilon_1, \ldots, \varepsilon_{n-1}$ such that $\rho_1 = \rho_2 = \ldots = \rho_n$ and solve the recurrence on the ε_i values, this quantity can be lower bounded by

$$\frac{1}{n\gamma_1} \mathbb{E}(MY(n, F)) \geqslant \frac{1}{\beta^*} \mathbb{E}(MY(n, F)) \approx 0.745 \mathbb{E}(MY(n, F))\,.$$

See Algorithm 5.4 for our algorithm in the case of monotone virtual valuations.

Algorithm 5.4: Algorithm for the adaptive posted price mechanism.

1 Partition the interval $[0, 1]$ into intervals $A_i = [a_{i-1}, a_i]$, s.t. $a_0 = 0$, $a_n = 1$;
2 Sample q_i from A_i with an appropriately chosen distribution ;
3 When the i—th buyer comes, offer price $p_i = \max\{F^{-1}(1-q_i), v^*\}$, where v^* is the reservation price of the optimal auction ;

We believe that the bound of Corollary 5.24 is tight. Although the best upper bound known for the i.i.d. case, due to Blumrosen and Holenstein [13], proves that no algorithm can achieve a fraction of at least 0.79, we believe that the family of instances provided by Hill and Kertz [58] in the context of prophet inequalities for i.i.d. random variables can be transformed into a tight family of instances for Corollary 5.24. We remark here that recent work of Dütting, Fischer, and Klimm [30] also studies the benefit of adaptivity in the i.i.d. case, but from a different perspective.

5.5 CONCLUDING REMARKS

This chapter considers two different fields. In the field of optimal stopping theory, we presented a non-adaptive and an adaptive threshold rule maximizing the expected reward of a gambler facing a sequence of non-negative random variables whose realizations arrive over time in a uniform random fashion. In both settings we present an algorithm whose approximation guarantee matches lower bounds obtained from tight instances. In the field of posted price mechanisms, these algorithms translate to pricing schemes for both the non-adaptive and the adaptive setting. For the non-adaptive setting the approximation guarantee is tight, and for the adaptive setting the approximation guarantee is believed to be tight. Formally proving this conjecture would be interesting.

The approach in this chapter is new compared to related literature. In future research, a similar approach can perhaps be taken in related problems in both fields. This new perspective on the analysis of problems like the ones in this chapter could turn out to be fruitful.

Another interesting open direction is to investigate the computational complexity of the algorithms in this chapter. Does the PTAS presented in Cominetti et al. [27] naturally extend to this setting for example? This extension seems to be quite subtle and it is not clear yet whether the PTAS can be extended or not.

Furthermore, the amount of information our stopping rules require from the underlying distributions is little; only quantile distribution information is necessary. Future research could further investigate the trade off between the amount of distribution information and the achievable approximation guarantee.

BIBLIOGRAPHY

[1] M. Abolhassani, S. Ehsani, H. Esfandiari, M. T. Hajiaghayi, R. Kleinberg, and B. Lucier. „Beating 1-1/e for Ordered Prophets". In: *Proceedings of 49th Annual ACM SIGACT Symposium on the Theory of Computing*. STOC '17. ACM, 2017.

[2] H. Ackermann, H. Röglin, and B. Vöcking. „On The Impact of Combinatorial Structure on Congestion Games". In: *Journal of the ACM* 55.6 (2008), pp. 1–22.

[3] M. Adamczyk, A. Borodin, D. Ferraioli, B. de Keijzer, and S. Leonardi. „Sequential Posted Price Mechanisms with Correlated Valuations". In: *Proceedings of the 11th International Conference on Web and Internet Economics - Volume 9470*. WINE 2015. Springer-Verlag New York, Inc., 2015, pp. 1–15.

[4] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid. „Approximation Schemes for Scheduling on Parallel Machines". In: *Journal of Scheduling* 1 (1998), pp. 55–66.

[5] M. Andrews, S. Antonakopoulos, and L. Zhang. „Minimum-Cost Network Design with (Dis)economies of Scale". In: *Proceedings of the 51st Annual IEEE Symposium on the Foundations of Computer Science*. FOCS '10. 2010, pp. 585–592.

[6] S. Antonakopoulos, C. Chekuri, F. B. Shepherd, and L. Zhang. „Buy-at-Bulk Network Design with Protection". In: *Mathematics of Operations Research* 36.1 (2011), pp. 71–87.

[7] D. Assaf, L. Goldstein, and E. Samuel-Cahn. „Ratio prophet inequalities when the mortal has several choices". In: *The Annals of Applied Probability* 12.3 (2002), pp. 972–984.

[8] G. Ausiello, A. D'Atri, and M. Protasi. „Structure Preserving Reductions among Convex Optimization Problems". In: *Journal of Computer and System Sciences* 21.1 (1980), pp. 136 –153.

[9] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. „Online Auctions and Generalized Secretary Problems". In: *SIGecom Exchanges* 7.2 (June 2008), 7:1–7:11.

[10] N. Bansal, T. Oosterwijk, T. Vredeveld, and R. van der Zwaan. „Approximating Vector Scheduling: Almost Matching Upper and Lower Bounds". In: *Algorithmica* 76.4 (2016), pp. 1077–1096.

[11] D. Bertsimas, D. Gamarnik, and J. Sethuraman. „From Fluid Relaxations to Practical Algorithms for High-Multiplicity Job-Shop Scheduling: The Holding Cost Objective". In: *Operations Research* 51.5 (2003), pp. 798–813.

[12] L. Blumrosen and S. Dobzinski. „Welfare Maximization in Congestion Games". In: *IEEE Journal on Selected Areas in Communication* 25.6 (2007), pp. 1224–1236.

[13] L. Blumrosen and T. Holenstein. „Posted Prices vs. Negotiations: An Asymptotic Analysis". In: *Proceedings of the 9th ACM Conference on Electronic Commerce*. EC '08. ACM, 2008, pp. 49–49.

[14] F. F. Boctor. „The Two-Product, Single-Machine, Static Demand, Infinite Horizon Lot Scheduling Problem". In: *Management Science* 28.7 (1982), pp. 798–807.

[15] V. Bonifaci and A. Wiese. „Scheduling Unrelated Machines of Few Different Types". In: *CoRR* abs/1205.0974 (2012).

[16] N. Brauner, Y. Crama, A. Grigoriev, and J. van de Klundert. „A Framework for the Complexity of High-Multiplicity Scheduling Problems". In: *Journal of Combinatorial Optimization* 9.3 (2005), pp. 313–323.

[17] N. Brauner, Y. Crama, A. Grigoriev, and J. van de Klundert. „Multiplicity and Complexity Issues in Contemporary Production Acheduling". In: *Statistica Neerlandica* 61.1 (2007), pp. 75–91.

[18] J. R. Bult and T. Wansbeek. „Optimal Selection for Direct Mail". In: *Marketing Science* 14.4 (1995), pp. 378–394.

[19] C. Calabro, R. Impagliazzo, and R. Paturi. „A Duality between Clause Width and Clause Density for SAT". In: *IEEE Conference on Computational Complexity*. 2006, pp. 252–260.

[20] D. Chakrabarty, A. Mehta, and V. Nagarajan. „Fairness and Optimality in Congestion Games". In: *Proceedings of the 6th ACM Conference on Electronic Commerce*. EC '05. 2005, pp. 52–57.

[21] S. Chawla, J. D. Hartline, and R. Kleinberg. „Algorithmic pricing via virtual valuations". In: *Proceedings of the 8th ACM Conference on Electronic Commerce*. EC 2007. 2007.

[22] S. Chawla, J. D. Hartline, D. L. Malec, and B. Sivan. „Multi-parameter Mechanism Design and Sequential Posted Pricing". In: *Proceedings of the 42nd ACM Symposium on Theory of Computing*. STOC '10. ACM, 2010, pp. 311–320.

[23] C. Chekuri and S. Khanna. „On Multidimensional Packing Problems". In: *SIAM Journal on Computing* 33.4 (2004), pp. 837–851.

[24] V. Choudhary, A. Ghose, T. Mukhopadhyay, and U. Rajan. „Personalized Pricing and Quality Differentiation". In: *Management Science* 51.7 (2005), pp. 1120–1130.

[25] J. J. Clifford and M. E. Posner. „High Multiplicity in Earliness-Tardiness Scheduling". In: *Operations Research* 48.5 (2000), pp. 788–800.

[26] J. J. Clifford and M. E. Posner. „Parallel Machine Scheduling with High Multiplicity". In: *Mathematical Programming* 89.3 (2001), pp. 359–383.

[27] R. Cominetti, J. R. Correa, T. Rothvoß, and J. San Martín. „Optimal Selection of Customers for a Last-Minute Offer". In: *Operations Research* 58.4 (2010), pp. 878–888.

[28] J. R. Correa, P. Foncea, R. Hoeksma, T. Oosterwijk, and T. Vredeveld. „Posted Price Mechanisms for a Random Stream of Customers". In: *Proceedings of the 2017 ACM Conference on Economics and Computation*. EC '17. Cambridge, Massachusetts, USA: ACM, 2017, pp. 169–186.

[29] W. Cunningham. „Improved Bounds for Matroid Partition and Intersection Algorithms". In: *SIAM Journal on Computing* 15.4 (1986), pp. 948–957.

[30] P. Dütting, F. Fischer, and M. Klimm. „Revenue Gaps for Discriminatory and Anonymous Sequential Posted Pricing". In: *ArXiv e-prints* (July 2016).

[31] F. Eisenbrand and G. Shmonin. „Carathéodory Bounds for Integer Cones". In: *Operations Research Letters* 34.5 (2006), pp. 564–568.

[32] L. Epstein and T. Tassa. „Vector Assignment Problems: A General Framework". In: *Journal of Algorithms* 48.2 (2003), pp. 360–384.

[33] L. Epstein and T. Tassa. „Vector Assignment Schemes for Asymmetric Settings". In: *Acta Informatica* 42.6-7 (2006), pp. 501–514.

[34] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, and M. Monemizadeh. „Prophet Secretary". In: *Algorithms - ESA 2015: 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*. Ed. by N. Bansal and I. Finocchi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 496–508.

[35] A. Fabrikant, C. Papadimitriou, and K. Talwar. „The Complexity of Pure Nash Equilibria". In: *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*. Ed. by L. Babai. STOC '04. 2004, pp. 604–612.

[36] P. von Falkenhausen and T. Harks. „Optimal Cost Sharing for Resource Selection Games". In: *Mathematics of Operations Research* 38.1 (2013), pp. 184–208.

[37] U. Feige. „A Threshold of $\ln n$ for Approximating Set Cover". In: *Journal of the ACM* 45.4 (July 1998), pp. 634–652.

[38] C. Filippi and G. Romanin-Jacur. „Exact and Approximate Algorithms for High-Multiplicity Parallel Machine Scheduling". In: *Journal of Scheduling* 12.5 (Oct. 2009), pp. 529–541.

[39] A. Frank and É. Tardos. „An Application of Simultaneous Diophantine Approximation in Combinatorial Optimization". In: *Combinatorica* 7 (1987), pp. 49–65.

[40] S. Fujishige. *Submodular Functions and Optimization*. Annals of discrete mathematics. Amsterdam, Boston, Paris: Elsevier, 2005.

[41] M. Gabay, A. Grigoriev, V. J. C. Kreuzen, and T. Oosterwijk. „High Multiplicity Scheduling with Switching Costs for Few Products". In: *Operations Research Proceedings 2014, Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), RWTH Aachen University, Germany, September 2-5, 2014*. 2014, pp. 437–443.

[42] H. N. Gabow. „Algorithms for Graphic Polymatroids and Parametric s-Sets". In: *Journal of Algorithms* 26.1 (1998), pp. 48–86.

[43] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[44] E. Gassner, J. Hatzl, S. O. Krumke, H. Sperber, and G. J. Woeginger. „How Hard is it to Find Extreme Nash Equilibria in Network Congestion Games?" In: *Theoretical Computer Science* 410.47-49 (2009), pp. 4989–4999.

[45] J. P. Gilbert and F. Mosteller. „Recognizing the Maximum of a Sequence". In: *Journal of the American Statistical Association* 61 (1966), pp. 35–76.

[46] S. K. Goyal. „Scheduling a Multi-Product Single Machine System". In: *Journal of the Operational Research Society* 24.2 (1973), pp. 261–269.

[47] H. Groenevelt. „Two Algorithms for Maximizing a Separable Concave Function over a Polymatroid Feasible Region". In: *European Journal of Operational Research* 54.2 (1991), pp. 227 –236.

[48] K. Haase. „Capacitated Lot-Sizing with Sequence Dependent Setup Costs". In: *Operations-Research-Spektrum* 18.1 (1996), pp. 51–59.

[49] K. Haase and A. Kimms. „Lot Sizing and Scheduling with Sequence-Dependent Setup Costs and Times and Efficient Rescheduling Opportunities". In: *International Journal of Production Economics* 66.2 (2000), pp. 159 –169.

[50] M. T. Hajiaghayi, R. Kleinberg, and D. C. Parkes. „Adaptive Limited-supply Online Auctions". In: *Proceedings of the 5th ACM Conference on Electronic Commerce*. EC '04. New York: ACM, 2004, pp. 71–80. ISBN: 1-58113-771-0.

[51] M. Hajiaghayi, R. Kleinberg, and T. Sandholm. „Automated online mechanism design and prophet inequalities". In: *Proceedings of the 22nd Conference on Artificial Intelligence*. AAAI 2007. 2007.

[52] T. Harks and P. von Falkenhausen. „Optimal Cost Sharing for Capacitated Facility Location Games". In: *European Journal of Operations Research* 239.1 (2014), pp. 187–198.

[53] T. Harks, M. Klimm, and B. Peis. „Resource Competition on Integral Polymatroids". In: *Web and Internet Economics - 10th International Conference, WINE 2014, Beijing, China, December 14-17, 2014. Proceedings*. 2014, pp. 189–202.

[54] T. Harks, T. Oosterwijk, and T. Vredeveld. „A Logarithmic Approximation for Polymatroid Congestion Games". In: *Operations Research Letters* 44.6 (2016), pp. 712–717.

[55] T. Harks, M. Hoefer, M. Klimm, and A. Skopalik. „Computing Pure Nash and Strong Equilibria in Bottleneck Congestion Games". In: *Mathematical Programming* 141.1-2 (2013), pp. 193–215.

[56] J. D. Hartline. *Mechanism Design and Approximation*. 2017.

[57] T. P. Hill. „Prophet inequalities and order selection in optimal stopping problems". In: *Proceedings of the American Mathematical Society* 88.1 (1983), pp. 131–137.

[58] T. P. Hill and R. P. Kertz. „Comparisons of Stop Rule and Supremum Expectations of i.i.d. Random Variables". In: *The Annals of Probability* 10.2 (1982), pp. 336–345.

[59] T. P. Hill and R. P. Kertz. „A survey of prophet inequalities in optimal stopping theory". In: *Contemporary Mathematics* 125 (1992), pp. 191–207.

[60] D. S. Hochbaum and R. Shamir. „Strongly Polynomial Algorithms for the High Multiplicity Scheduling Problem". In: *Operations Research* 39.4 (1991), pp. 648–653.

[61] D. S. Hochbaum and D. B. Shmoys. „Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results". In: *Journal of the ACM* 34.1 (1987), pp. 144–162.

[62] D. S. Hochbaum and D. B. Shmoys. „A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach". In: *SIAM Journal of Computing* 17.3 (1988), pp. 539–551.

[63] S. Ieong, R. McGrew, E. Nudelman, Y. Shoham, and Q. Sun. „Fast and Compact: A Simple Class of Congestion Games". In: *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*. AAAI'05. Pittsburgh, Pennsylvania: AAAI Press, 2005, pp. 489–494.

[64] R. Impagliazzo, R. Paturi, and F. Zane. „Which Problems Have Strongly Exponential Complexity?" In: *Journal of Computer and System Sciences* 63.4 (2001), pp. 512–530.

[65] K. Jansen. „An EPTAS for Scheduling Jobs on Uniform Processors: Using an MILP Relaxation with a Constant Number of Integral Variables". In: *SIAM Journal on Discrete Mathematics* 24.2 (2010), pp. 457–485.

[66] R. Kannan. „Minkowski's Convex Body Theorem and Integer Programming". In: *Mathematics of Operations Research* 12 (1987), pp. 415–440.

[67] N. Karmarkar and R. M. Karp. „An Efficient Approximation Scheme for the Onedimensional Bin-Packing Problem". In: *23rd Annual Symposium on Foundations of Computer Science*. FOCS '82. 1982, pp. 312–320.

[68] R. M. Karp. „A Characterization of the Minimum Cycle Mean in a Digraph". In: *Discrete Mathematics* 23.3 (1978), pp. 309 –311.

[69] B. de Keijzer and G. Schäfer. „Finding Social Optima in Congestion Games with Positive Externalities". In: *Proceedings of the 20th Annual European Symposium on Algorithms*. ESA '12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 395–406.

[70] D. P. Kennedy. „Prophet-type inequalities for multi-choice optimal stopping". In: *Stochastic Processes and their Applications* 24 (1987), pp. 77–88.

[71] R. P. Kertz. „Stop Rule and Supremum Expectations of i.i.d. Random Variables: A Complete Comparison by Conjugate Duality". In: *Journal of Multivariate Analysis* 19 (1986), pp. 88–112.

[72] O. Kharif. „Supermarkets Offer Personalized Pricing". In: *Bloomberg* (Nov. 2013).

[73] U. Krengel and L. Sucheston. „Semiamarts and Finite Values". In: *Bulletin of the American Mathemical Society* 83 (1977), pp. 745–747.

[74] U. Krengel and L. Sucheston. „On Semiamarts, Amarts, and Processes with Finite Value". In: *Advances in Probability* 4 (1978), pp. 197–266.

[75] H. W. Lenstra. „Integer Programming with a Fixed Number of Variables". In: *Mathematics of Operations Research* 8.4 (1983), pp. 538–548.

[76] E. Lindelöf. „Sur l'Application de la Méthode des Approximations Successives aux Équations Différentielles Ordinaires du Premier Ordre". In: *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences* 116 (1894), pp. 454–457.

[77] B. Lucier. „An economic view of prophet inequalities". In: *ACM SIGecom Exchanges* 16.1 (2017), pp. 24–47.

[78] J. G. Madigan. „Scheduling a Multi-Product Single Machine System for an Infinite Planning Period". In: *Management Science* 14.11 (1968), pp. 713–719.

[79] D. Mattioli. „On Orbitz, Mac Users Steered to Pricier Hotels". In: *The Wall Street Journal* (Aug. 2012).

[80] C. A. Meyers and A. S. Schulz. „The Complexity of Welfare Maximization in Congestion Games". In: *Networks* 59.2 (2012), pp. 252–260.

[81] A. Meyerson, A. Roytman, and B. Tagiku. „Online Multidimensional Load Balancing". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*. 2013, pp. 287–302.

[82] I. Milchtaich. „Congestion Games with Player-Specific Payoff Functions". In: *Games and Economic Behavior* 13.1 (1996), pp. 111–124.

[83] R. B. Myerson. „Optimal Auction Design". In: *Mathematics of Operations Research* 6.1 (1981), pp. 58–73.

[84] M. A. Narro Lopez and B. G. Kingsman. „The Economic Lot Scheduling Problem: Theory and Practice". In: *International Journal of Production Economics* 23.1-3 (1991), pp. 147–164.

[85] E. Petrank. „The Hardness of Approximation: Gap Location". In: *Computational Complexity* 4 (1994), pp. 133–157.

[86] R. Rosenthal. „A Class of Games Possessing Pure-Strategy Nash Equilibria". In: *International Journal of Game Theory* 2.1 (1973), pp. 65–67.

[87] M. Rothkopf. „The Traveling Salesman Problem: On the Reduction of Certain Large Problems to Smaller Ones". In: *Operations Research* 14.3 (1966), pp. 532–533.

[88] T. Rothvoß. „Approximating Bin Packing within O(log OPT * Log Log OPT) Bins". In: *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. 2013, pp. 20–29.

[89] T. Roughgarden. „Barriers to Near-Optimal Equilibria". In: *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. FOCS '14. Washington, DC, USA: IEEE Computer Society, 2014, pp. 71–80.

[90] U. Saint-Mont. „A Simple Derivation of a Complicated Prophet Region". In: *Journal of Multivariate Analysis* 80 (2002), pp. 67–72.

[91] E. Samuel-Cahn. „Comparisons of Threshold Stop Rule and Maximum for Independent Nonnegative Random Variables". In: *The Annals of Probability* 12.4 (1984), pp. 1213–1216.

[92] E. Samuel-Cahn. „Prophet inequalities for bounded negatively dependent random variables". In: *Statistics and Probability Letters* 12 (1991), pp. 213–216.

[93] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Berlin, Germany: Springer, 2003.

[94] C. Shapiro and H. R. Varian. *Information Rules: A Strategic Guide to the Network Economy*. Cambridge, MA: Harvard Business School Press, 1999.

[95] A. Skopalik and B. Vöcking. „Inapproximability of Pure Nash Equilibria". In: *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*. STOC '08. 2008, pp. 355–364.

[96] H. Sperber. „How to Find Nash Equilibria with Extreme Total Latency in Network Congestion Games?" In: *Mathematical Methods of Operations Research* 71.2 (2010), pp. 245–265.

[97] V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[98] R. Werneck, J. Setubal, and A. da Conceicão. „Finding Minimum Congestion Spanning Trees". In: *Journal of Experimental Algorithmics* 5, 11 (2000).

[99] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. 1st. New York: Cambridge University Press, 2011.

[100] L. A. Wolsey. „An Analysis of the Greedy Algorithm for the Submodular Set Covering Problem". In: *Combinatorica* 2.4 (1982), pp. 385–393.

[101] Q. Yan. „Mechanism Design via Correlation Gap". In: *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete algorithms*. SODA '11. ACM-SIAM, 2011, pp. 710–719.

NEDERLANDSE SAMENVATTING

In dit proefschrift worden vier optimalisatieproblemen bekeken die voortkomen uit de onderzoeksgebieden van algoritmische speltheorie, planning en mechanisme-ontwerptheorie.

Voor elk van de vier problemen laten we zien dat het waarschijnlijk niet mogelijk is om efficiënt een optimale oplossing hiervoor te vinden. Om iets nauwkeuriger te zijn bewijzen we, onder bepaalde aannames die een groot deel van de wetenschappers waarschijnlijk acht, maar nog niemand aan heeft kunnen tonen, dat een optimale oplossing voor deze problemen niet gevonden kan worden binnen een bepaalde tijd. Derhalve wordt er vaak gekeken naar de complexiteit van het vinden van oplossingen die wellicht niet optimaal zijn, maar wel in korte tijd berekend kunnen worden. Methoden die dergelijke bijna-optimale oplossingen trachten te vinden, heten *benaderingsalgoritmen*. Voor elk van de vier problemen in dit proefschrift vinden wij een benaderingsalgoritme die (bijna) net zo goed of zo snel is als waar we theoretisch gezien op kunnen hopen.

Bij problemen in de tak van speltheorie wordt rekening gehouden met het natuurlijke gedrag van mensen. Beschouw ter illustratie een verkeersnetwerk waar mensen zich in bevinden. Iedere chauffeur kiest een route waarvan hij of zij denkt dat deze de beste route voor hem of haar is, bijvoorbeeld de kortste route qua verwachte tijd. Een stabiele situatie waarin iedere speler de keuze maakt die voor hem of haar het beste is, gegeven de andere weggebruikers, noemen we een *Nash-evenwicht*. Een dergelijk Nash-evenwicht functioneert in theoretisch onderzoek als indicator voor situaties die in de praktijk voorkomen. Door deze egoïstische keuzes van mensen zijn de totale kosten van het systeem hoger dan noodzakelijk; als we mensen centraal aan zouden kunnen sturen hoe ze zich door het netwerk dienen te begeven, zouden we kunnen werken naar een situatie waarin bijvoorbeeld de totale reistijd of het totale brandstofverbruik lager is, of zelfs minimaal.

In Hoofdstuk 2 bekijken we een benaderingsalgoritme om deze totale kosten zo laag mogelijk te houden als het netwerk aan bepaalde voorwaarden voldoet. We beschouwen daar een opstoppingsmodel voor netwerken waarin de kosten voor iedere connectie omhoog gaan naarmate er zich

meer gebruikers op die connectie bevinden, net zoals in een verkeersmodel. Specifiek wordt er gekeken naar netwerken waarin iedere gebruiker een strategie moet kiezen die beschreven kan worden door middel van een polymatroïde, wat een abstracte wiskundige structuur is met vele toepassingen. Op deze manier kunnen we verscheidene praktijkproblemen die ogenschijnlijk weinig met elkaar te maken hebben, in één keer behandelen. We bewijzen dat het waarschijnlijk niet mogelijk is om efficiënt een oplossing te vinden die maximaal een bepaalde factor slechter is dan de optimale oplossing, en zetten een concrete methode uiteen die een oplossing van die best mogelijke kwaliteit vindt.

In het gebied van planning is de doelstelling in het algemeen om taken zo goed mogelijk op machines te plannen. Men kan bijvoorbeeld zoeken naar methoden die ervoor zorgen dat de taken zo snel mogelijk afgerond zijn, of zo goedkoop mogelijk worden uitgevoerd.

In Hoofdstuk 3 bekijken we een dergelijk planningsprobleem. We beschouwen het probleem waarin een aantal producten op één machine geproduceerd dient te worden. De machine kan slechts één product tegelijkertijd verwerken. Van elk product is bekend hoeveel er per tijdseenheid geproduceerd kan worden, hoeveel vraag er naar het product is per tijdseenheid en hoeveel het per tijdseenheid kost om één eenheid op te slaan in de voorraad. Naast deze voorraadkosten dienen er overstapkosten te worden betaald wanneer de machine overschakelt van de productie van het ene product naar het andere product. Het doel is om een zo goedkoop mogelijke cyclische planning te maken waarin op elk moment aan de vraag van alle producten voldaan wordt. We beschouwen bovendien drie mogelijke varianten, afhankelijk van de vraag of de machine altijd op volle snelheid moet produceren of de productiesnelheid verlaagd kan worden, en van de vraag of de machine op elk moment kan overschakelen van productie of alleen aan het einde van een tijdseenheid (bijvoorbeeld alleen 's nachts). Dit probleem is op grond van twee redenen extra moeilijk. Niet alleen leiden de overstapkosten tot een verhoogde complexiteit, bovendien is het mogelijk om een instantie van dit probleem zeer bondig te formuleren, wat tot gevolg heeft dat efficiënte algoritmen significant minder tijd tot hun beschikking hebben om een oplossing te vinden. Dit laatste staat bekend als *hoge-multipliciteitscodering*.

Ook voor dit probleem bewijzen we dat efficiënt een optimale oplossing vinden waarschijnlijk onmogelijk is. Daarentegen karakteriseren we optimale oplossingen grotendeels door een aantal structurele eigenschappen

van optimale oplossingen aan te tonen, en vinden we optimale oplossingen voor situaties waarin het aantal producten laag is. Aangezien dit laatste het voornaamste geval is bij situaties waarin dit probleem zich in de praktijk manifesteert, is dit een redelijke aanname. Vervolgens vinden we voor twee varianten met een algemeen aantal producten een benaderingsalgoritme. Voor één variant kan een oplossing worden gevonden met een kwaliteit die arbitrair dicht bij de optimale waarde ligt, voor de andere variant is de kloof niet arbitrair klein.

Hoofdstuk 4 behandelt ook een planningsprobleem, weliswaar van een heel andere aard. In dit planningsprobleem zijn er een aantal taken gegeven, en elke taak heeft bepaalde eigenschappen. Men zou bijvoorbeeld kunnen denken aan taken die gepland moeten worden op computers, en elke taak heeft vereisten aan het CPU-gebruik, het RAM-gebruik et cetera. Het doel is om deze verzameling taken te verdelen over een reeks computers, zodanig dat geen enkele computer in enig aspect overbelast raakt.

Voor dit probleem geven we, onder bepaalde waarschijnlijk geachte theoretische aannames, een harde ondergrens aan de rekentijd die een benaderingsalgoritme nodig heeft om een oplossing te geven die arbitrair dicht bij de optimale oplossing ligt: deze tijd is namelijk minimaal dubbel exponentieel. Bovendien tonen we aan dat de toevoeging van een klein aantal computers deze minimaal benodigde rekentijd niet significant vermindert. Dit staat in schril contrast met een flink aantal optimalisatieproblemen waarbij een vermeerdering van de bruikbare bronnen wel een beduidende reductie in rekentijd of stijging in kwaliteit teweegbrengt. Ten slotte geven we een benaderingsalgoritme die een rekentijd behoeft die nagenoeg gelijk is aan de rekentijd van de dubbel exponentiële ondergrens, wat derhalve een vrijwel optimale rekentijd bedraagt.

Ten slotte bekijkt Hoofdstuk 5 twee schijnbaar ongerelateerde problemen, waarvoor met dezelfde technieken corresponderende resultaten kunnen worden geboekt. Het eerste probleem komt uit de theorie van optimaal stoppen, waarin aan een gokker één voor één een realisatie van stochastische variabelen wordt gepresenteerd. Ter illustratie, neem een P&O-manager die een nieuwe werknemer aan wil nemen. Voorafgaand aan elk sollicitatiegesprek heeft de P&O-manager wel een idee van de kwaliteit van de kandidaat, maar diens daadwerkelijke kwaliteit openbaart zich pas tijdens het sollicitatiegesprek. Direct daarna dient de P&O-manager de betreffende kandidaat aan te nemen, of onherroepelijk af te wijzen. Welke strategie dient hij of zij te gebruiken in de hoop iemand met een zo hoog

mogelijke kwaliteit in dienst te nemen? Hierin onderscheiden we een niet-adaptieve en een adaptieve strategie. In de eerste wordt a priori een drempelwaarde voor elke kandidaat geselecteerd en hij of zij wordt aangenomen indien de gebleken kwaliteit deze drempelwaarde overschrijdt. In de adaptieve strategie kan de drempelwaarde voor elke kandidaat worden aangepast op basis van welke kandidaten er voor hem of haar reeds afgewezen zijn.

Het tweede probleem komt uit de mechanisme-ontwerptheorie. Stel dat je één object wilt verkopen aan een koper uit een groep potentiële consumenten met als doel je omzet te maximaliseren. Je kunt gebruikmaken van een niet-adaptieve strategie door iedere potentiële consument een mail te sturen met een persoonlijke prijsaanbieding, en de eerste consument die zijn of haar persoonlijke prijs accepteert, koopt je object voor de geboden prijs. Anderzijds kun je gebruikmaken van een adaptieve strategie waarbij je de potentiële consumenten iteratief benadert en de prijs die je iemand aanbiedt mag afhangen van welke consumenten hun aanbod daarvoor al afgewezen hebben.

Voor beide problemen bewijzen we dat er een niet-adaptieve strategie bestaat die een kwaliteit of omzet kan garanderen van ten minste $1 - 1/e \approx 63\%$ van de best mogelijke kwaliteit of omzet. Als we daarentegen meer kracht geven aan onze strategie door deze gebruik te laten maken van adaptiviteit, loopt deze garantie op tot ongeveer 74.5% van de optimale kwaliteit of omzet. Binnen beide contexten is de gevonden strategie bovendien het best haalbare: we tonen aan dat het met de gegeven mogelijkheden van de strategieën onmogelijk is om een betere garantie te geven.

VALORISATION

This chapter discusses the contribution of the research of this thesis to society. The first part outlines the contributions of the general fields, and the second part focuses more on the actual contents of the chapters.

This thesis deals with different optimization problems, in the fields of congestion models, high multiplicity scheduling, vector scheduling, and optimal stopping theory and posted price mechanisms. In every optimization problem, the goal is to minimize costs, maximize profit or to solve something as quickly as possible. The optimization of timetables for public transport, vehicle routing problems, scheduling problems and other applications has provided major efficiency improvements. Timetables are optimized to reduce travelling time; vehicle routes are optimized to reduce time, fuel costs and the impact on the environment; scheduling jobs on machines in factories or tasks in processes is optimized to improve efficiency and so on. The influence of mathematical optimization to society has been huge.

Although most optimization problems occurring in practice are intrinsically hard to solve to optimality, the impact of implementing non-optimal solutions is still huge. Approximation algorithms compute solutions whose value are not too far from the optimum. Though suboptimal, they can still provide a huge decrease in costs, time requirements, or fuel costs, having a positive impact on the users and the environment.

Chapter 2 deals with congestion models. Since there is a lot of congestion on road and data networks, this is a field with relevant applications to practice. By understanding where congestion comes from and the dynamics underlying it, congestion can be reduced to save costs, time and the negative impact on the environment. For example, drivers arrive sooner and cheaper at their destination, thereby using less fuel which is good for the environment. And by reducing congestion in internet networks, data packets arrive faster.

Chapter 3 and Chapter 4 are problems in the field of scheduling. This field is concerned with scheduling jobs on machines in general, but its applications are more varied and range from scheduling exams to scheduling tasks on assembly lines. Companies use techniques from this field to optimize their processes, educational institutes use scheduling methods to find timetables, timetables for public transport are constructed using schedul-

ing, and so on. In all applications, time or costs are saved by optimizing the efficiency.

Finally, Chapter 5 considers problems in optimal stopping theory and posted price mechanisms, in which the objective is to maximize the expected reward or profit. Both fields are concerned with optimizing expected values given a random arriving sequence of events, such as applicants for a vacant position or customers for an item someone is selling. In particular, posted price mechanisms can be used in auctions and by companies who want to set personalized prices for their customers, in order to maximize their profit.

The remainder of the valorisation will outline more details about the contributions of the results in the different chapters.

Chapter 2. Polymatroid Congestion Models

The problem considered in Chapter 2 is to minimize a non-decreasing separable function over a polymatroid. A polymatroid is an abstract structure that incorporates many different structures like singletons, spanning trees in graphs and matroids. The results in this chapter show that efficiently finding an optimal solution in such a system is hard, i.e. there is an intrinsic hardness in the problem that hinders researchers to find an optimal solution. To complement this hardness result, the chapter provides an algorithm that finds the best possible solution, up to a constant factor, one can find in an efficient amount of time. Researchers in discrete optimization can use the method in this chapter whenever they need to minimize a function over a polymatroid to find a satisfactory solution.

Minimizing a function over such a general structure subject to common constraints has a wide variety of applications in different theoretical topics. This of course includes the perspective from which the chapter is written, namely polymatroid congestion models. Here, the strategic choices of a set of players induce a strategy profile whose social costs we are minimizing. Researchers working in congestion models can draw inspiration from the methods and analysis leading to the result of this chapter. Moreover, minimum cost solutions can serve as building blocks for other cost-efficient solutions in related problems.

In practice, the results of this chapter are mainly relevant in two different scenarios. The first one is in cooperative games, where players can collaborate to minimize their costs. Collaborations between different parties is happening more and more. Whenever the strategy space of every player

can be modelled as a polymatroid, the result of this chapter can be applied. This is for example true when players need to connect (a subset of) different objects in a network, e.g. computers, servers, or physical locations. Using the algorithm of this chapter results in a lower total cost, saving e.g. money, time or fuel, which has a positive impact on the players and the environment.

The second scenario is the situation in which a planner can implement a solution all players adhere to. This can occur e.g. in internet protocols or when autonomous cars make use of the road network. In both applications, there is a central planner that implements some protocol or software that tries to find a satisfactory solution for everyone. Minimizing costs can result in lower electricity usage, costs, time usage and fuel consumption.

Chapter 3. High Multiplicity Scheduling

Chapter 3 is dedicated to scheduling a set of products on a single machine that can produce one product at a time. Every product is associated with a maximum production rate, a demand rate and holding costs per time unit. Moreover, there are sequencing costs that need to be paid when the machine switches production. The objective is to find a cyclic schedule that minimizes the average costs.

The research in this chapter was inspired by a problem in practice. A multinational textile company wished to find the optimal cycle length for their production, of only three types of lycra in extremely large quantities on a single machine. The results in this chapter directly relate to problems like this, where a small number of products need to be scheduled on a machine. For a small number of products, the chapter gives the best possible schedule. For a general number of products, we show the problem is hard to solve efficiently and present an algorithm that finds a solution whose costs are not too far off from the costs of the optimal solution. All companies with similar problems can apply the methods in this chapter to find a solution to their problem.

This is the first research that tackles the problem in this form. By including both the high multiplicity encoding of the input and the sequencing costs, the problem considered in the chapter is closer to problems occurring in reality. Moreover, this research is a starting point for research in more complex problems and the ideas and results can help other researchers and practitioners to find methods to solve problems with additional practical constraints.

Chapter 4. Vector Scheduling

In Chapter 4 the Vector Scheduling problem is considered. There, a set of jobs with certain requirements need to be scheduled on a set of machines such that no machine is overloaded. One can think e.g. of applications on a computer that both require some amount of memory and have some CPU requirements, that need to be scheduled on a set of computers such that every computer can handle the workload.

The theoretical contribution is clear. We show almost matching upper and lower bounds on the running time required to find an approximate solution to the problem. Thus, it almost settles the theoretical complexity of the problem in terms of its approximation schemes.

In practice, the main contribution of this chapter is the knowledge that computing near-optimal solutions to this problem can take a rather long time. The lower bounds we show on the running time of these approximation schemes are high, indicating that no resources should be wasted on trying to find near-optimal solutions. The running time of the approximation scheme presented in the chapter, though nearly matching the provided lower bounds, is still too large for practical purposes.

There is a wide range of problems occurring in practice that can be modelled as an instance of the Vector Scheduling problem. The example above seems the most natural, in which the objective is to schedule a set of jobs on a set of computers, such that no computer is overloaded. For all these problems, and extensions of the problem, the results in this chapter show that finding near-optimal solutions to this problem is simply too hard to do efficiently.

Chapter 5: Optimal Stopping and Posted Prices

Finally, Chapter 5 concerns itself with two similar problems in different fields. In optimal stopping theory, a gambler faces a finite sequence of non-negative random variables whose values become apparent upon arrival. At every arrival, the gambler can claim the realized value and stop, or reject the value and continue, with the goal of maximizing his expected reward. In posted price mechanisms, a seller has a single item to sell to a set of potential customers who arrive uniform at random. The seller can set individual prices for the customers and makes a profit equal to the price he set for the first customer whose valuation exceeded the price. In both fields, the chapter provides approximation algorithms for a non-adaptive and an adaptive version of both problems.

The problem in optimal stopping theory has applications in human resources management. Suppose a company needs to hire a new employee and has interviews with some applicants. During the interviews, they can assess the value of the applicant for the position. The algorithms from this chapter provide a way to decide which of the applicants to hire, with the objective of maximizing the value of the new employee.

This is true for any practical problem in which true values are learned upon the arrivals, and the goal is to maximize the value of the first arrival that is accepted.

The applications following from the second field of posted price mechanisms are quite natural. Whenever some person or company needs to sell one item and wants to make the highest revenue, they can apply the algorithms in this chapter to do so. An example of the non-adaptive scenario is a retailer that sends a direct mail to all its potential customers, where the first customer to accept the price offered to her gets to buy the item. An example of the adaptive scenario is the situation in which at the gate of an airport, the employees sequentially offer travellers an upgrade to business class, where the extra revenue they make equals the price offered to the first accepting traveller.

CURRICULUM VITAE

Tim Oosterwijk was born in Nijmegen, the Netherlands on October 24, 1989. In 2007, he received his Gymnasium diploma from the Graaf Huyn College in Geleen. In September of the same year he started studying Industrial and Applied Mathematics with a focus on Discrete Mathematics and Applications at Eindhoven University of Technology. Under supervision of prof. dr. Nikhil Bansal, he received his Master's degree cum laude in October 2013.

From September 2013 until January 2017 he was a PhD student at Maastricht University under supervision of prof. dr. Rudolf Müller and dr. Tjark Vredeveld. Parts of the results of his research are presented in this thesis. He presented his work at various international conferences and all chapters of this thesis are published or under review in international academic journals.

From February 2017 until February 2018, he was a lecturer at Maastricht University. In March 2018, he will start a combined post-doctoral fellowship at the Universidad de Chile in Santiago, Chile and at the Max Planck Institut für Informatik in Saarbrücken, Germany.