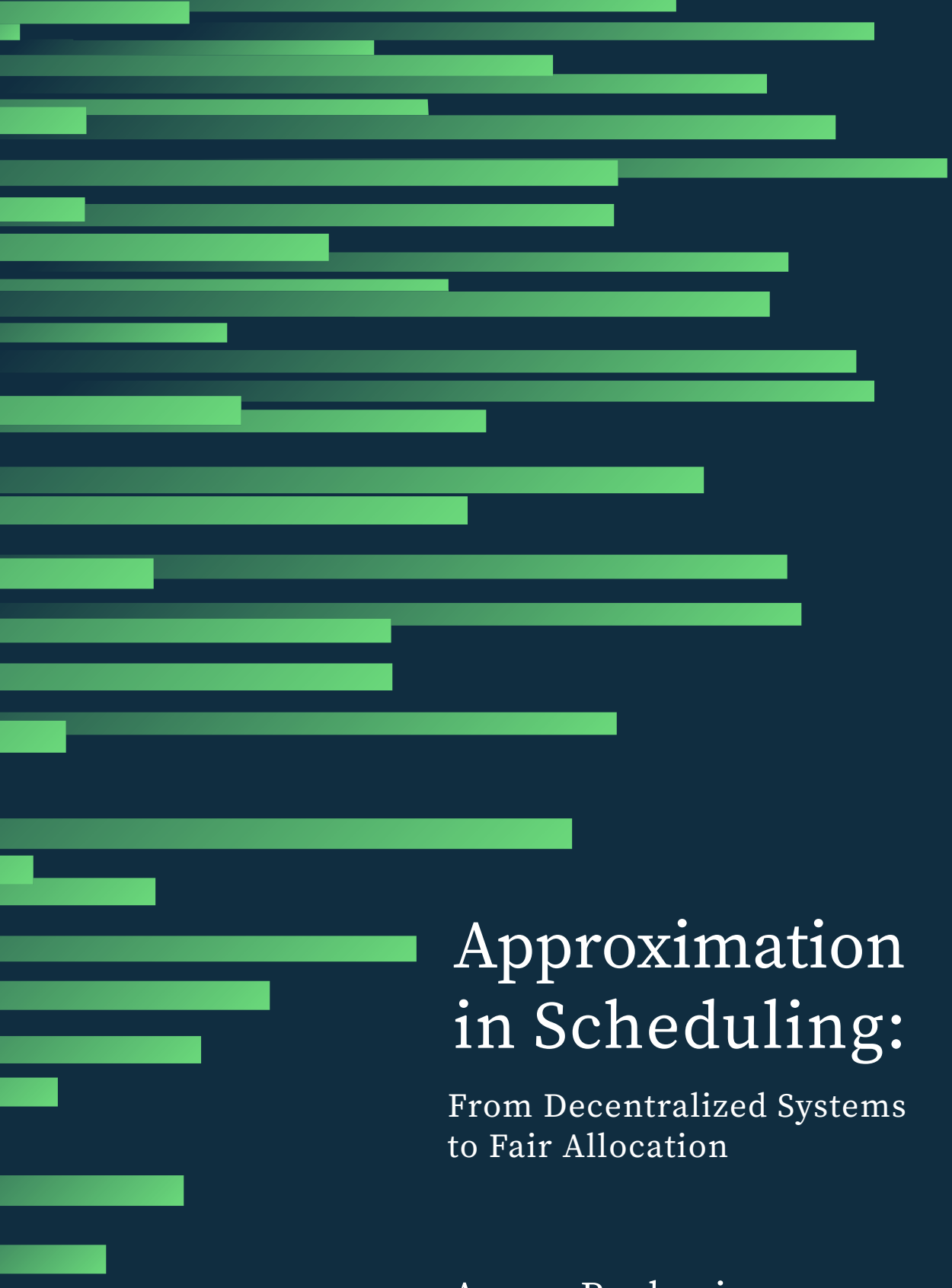


Approximation in Scheduling:

From Decentralized Systems to Fair Allocation

Arman Rouhani

Propositions (stellingen) accompanying the PhD-thesis

Approximation in Scheduling: From Decentralized Systems to Fair Allocation

Arman Rouhani

1. When you tackle a hard problem, it sometimes helps to relax first. (Chapter 2)
2. In decentralized scheduling, the selfish machine selection need not yield a globally optimal schedule for total completion time, even when every machine sequences its assigned jobs optimally for the same objective. (Chapter 2)
3. In fixed-order scheduling with deadlines, allowing a greedy algorithm to consider more open machines when assigning each job does not guarantee that it will use fewer machines. (Chapter 3)
4. More gifts do not necessarily make Christmas fairer. (Chapter 4)
5. A counterexample may end one conjecture and begin a better one.
6. A difficult part of optimization is deciding what to optimize.
7. A solution with an approximation guarantee, delivered on time, can be more useful than an optimal solution delivered too late.
8. Proving that an approximation guarantee cannot be improved can be as valuable as improving it.
9. One challenge of a PhD is recognizing that slow progress is still progress.
10. For international PhD candidates, career choices can be shaped not only by academic goals but also by immigration rules.
11. A well-optimized life is not necessarily a happy one.
12. The purpose of computing is insight, not numbers.

- Richard Hamming

Doctoral thesis

APPROXIMATION IN SCHEDULING: FROM DECENTRALIZED SYSTEMS TO FAIR ALLOCATION

Arman Rouhani

2026

APPROXIMATION IN SCHEDULING: FROM DECENTRALIZED SYSTEMS TO FAIR ALLOCATION

Dissertation

To obtain the degree of Doctor at Maastricht University,
on the authority of the Rector Magnificus, Prof. Dr. Jan Smits,
in accordance with the decision of the Board of Deans,
to be defended in public
on Friday the 9th of October 2026, at 10.00 hours

by

Arman Rouhani

Supervisor
Dr. André Berger

Co-supervisor
Dr. Marc Schröder

Assessment Committee
Prof. Dr. ir. Constantinus van Hoesel (chair), *Maastricht University*
Dr. Matus Mihalak, *Maastricht University*
Prof. Dr. Guido Schäfer, *CWI Amsterdam*
Prof. Dr. rer. nat. Laura Vargas Koch, *RWTH Aachen University*

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

© Arman Rouhani, Maastricht 2026.
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior written permission of the author.

This book has been typeset by the author using LaTeX.

Cover	Natalia Kubalski, 2026
Production	ProefschriftMaken || proefschriftmaken.nl
ISBN	978-94-6534-594-9

To my loved ones

Acknowledgments

When I applied for my PhD position, I could not have imagined how much the years ahead would change my life. It was during the COVID-19 pandemic, when daily life was already full of uncertainty, and planning for the future was difficult. Looking back, I am grateful to the person I was then for having the courage to take on the challenge. The years that followed gave me the chance to work on ideas that genuinely interested me, present my research at conferences, travel to new places, and experience different cultures. They also taught me how to live far from my family and friends and deal with many new responsibilities on my own. Research itself did not always move in a straight line. Some promising ideas led nowhere, while others grew slowly through patience, discussion, and many revisions. Throughout this journey, I was fortunate to be surrounded by people whose encouragement, guidance, support, and friendship made these years both possible and meaningful, and for that I am deeply grateful.

First, I would like to thank my supervisor. André, thank you for giving me the freedom to explore different topics and develop my own research ideas, while always being available when I needed advice or support. Your feedback was always clear, constructive, and to the point, and it helped me improve both my research and the way I present it. I am also grateful for how enjoyable and easy-going our collaboration was. Even during difficult moments, I never felt that I had to face the challenges alone. Your support, trust, and encouragement gave me the motivation and confidence to keep moving forward.

I am also deeply grateful to my co-supervisor, Marc. Thank you for always being there for me and for the many things I learned from you. You always seemed to have the perfect counterexample ready. Sometimes, after I had finally convinced myself that a proof worked, you needed less than a minute to disprove it with a counterexample. Although this was not always pleasant at the time, it taught me how to

think carefully about a problem. I always enjoyed our brainstorming sessions in your office, especially around what may be the only chalkboard in the department. I greatly admire the clear way you think about problems and the even clearer way you explain them to others. You have inspired me both as a researcher and as a teacher. I also enjoyed our padel games, where your competitive spirit was always easy to notice.

Lars, I am very grateful for the chance to work with you. I learned a great deal from you, especially about academic writing and about how methods from related work can lead to new research ideas. I also admire your positive and thoughtful approach to research. Beyond our work, I am especially grateful that you helped me move with your trailer. You and Faiza were always kind and welcoming. Thank you for the wonderful house party, the delicious meals, and all the moments we shared. I truly appreciate everything you both did for me.

I would also like to thank the members of my assessment committee, Constantinus van Hoesel, Guido Schäfer, Matus Mihalák, and Laura Vargas Koch, for carefully evaluating my thesis and for their thoughtful feedback and flexibility in arranging the defense. Stan, thank you also for inviting me to your home on Christmas Day during my first year and making me feel welcome. Guido, thank you for encouraging me to continue with my first research topic at the LNMB conference and for your excellent courses on Algorithmic Game Theory. Matus, I truly appreciate the time and care you put into reading my thesis and providing such detailed and constructive comments. Laura, I still remember the stretching exercise you led after lunch at the Computational Game Theory Day. It brought new energy to the room, and I sometimes use the same idea in my own tutorials.

I am also grateful to Tjark Vredeveld, Tom Demeulemeester, and Veerle Timmermans for agreeing to serve as members of the opposition. Tjark, thank you for your kindness and support throughout these years. Tom, thank you for your support, your kind words, and the many friendly conversations we had in the hallway. Veerle, I always enjoyed

our conversations between tutorials, especially when we discussed the courses and how they could be improved.

I am very happy to have Niloufar and Matthijs as my paranymphs. Niloufar, you have been kind and supportive from the first day we met in Maastricht. I have always enjoyed our conversations, your excellent taste in music, and the warmth you brought to every gathering. Sharing the same language and cultural background gave us plenty to talk and laugh about. I may have sometimes let you win our chess games until now, but those days are over, so prepare yourself. Matthijs, thank you for your positive and easy-going attitude and for the energy you brought to my PhD years. I always enjoyed our conversations between tutorials, whether we were discussing teaching or exchanging movie recommendations. Our coffee breaks and the football predictions you started made many working days more enjoyable.

I would also like to thank everyone at the Department of Quantitative Economics for creating such a friendly, supportive, and inspiring atmosphere. Being part of the department taught me a great deal, both as a researcher and as a teacher. A special thank you to Mandy and Dany for always being kind, approachable, and ready to help, and for making KE feel a little more like home.

My PhD years would not have been the same without the fellow PhD students and colleagues with whom I shared everyday life at KE. The lunches, coffee breaks, office conversations, and many small everyday moments made those years much more enjoyable. I am grateful for the support, laughter, and friendship we shared, both inside and outside the department.

János, I have always admired your wisdom, kindness, and humility, and I am grateful for how supportive you have been. I greatly valued our many conversations over the years. You are one of the most genuine people I have met.

Joep, thank you for always being willing to help when I had a question, and for all the coffee breaks, funny moments, and conversations we

shared. You inspired me to start playing padel, and I am grateful for your patience when I was still a beginner.

Anh, thank you for being such a kind friend and for always taking such good care of Tony whenever I was away at conferences. Judging by how much he likes you, you clearly hold a special place in his heart.

Francesco, I am grateful for all the moments we shared in Maastricht. Your sense of humor always brought warmth and laughter to our gatherings, and I will remember those days fondly.

François, thank you for all the memories we shared, from chess games in the staff lounge and bouldering to skiing at SnowWorld, the boat party, and our trip to Brussels. I will always remember those moments with a smile.

Ivan, I am grateful for our chess games and movie discussions, for encouraging me to go bouldering, and for being a genuine person.

I would also like to thank all the other PhD students and colleagues who were part of these years: Ashkan, Moritz, Daniel, Freija, Marie T., Shasha, Leo, Thomas, Lucas, Dewi, Doudou, Farzaneh, Mohsen, Shahrezad, Luke, Uğur, Halil, Tim, Enrico, Marie C., Pauline, Peter, Ruben, Son, Thijs, Aryan, Saleh, Mahdi, Navid, Bas and Anna. Thank you for the lunches, coffee breaks, conversations, and many small moments that made everyday PhD life more enjoyable. I am grateful for the memories we made together.

To my friends, within and outside academia, thank you for being part of my life. Amirreza, thank you for the friendly competition throughout our university years and for a friendship that has continued ever since. I am grateful for all the memories we have made in recent years, whether on trips together or during our phone calls.

I would also like to thank all my friends on the padel court, especially Thomas. Whenever we played padel together, I forgot about my research problems and simply enjoyed the moment. The games,

conversations, and friendly competition brightened many of my days and made the challenges of the PhD easier to handle.

Trine, I sometimes think that our paths crossed at exactly the right time. Your love, patience, and support have made these years much brighter, and I cannot imagine having gone through them without you. I am deeply grateful for everything we have shared and for the kindness you have always shown me. I am also proud to have seen you graduate and grow in so many ways. I look forward to everything that still lies ahead of us.

Finally, to my family, thank you for your unconditional love and support. My dear mother, you have always been my greatest teacher. I am grateful for your constant support and encouragement and for the values you taught me. You carried the responsibilities of both parents with remarkable love and strength. I could never have asked for a better mother, and words will never fully express how grateful I am to you. My dear brother, I am very proud of you. I have always admired the way you face difficult times and keep going, and I feel very lucky to have someone like you to rely on. To my father, I wish you were here with us today, so that we could celebrate together.

Arman Rouhani
Maastricht
August 31, 2026

Contents

Acknowledgments **vii**

1 Introduction **1**
1.1 Combinatorial Optimization 2
1.2 Computational Complexity Theory 4
1.3 Approximation Algorithms 6
1.3.1 Greedy Algorithms 8
1.3.2 Local Search 9
1.3.3 Dynamic Programming 9
1.3.4 Linear Programming Relaxation 10
1.4 Scheduling Problems 13
1.5 Outline of the Thesis 15
1.5.1 Measuring the Inefficiency of Decentralized Scheduling on Related Machines 15
1.5.2 Fixed Order Scheduling with Deadlines 17
1.5.3 Cost Preserving Dependent Rounding for Allocation Problems 19
1.6 Publications 21

2 Measuring the Inefficiency of Decentralized Scheduling on Related Machines **23**
2.1 Introduction 23
2.2 Preliminaries 27
2.2.1 Optimal Schedule for Jobs 29
2.2.2 Nash Equilibria and Price of Anarchy 32
2.3 Upper Bounds on PoA under the SPT Rule 37
2.3.1 Two Machines 39
2.3.2 Three or more Machines 44
2.4 A Lower Bound on PoA under the WSPT Rule 52
2.5 Concluding Remarks 56

3 Fixed Order Scheduling with Deadlines **61**
3.1 Introduction 61
3.2 Preliminaries 64
3.2.1 Optimal Schedule 66
3.3 Analysis of Approximation Factors 67
3.3.1 Unit Processing Times 69
3.3.2 Non-Increasing Slacks 75
3.3.3 Non-Decreasing Slacks 80
3.3.4 Non-Increasing Deadlines 83
3.3.5 Approximation Factors with Few Machines 86
3.3.6 Logarithmic Approximation for Arbitrary Orders 93
3.4 Concluding Remarks 95

4 Cost Preserving Dependent Rounding for Allocation Problems **99**
4.1 Introduction 99
4.2 Budgeted Dependent Rounding 107
4.2.1 Omitted Proofs of Dependent Rounding 114
4.2.2 Chernoff Bounds 122
4.3 Application to Budgeted Santa Claus Problem 126
4.3.1 Linear Programming Formulation 126
4.3.2 Technical Goals 128
4.3.3 Rounding of Big Resources 129
4.3.4 Rounding of Small Resources 132
4.3.5 Approximation Factor 136
4.4 Concluding Remarks 136

Bibliography **139**

Summary **153**

Valorisation **155**

Nederlandse Samenvatting **161**

Curriculum Vitae **163**

Chapter 1

Introduction

Operations Research (OR) is a field of applied mathematics that focuses on making better decisions using analytical methods. It uses mathematical tools such as mathematical modeling, algorithmic techniques, simulations, and statistics to aid decision-making. OR provides a structured way to model real-world systems, analyze their behavior, and optimize their performance. The origins of OR can be traced back to World War II, when scientists and military planners worked together to improve the efficiency of logistics, radar operations, and resource allocation. From its origins to its current data-driven discipline, OR has kept the same important goal: to support decision making through mathematical modeling and analytical methods that bridge theory and practical insight [104].

OR employs tools to model and solve complex real-life problems. The scarcity of resources is inherent to most of these problems, posing many challenges for decision-makers in allocating resources to activities over time. These problems are known as *scheduling* problems, where the objective is to assign a set of tasks to limited resources in a way that optimizes one or more criteria. Scheduling arises in a wide range of real-world applications. It ranges from job-shop manufacturing systems [89] to airline and railway crew scheduling [54, 60], and operating room scheduling in hospitals [25].

In this thesis, we investigate three different scheduling problems inspired by practical applications. This introductory chapter provides important concepts and notations underlying the results of this thesis. We begin in Section 1.1 with an overview of *combinatorial optimization*, which is followed by a discussion of the *efficiency* of algorithms. Section 1.2 introduces the basic ideas of *computational complexity theory*. In Section 1.3, we introduce the concept of *approximation algorithms* and outline several key techniques for their design and analysis. Subsequently, Section 1.4 formally defines *scheduling problems*. Finally, Section 1.5 provides an overview of the structure and main contributions of this dissertation.

1.1 Combinatorial Optimization

Combinatorial optimization is an important subfield of mathematical optimization with the additional requirement that the set of feasible solutions is either finite or countably infinite. The classical *one-dimensional Bin Packing Problem* (BPP) is a famous example of combinatorial optimization problems that asks for the minimum number of identical bins of capacity B needed to pack n items with sizes $s(a_1), \ldots, s(a_n) \leq B$. It was first motivated by memory–allocation studies by Johnson [69]. Coffman, Garey, and Johnson [32] give a comprehensive survey.

Definition 1.1 (Bin Packing Decision Problem; Garey and Johnson [52])**.** *An instance of the Bin Packing Problem consists of a finite set* $A = \{a_1, a_2, \ldots, a_n\}$ *of items, a size* $s(a_i) \in \mathbb{Z}^+$ *for each* $a_i \in A$*, a bin capacity* $B \in \mathbb{Z}^+$*, and a positive integer* K*.*

Question*: Is there a partition of* A *into* K *disjoint subsets* $A_1, A_2, \ldots, A_K$ *such that*

$$\sum_{a \in A_i} s(a) \leq B \quad \forall i \in \{1, \ldots, K\}.$$

As mentioned earlier, combinatorial optimization deals with searching for the best solution to an *optimization* problem among a finite (or

countably infinite) set of solutions. An optimization problem is defined by its collection of instances, as follows.

Definition 1.2. *An instance I of an optimization problem $\mathcal{P}$ consists of a set of feasible solutions $\mathcal{S}(I)$ and an objective function $\boldsymbol{f} : \mathcal{S}(I) \to \mathbb{R}$ that maps each feasible solution to a real value. The goal is to find a feasible solution $s^* \in \mathcal{S}(I)$ such that $\boldsymbol{f}(s^*) \geq \boldsymbol{f}(s)$ for all $s \in \mathcal{S}(I)$ when $\mathcal{P}$ is a maximization problem, or such that $\boldsymbol{f}(s^*) \leq \boldsymbol{f}(s)$ for all $s \in \mathcal{S}(I)$ when $\mathcal{P}$ is a minimization problem.*

To efficiently solve optimization problems, researchers develop *algorithms*, which are systematic procedures that provide clear sequential instructions to obtain an optimal solution, s^*, to any given instance of the problem. The *efficiency* of an algorithm can be measured by its *worst-case running time*, which denotes the time it takes for an algorithm to find the solution in the worst possible case. The time efficiency of an algorithm is quantified with respect to the *input size*. The input size $|I|$ of an instance I is the number of bits needed to encode the instance.

To find the optimal solution to some problems, fast algorithms are developed. For many problems like Problem 1.1, finding an optimal solution is computationally intractable. Optimization problems vary in terms of their difficulty, and understanding this variation is a central focus of *computational complexity theory* (see Section 1.2). For computationally hard problems, considerable research has focused on the design and analysis of *approximation algorithms* (see Section 1.3). These algorithms do not guarantee an optimal solution but aim to efficiently produce solutions that are "good" enough. Some approximation algorithms are developed from a theoretical perspective, offering worst-case guarantees on both solution quality and running time. Others are driven by practical performance: they tend to work well on real-world instances, but offer little to no formal guarantees regarding their accuracy or computational complexity. In this thesis, we focus on developing approximation algorithms for

scheduling problems and derive worst-case guarantees that are valid for adversarial or perturbed inputs.

1.2 Computational Complexity Theory

Computational complexity theory studies the classification of problems based on their difficulty. Some problems turn out to be *easier* to solve than others, and we can find a *polynomial-time algorithm* that optimally solves them. The running time of a polynomial-time algorithm is bounded by a polynomial in the size of the input.

Example 1.1 (Airport Gate Optimization)**.** *Consider an airport terminal that seeks to minimize the number of gates used. Each flight has a scheduled arrival time, departure time, and boarding duration (which depends on aircraft type and passenger load). For safety, coordination, and runway efficiency, flights follow a predetermined global service order set by air traffic control or operational policies (e.g., departure slots, runway constraints, fairness across airlines). This order must hold locally: if two flights share a gate, they must board in the same relative order as in the global sequence. This constraint ensures that the overall operational plan is not disrupted and that sequencing guarantees, such as prioritized take-off windows or ground crew rotations, are preserved. The input to the problem consists of a set of triples* (r_j, p_j, d_j)*, where each triple represents the release time, processing (boarding) time, and deadline (departure time) of flight* j*, respectively, together with a predetermined processing order of the flights. The objective is to assign flights to gates and schedule their boarding so that all time constraints are satisfied, the prescribed order is respected, and the total number of gates required is minimized.*

Consider the case where all flights share a common arrival time, and boarding the passengers takes the same amount of time for all flights. In Chapter 3 we show that the greedy first-fit algorithm finds the optimal schedule for this problem in polynomial time.

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

The computational complexity of optimization problems is typically studied through their associated *decision problems*. A *decision problem* is defined as a mathematical problem where each instance requires a "yes" or "no" answer to a clearly stated question about the input (see Problem 1.1). For example, the decision version of the BPP asks: Can all items be packed into at most K bins? The optimization version of BPP, however, seeks the minimum number of bins required to pack all items. A decision problem is said to belong to class P if there exists a polynomial-time algorithm for solving it (see Garey and Johnson [51]). Class P is a subclass of class *NP*, which includes all decision problems whose "yes" answer can be verified in polynomial time. While problems in class P can be solved in polynomial time, it is unknown whether every problem in *NP* can also be solved in polynomial time. Although it is widely conjectured that $P \neq NP$, this question remains one of the most important open problems in computer science. Class *NP-complete* includes the most difficult problems in *NP*, meaning that these problems can be shown to be as hard as any other problems in *NP*. Hence, $P = NP$ if a polynomial time algorithm can be found for one of the problems in *NP-complete*. While it is enough to show that a problem is in P by developing a polynomial time algorithm, to show that a problem is *NP-complete*, we rely on polynomial *reductions* which show that solving one problem efficiently would imply an efficient solution for another. A decision problem is said to be *NP-hard* if every problem in *NP* can be polynomially reduced to it, i.e., it is at least as hard as the hardest problems in *NP*. If, in addition, the problem itself belongs to *NP*, then it is called *NP-complete*. Consider the BPP (Problem 1.1). This problem belongs to the class of problems called *NP-hard* problems. The problems belonging to the class *NP-hard* are commonly believed not to be solvable in polynomial time. Formally, a problem is *NP-hard* if its decision version is *NP-complete*.

As mentioned in Definition 1.3, the running time of a polynomial-time algorithm is bounded by a polynomial in the size of the input. Formally, the *input size* $|I|$ of an instance I is the number of bits needed to encode the instance succinctly. An *NP-complete* problem is said to be *strongly NP-complete* if it remains *NP-complete* even when all numerical parameters in the input are bounded by a polynomial in the size of the input. The notion of *strong NP-completeness* is closely related to *pseudo-polynomial* algorithms. The running time of a pseudo-polynomial time algorithm is bounded by a polynomial in the numerical value of the input (e.g., the largest integer parameter in the instance), but not necessarily by a polynomial in the length of the input encoding, i.e., the number of bits required to represent the input. Unless $P = NP$, a *strongly NP-complete* problem does not admit a pseudo-polynomial time algorithm. Furthermore, an optimization problem is said to be *strongly NP-hard* if its corresponding decision problem is *strongly NP-complete*.

Example 1.2 (Airport Gate Problem)**.** *Consider a variant of the problem described in Example 1.1 where there is no predetermined fixed order for the flights, the arrival times are the same, and all flights have the same departure time. This variant is simply the BPP that is proved to be strongly NP-complete through reduction from the Partition Problem by Garey and Johnson [52].*

1.3 Approximation Algorithms

For many problems, polynomial-time exact algorithms have not been found, that is, they probably lie outside the class P. To deal with the intractability of these problems, it makes sense to pursue ideas other than finding an exact optimal algorithm. One idea is that instead of solving the general problem, a rather tractable special version (a special problem structure) is exploited. Another method is designing *heuristics*, which are the procedures that provide reasonable estimates but for which no proven guarantees exist. One can also relax the requirement

of finding an optimal solution, but with a goal to relax this as little as possible, and develop *approximation algorithms*. Here, the goal is to find a solution that closely approximates the optimal solution in terms of its value. We assume that there is an objective function mapping each feasible solution of an optimization problem to a nonnegative value, and that an optimal solution is one that either minimizes or maximizes this value. We then formally define an approximation algorithm in Definition 1.4.

Definition 1.4. *Let $\mathcal{P}$ be an optimization problem. If $\mathcal{P}$ is a minimization problem, an algorithm for $\mathcal{P}$ has an approximation guarantee $\alpha \geq 1$ if, for every instance of $\mathcal{P}$, it returns a feasible solution whose value is at most α times the optimal value of that instance. If $\mathcal{P}$ is a maximization problem, an algorithm for $\mathcal{P}$ has an approximation guarantee $\alpha \leq 1$ if, for every instance of $\mathcal{P}$, it returns a feasible solution whose value is at least α times the optimal value of that instance.*

The closer α is to 1, the closer the solution value of the approximation algorithm is to the optimal solution. In the literature, α is also often called the *approximation ratio* or *approximation factor* of the algorithm, and the algorithm is an α-approximation algorithm. For an extensive introduction to approximation algorithms, we refer to the books by Vazirani [106] or Williamson and Shmoys [110].

Example 1.3 (Airport Gate Problem)**.** *Consider the problem described in Example 1.1. In Chapter 3, using a dynamic programming approach, we provide an approximation algorithm with a factor of $O(\log n)$. Define the slack as the difference between the departure time and the boarding time of a flight. We also prove that the simple greedy first-fit algorithm has an approximation factor of 2 in four different classes of instances: if the fixed order aligns with slacks that are in a non-increasing or non-decreasing order, if the fixed order aligns with deadlines that are in a non-increasing order, and if the optimal solution uses at most 3 gates.*

Approximation algorithms trade off optimality for efficiency. They have become important for tackling NP-hard optimization tasks where exact algorithms are infeasible. In the remainder of this section, we introduce several major techniques for designing approximation algorithms.

1.3.1 Greedy Algorithms

Greedy algorithms build a solution iteratively, at each step making a locally optimal choice in the hope that the cumulative result will be near global-optimal. These approximation algorithms are designed with a simple idea: repeatedly make choices that get one closer to a feasible solution for the problem. In other words, these algorithms try to be as opportunistic as possible.

Greedy strategies are often simple and fast, and for many problems they yield good approximation guarantees. A classical example is the Set Cover problem. The set cover problem is an optimization problem that seeks to find the smallest collection of subsets from a given set of subsets that, when combined, includes all elements of the original set. It involves a "universe" of elements and a collection of subsets of that universe, with the goal of identifying the fewest subsets needed to "cover" all elements in the universe. The greedy algorithm for this problem picks the set that covers the largest number of uncovered elements at each iteration. This heuristic is guaranteed to produce a set cover whose size is at most $1 + \frac{1}{2} + \cdots + \frac{1}{n} = H_n$ times the minimum possible, where n is the size of the universe. Here, H_n is the n-th *harmonic number*. In other words, greedy set cover is an $O(\log n)$-approximation algorithm [106]. Greedy algorithms also achieve the optimal approximation ratio for a variety of other NP-hard problems. For instance, a simple greedy strategy for Vertex Cover achieves a factor 2 approximation [106]. While greedy methods are not universally applicable, they often provide a natural starting point and sometimes can be coupled with more sophisticated analysis (e.g., primal-dual analysis) to establish performance guarantees.

1.3.2 Local Search

Local search is a set of methods that iteratively search through the set of solutions. It starts from an arbitrary feasible solution and then iteratively tries to increase solution quality by "moves" (localized modifications). At each step, the local search procedure moves from one feasible solution to another in the *neighborhood* (typically defined by exchanging or altering a limited number of solution components) until some stopping criteria are met. At this point, a *locally optimal* solution is reached, which is a solution that cannot be improved by any allowed local move. Although a local optimum need not be globally optimum, one can often prove that it is within some factor of the global optimum. The key is to bound the *locality gap*, i.e., the worst-case ratio between the value of a local optimum and a true optimum. The choice of a suitable neighborhood function has an important influence on the performance of the local search.

As an example, consider the metric k-MEDIAN clustering problem, which is NP-hard. A known approximation algorithm for k-Median uses local search with swap-based moves: at each iteration, it tries to swap one of the currently chosen centers with a center not in the solution if the swap yields a better clustering cost. It has been shown that allowing swaps of up to 2 facilities at a time yields a constant-factor approximation for k-Median. In particular, the natural 2-swaps local search algorithm for metric k-Median finds a solution provably within a factor 4 of the optimum [57].

By carefully designing neighborhoods (e.g., allowing multi-element swaps), one can balance the trade-off between solution quality and runtime, achieving good approximation ratios while keeping the local search polynomial time.

1.3.3 Dynamic Programming

It is often the case in algorithmic design that, when optimal solutions to small pieces of an instance are known, those solutions can be merged in a way that yields an optimal solution for the entire instance. This

is exactly the idea behind dynamic programming: the pieces of the problem are solved, and then their solutions are saved in order to solve the entire problem later on. As an example, consider the Unsplittable Flow Problem on a Path (UFPP) where we are given a capacitated path P and n tasks, each task having a demand, a profit, and start and end vertices. The goal is to compute a maximum profit set of tasks, such that for each edge e of P, the total demand of selected tasks that use e does not exceed the capacity of e. A dynamic programming approach results in a constant approximation factor for UFPP [18]. Dynamic programming can also be applied to the BPP (Definition 1.1). Although BPP is NP-hard, it admits a pseudo-polynomial time dynamic programming algorithm, closely related to the classical knapsack formulation [34, 79].

1.3.4 Linear Programming Relaxation

Linear programming is a fundamental optimization technique for modeling and solving optimization problems with linear objective functions and linear constraints. A *linear program (LP)* is an optimization problem in which one maximizes or minimizes a linear objective function subject to linear equalities and inequalities in real variables. If all variables are restricted to integer values (often $\{0, 1\}$), we obtain an *integer linear program (ILP)*. Whereas ILPs are generally *NP-hard*, LPs can be solved in polynomial time using algorithms such as the *ellipsoid* method [72] and *interior-point* methods [70]. Although the *simplex* method [41] is not polynomial in the worst case, it is widely used in practice due to its excellent empirical performance. For certain structured LPs, specialized combinatorial algorithms that are strongly polynomial are also known [105]. Linear programming plays an important role in the design of approximation algorithms for many combinatorial optimization problems (see [87, 100]).

One of the most general and successful paradigms for designing approximation algorithms is the use of *LP relaxations*. Many combinatorial optimization problems admit natural ILP formulations. In the LP relaxation approach, we formulate the optimization problem

as an ILP and then relax the integrality constraints to allow fractional values. Solving the resulting LP gives an optimal fractional solution, which can be found in polynomial time. The LP optimum provides a lower bound for minimization problems or an upper bound for maximization problems on the value of an optimal integer solution, since the relaxation enlarges the feasible region. The fractional solution itself may not correspond to any valid integer solution of the original problem (e.g., it might use half of a gate in Example 1.1, or schedule a fraction of a task in a scheduling problem). The main challenge here is to convert the fractional LP solution into an integral solution, while controlling the loss in solution quality. This step is known as *rounding*. Typically, the rounding step can be done either *deterministically* or *probabilistically*.

A simple deterministic rounding strategy is to set a threshold and round all fractional values based on it. Such methods work for certain problems (e.g., rounding $x_i \geq 0.5$ to 1 in the LP solution for the vertex cover problem yields a 2-approximation [106]). In many optimization settings, deterministic rounding rules can be too coarse to preserve feasibility or objective quality. A more flexible approach is *randomized rounding*, introduced by Raghavan and Thompson [90], which converts a fractional LP solution into an integral one by sampling according to carefully defined probability distributions that maintain the LP's structural properties in expectation [90, 110]. This allows one to exploit concentration inequalities such as *Chernoff–Hoeffding* bounds (see Section 4.2.2) to show that the resulting integral solution achieves near-optimal performance with high probability [45].

The LP relaxation and rounding paradigms have become central design tools for developing approximation algorithms for scheduling, routing, facility location, and many other problems [110].

Independent vs. Dependent Randomized Rounding

In independent randomized rounding, each decision variable is rounded up or down independently according to its fractional value.

Formally, let $x_j \in [0,1]$ denote the fractional value of a binary decision variable in the LP solution. We define a binary random variable X_j that represents the rounded value of this decision. Independent randomized rounding sets $\mathbb{P}[X_j = 1] = x_j$ and $\mathbb{P}[X_j = 0] = 1 - x_j$, independently for each j. This approach is simple and has marginal preservation. By linearity of expectation, the expected cost or profit of the rounded solution equals the value of the fractional solution, and one can often argue that no constraint is violated in expectation. Moreover, independence allows powerful tail-bound analyses. If one needs to bound the probability that some sum of selected variables deviates from its expectation, standard Chernoff–Hoeffding bounds can be applied due to the independence assumption. Independent rounding has been applied in numerous approximation algorithms (e.g., randomized rounding algorithms for network design and packing problems [90]). Independent randomized rounding is suitable when feasibility is defined by a few loose inequalities and one only needs to preserve expectations, but it is generally inappropriate when the rounded solution must satisfy strict global structure (e.g., spanning trees, perfect matchings, matroid bases) or hard equalities such as fixed cardinalities, per-vertex degree constraints, or capacity/budget constraints. In these cases, rounding the variables independently produces, with non-negligible probability, solutions that violate the required constraints, and any subsequent repair step typically worsens the approximation guarantee.

Consequently, dependent rounding introduces controlled correlations among the rounding decisions in order to enforce specified global linear constraints while preserving the marginals. For example, given a fractional vector $x \in [0,1]^d$ and a family $\mathcal{F}$ of linear constraints of the form $\sum_{i \in S} X_i \in \{\lfloor \sum_{i \in S} x_i \rfloor, \lceil \sum_{i \in S} x_i \rceil\}$ (e.g., fixed cardinalities, per-vertex degree bounds, assignment constraints), a dependent rounding scheme outputs $X \in \{0,1\}^d$ such that $\mathbb{E}[X_i] = x_i$ for all i and all constraints in $\mathcal{F}$ are satisfied.

The use of dependent randomized rounding is crucial when independent rounding would violate hard constraints or when one

requires strong concentration without additional corrective steps. This technique was first systematically used in the design of approximation algorithms by Gandhi et al. [50], who developed a dependent rounding method with negative correlation for applications in network design and scheduling. Since then, dependent rounding has been employed in a range of approximation algorithms that involve assignment or partitioning constraints. For example, dependent rounding schemes with pairwise negative correlation have been used in approximation algorithms for scheduling on unrelated machines, where each job must be assigned to exactly one machine [59]. In such problems, dependent rounding ensures feasibility (each job gets at most one assignment) and, at the same time, allows one to apply concentration bounds, because the negative correlation property guarantees that no subset of jobs is likely to all choose the same machine at random.

In summary, independent rounding is often preferred for its simplicity when it produces feasible solutions, but dependent rounding is the method of choice when we need to respect global constraints or achieve tighter probabilistic guarantees on the solution structure.

1.4 Scheduling Problems

According to the Cambridge dictionary, scheduling stands for “the job or activity of planning the times at which particular tasks will be done, or events will happen" [24]. In OR, the term has a more specific meaning: scheduling is the problem of assigning a set of jobs to available machines or processors over time, subject to constraints, in order to optimize a given performance measure such as completion time, lateness, or resource utilization [89, 21]. Real-world scheduling problems, such as the airport gate assignment problem in Example 1.1, often involve complex operational constraints. A central challenge lies in formulating these problems as mathematical models and developing efficient algorithms to obtain optimal solutions when possible, or near-optimal solutions with provable performance guarantees.

From a theoretical perspective, scheduling problems are a sub-family of *combinatorial optimization* problems (see Section 1.1) which can be classified using the *three-field notation* introduced by Graham et al. [55] represented as $\alpha|\beta|\gamma$. In this thesis, we consider that J with $|J|= n$ is the set of jobs to be scheduled on a set of m machines. Additionally, let p_{ij} be the processing time of job j on machine i.

The first field, α, describes the *machine environment*. We focus on scheduling problems involving multiple machines that operate in *parallel*, i.e., several machines can process jobs simultaneously, with each machine handling at most one job at a time. We distinguish between several types of parallel machine environments:

- **Unrelated machine environment ($\alpha = R$)**: The processing times, p_{ij}, are arbitrary and there are no restrictions.
- **Related/Uniform machine environment ($\alpha = Q$)**: Each machine i has a speed s_i, and the processing times of jobs are related to machine speeds $p_{ij} = p_j/s_i$.
- **Identical machine environment ($\alpha = P$)**: The processing times, p_{ij}, depend on the jobs and are the same for a job across all the machines, i.e., $p_{ij} = p_j$.
- **Single machine environment ($\alpha = 1$)**: Only a single machine is available, i.e., $p_{1j} = p_j$.

The field β describes the *job characteristics* which may concern an individual job, such as its release time r_j or deadline d_j, or describe interactions, for example, a fixed processing order π shared by all machines. Additional characteristics like precedence relations, sequence-dependent processing times, and machine-eligibility restrictions are studied in the literature (see [89]). Throughout this thesis we consider a *non-preemptive* setting: once a job is being processed, it must continue to completion without interruption or migration to another machine.

The field γ describes the *optimality criterion*. In the scheduling problems studied in this thesis, we use three different optimality criteria:

- **Job completion time**: The time at which a job j finishes being processed is its completion time, C_j. We consider the total completion time, $\sum C_j$, as the optimality criterion of the problem studied in Chapter 2.
- **Number of machines**: In Chapter 3, the objective is to minimize the number of machines used for scheduling the jobs.
- **Machine load**: The sum of the processing times of the jobs assigned to a machine i is its load, L_i. For the scheduling problem studied in Chapter 4, we consider maximizing the minimum load across all machines, $\min_i L_i$, as the optimality criterion.

1.5 Outline of the Thesis

In this thesis, we investigate several scheduling problems. This section summarizes the main results and outlines the structure of the remainder of the thesis.

1.5.1 Measuring the Inefficiency of Decentralized Scheduling on Related Machines

Chapter 2 is based on Berger, Rouhani, and Schröder [13]. Consider a scheduling problem where n jobs need to be processed on m machines with different speeds s_i, with the goal of minimizing the sum of completion times. Each job j has a processing time p_j, where the completion time of a job j on a machine i is the sum of the processing times of jobs processed on i before j plus the processing time of j, divided by s_i. On a single machine, the total completion time is minimized when the jobs are processed in non-decreasing order of their processing times, which is the Shortest Processing Time (SPT) rule [65, 36]. The Minimum Mean Flow Time (MFT) algorithm (Algorithm 1) produces an optimal schedule for the related machine scheduling problem. The *Ibarra-Kim* algorithm (Algorithm 3) is a

simple greedy heuristic for this problem, which was originally designed as an approximation algorithm for unrelated machine scheduling problem [66]. In this chapter, we analyze the approximation factor of the Ibarra-Kim algorithm. This analysis is of our particular interest since

1. Ibarra-Kim algorithm provides a very simple and intuitive approach for the related machine scheduling with the objective of minimizing the sum of completion times

2. The same analysis also establishes an upper bound on the inefficiency of the decentralized related machine scheduling with the objective of minimizing the sum of completion times, described below.

Therefore, by improving the upper bound on the inefficiency of the decentralized related machine scheduling with the objective of minimizing the sum of completion times, we are able to claim an improved upper bound for the Ibarra-Kim algorithm. In the decentralized setting, each job is controlled by an independent decision-maker (or *agent*), who wishes to minimize the completion time of their job. Assuming that the SPT rule is a local policy on each machine that determines job priorities, this setting defines a *scheduling game*. Selfish agents choose machines to minimize the completion time of their own jobs, without considering the effect of their decisions on the completion times of other jobs. Such behavior may lead to a loss in overall efficiency. Performance analysis of such distributed settings requires techniques from both classical optimization and *game theory*. The *price of anarchy* [74] has become a standard way to measure the deterioration caused by selfish behavior. Given a game instance I and a social cost function, the price of anarchy of I is simply the worst-case ratio of the social cost at a *Nash equilibrium* to the social cost at the social optimum where a Nash equilibrium is a state in a game where, given the strategies of all other players, no single player can improve their outcome by unilaterally changing their own strategy. The price of anarchy of a game is then obtained from an instance of

that game with the highest ratio of the price of anarchy. In other words, we are interested in the maximal ratio of the sum of completion times when players are in a Nash equilibrium and the optimal sum of completion times, which is known as the price of anarchy. For the problem we address, it is well known that, under the Local SPT policy, the set of (pure) Nash equilibria coincides exactly with the set of schedules produced by the Ibarra-Kim algorithm [61]. Moreover, an optimal solution is obtained by the MFT algorithm (Algorithm 1) by Horowitz and Sahni [65]. Therefore, by analyzing the price of anarchy of the introduced game, we can also provide an upper bound on the approximation factor of the Ibarra-Kim algorithm. It is proved [64] that the price of anarchy for this problem lies in the interval of $[e/(e-1), 2]$. The upper bound of 2 was improved to $2 - 2/(n^2 + n + nm + m)$ by Zhang et al. [114], which asymptotically still leads to the same bound of 2 even for a fixed number of machines. In Chapter 2, we use primal-dual techniques to prove an upper bound of $2 - 1/(4m - 2)$ as the approximation factor of the Ibarra-Kim algorithm, which only depends on the number of machines. Moreover, in the context of two machines, we improve the approximation ratio to $3/2$.

In the problem considered in Chapter 2, each machine processes its assigned jobs according to a fixed order (SPT) while every job is free to choose a machine. Hence, this setting naturally forms a *scheduling game* where measuring the loss of efficiency caused by the game is of our interest. This motivated us to consider the effect of *fixed order* for another interesting scheduling problem in the next chapter.

1.5.2 Fixed Order Scheduling with Deadlines

Chapter 3 is based on Berger, Rouhani, and Schröder [14]. Fixed order scheduling arises in many operational settings where jobs must be processed in a pre-specified global sequence while jobs can be processed simultaneously on several identical resources. Airport gate planning (Example 1.1) is a notable example: once the daily arrival–departure sequence is published, each gate must handle its assigned flights in that exact order to avoid towing, ground-crew,

and connection conflicts [44]. A comparable requirement exists in block-scheduled operating rooms, where the surgical list fixes the order in which each surgeon's patients enter the rooms; that order cannot be changed once staff and anesthesia teams are organized [25]. Fixed order is likewise essential in blood-platelet production centers, where batches are issued strictly first-in-first-out to ensure that the oldest platelets, whose remaining shelf-life is shortest, are processed first, even though several parallel production lines are available [58]. In all these settings, the goal is to partition the given sequence over as few resources as possible while guaranteeing that every task meets its release time and deadline. Conventional algorithms that rely on freely permuting jobs are therefore inapplicable. Hence, a formal study of fixed order scheduling is needed. In this chapter, we study a variant of scheduling problems where scheduling a set of jobs on the minimum number of machines is required while a global order for processing of jobs on each machine is maintained, and no job is processed after its deadline. We investigate the impact of the imposed processing order on job scheduling on a minimum number of machines. We first study the approximation factors of two simple greedy heuristics for the problem: first-fit and next-fit. Initially, we show that next-fit does not admit a constant approximation factor, neither for the general setting nor for the simplified variant with unit processing times. We then prove that first-fit solves the problem optimally when the processing times are equal and has an approximation factor of 2 in four different classes of instances: (1) if the fixed order aligns with slacks that are in a non-increasing order, (2) if the fixed order aligns with slacks that are in a non-decreasing order, (3) if the fixed order aligns with deadlines that are in a non-increasing order, and (4) if the optimal solution uses at most 3 machines, where the slack of a job is the difference between its processing time and deadline. Then, we provide an $O(\log n)$ approximation factor using a greedy algorithm that fits the largest number of unscheduled jobs on a machine using a dynamic programming approach.

In the problem considered in Chapter 3, each machine processes its

assigned jobs according to a fixed order, while the goal is to minimize the number of machines. However, many scheduling applications instead require balanced resource distribution. Here, instead of reducing the number of active machines, we now fix the set of machines and ask for a *fair* division: we wish to assign jobs to machines so that the least-loaded machine receives as much load as possible. This *max–min* formulation corresponds to the well-known *Santa Claus* problem. Chapter 4 provides a new dependent rounding scheme that can be used to approximate the *budgeted Santa Claus* problem.

1.5.3 Cost Preserving Dependent Rounding for Allocation Problems

Chapter 4 is based on Rohwedder, Rouhani, and Wennmann [92]. This chapter studies a max-min allocation problem, which is called the *budgeted Santa Claus* problem. We are given a set of indivisible resources with identical valuations that need to be distributed among a set of players. Each resource has a nonnegative value (the same for all agents). Resources may be assigned to players, where assigning a resource to a player incurs a nonnegative cost that depends on the pair, and the total assignment cost must not exceed a fixed budget. The objective is to select and allocate resources to players so that the minimum total value received by any agent is as large as possible, subject to the budget. This max-min objective models fairness under a hard cost constraint.

A linear programming relaxation yields a fractional allocation that satisfies the budget and assignment constraints (each item is fractionally assigned, with the total sum at most one). Converting this fractional allocation into an integral one is not straightforward. A simple deterministic rounding, such as rounding each fractional value based on a fixed threshold, is generally not suitable for this setting, as it does not guarantee the feasibility of the rounded solution with respect to the assignment constraints. Randomized rounding offers more flexibility, but independent randomized rounding is not suitable for converting this fractional solution into an integral one

either. First, the per-resource assignment constraints are equality constraints (each resource must be assigned to exactly one player). Independent rounding can assign an item to more than one player or none. Second, the budget constraint is a hard upper bound on total cost; independent rounding respects it only in expectation and can exceed it with non-negligible probability. In addition, it fails under the structural requirements of the rounded solutions (for example, perfect matchings or matroid bases). These are well-known limitations in models with equality assignment constraints and degree/budget bounds and motivate the use of dependent randomized rounding (see [50]).

We provide a dependent randomized rounding algorithm for many-to-many assignments on a bipartite graph $G = (A \cup B, E)$ with three guarantees: (1) cost preservation, i.e., for any cost vector $c \in \mathbb{R}^E$, the rounded integral solution $X \in \{0,1\}^E$ satisfies $c^\mathsf{T} X \leq c^\mathsf{T} x$, where x is the fractional solution of the LP relaxation, (2) degree preservation, i.e., for every vertex v, the number of incident integral edges equals either the floor or the ceiling of the value of the fractional degree of v in the LP solution, (3) concentration, i.e., for a set of vectors $a_j \in [0,1]^E$ with $j \in \{1,\ldots,k\}$, each supported on the incident edges of a vertex, the rounded solution X satisfies with constant probability $|a_j^\mathsf{T} x - a_j^\mathsf{T} X| \leq O(\max\{\log k, \sqrt{a_j^\mathsf{T} x \cdot \log k}\})$ for $j \in \{1,\ldots,k\}$. The algorithm runs in randomized polynomial time. This rounding scheme (Proposition 4.1) is of independent interest and can be used in other allocation problems.

Finally, we apply our proposed cost preserving dependent rounding scheme together with another rounding scheme from Shmoys and Tardos [101] to obtain a randomized $O(\log n)$-approximation for the budgeted Santa Claus problem with identical valuations.

1.6 Publications

The work in this dissertation is based on the following publications.

Published:

- Andre Berger, Arman Rouhani, and Marc Schröder. "An improved bound for the price of anarchy for related machine scheduling". In: *Discrete Optimization* 58 (2025). DOI:10.1016/j.disopt.2025.100911.
- Andre Berger, Arman Rouhani, and Marc Schröder. "Fixed order scheduling with deadlines". In: *Operations Research Letters* 61 (2025). DOI: 10.1016/j.orl.2025.107306.
- Lars Rohwedder, Arman Rouhani, and Leo Wennmann. "Cost Preserving Dependent Rounding for Allocation Problems". In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2025, pp. 127-1.

Chapter 2

Measuring the Inefficiency of Decentralized Scheduling on Related Machines

2.1 Introduction

In many real-world problems, such as assigning tasks to processors in computing systems [33], allocating runways to aircraft [22], managing urban parking spaces [102], or scheduling classrooms in universities [28], limited resources must be allocated efficiently among multiple tasks. In classical optimization problems, it is often assumed that one centralized decision maker possesses complete knowledge of the problem model and all required data [62], and is given the task to find a feasible solution that optimizes a given objective. For instance, this planner might schedule tasks across machines to minimize makespan [89], or determine efficient transportation routes within a network to move goods from origins to destinations [1]. However, in many practical settings, such as routing in communication networks [93], task assignment in distributed environments with self-interested agents [109], or market-based decision making without centralized coordination [84, 30], decisions are not taken by a single centralized

Parts of this chapter are adapted from: Andre Berger, Arman Rouhani, and Marc Schröder. "An improved bound for the price of anarchy for related machine scheduling". In: *Discrete Optimization* 58 (2025). DOI: 10.1016/j.disopt.2025.100911.

decision maker, but by multiple strategic decision makers (e.g., users or firms). These agents often pursue their own objectives, which often yield suboptimal outcomes from a system's perspective [85].

To analyze the inefficiency of such strategic behavior, Koutsoupias and Papadimitriou introduced the concept of the *price of anarchy* (PoA) [74]. The PoA measures inefficiency by comparing the solution of an equilibrium achieved by strategic decisions to the optimal centralized solution. If strategic decisions are made deterministically, the resulting inefficiency is measured by the *pure* PoA, whereas if decisions are randomized according to probability distributions, one considers the corresponding *mixed* PoA. The PoA has been widely utilized to analyze inefficiencies in decentralized problems, such as auctions (where strategic bidding reduces social welfare) [96, 84, 103, 16], network routing (where selfish routing decisions lead to congestion) [97, 95], and scheduling games [39, 74, 30, 113, 48, 73, 109].

Scheduling games are a particular class of resource allocation problems in the area of algorithmic game theory in which jobs have to be assigned to machines. It is usually assumed that each job is controlled by a self-interested agent and that the strategy set for each job is equal to the set of machines. The existing literature considers several variants of the problem and can be split based on the social objective. The two most popular social objectives are the *makespan* (egalitarian social cost) and the *total completion time* (utilitarian social cost). For the makespan objective, which is to minimize the maximum completion time of all jobs, we refer to [112, 8, 30, 7, 39, 49]. For the total completion time objective, which is to minimize the sum of completion times of all jobs, we refer to [15, 35, 38, 91].

In this chapter, we consider a scheduling game on related machines with n jobs that need to be processed on m machines with different speeds. Each job is considered as a player who acts selfishly and aims to deterministically choose a machine to minimize its own completion time. The social objective of the problem is to minimize the total sum of completion times of all jobs, given that the machines schedule

jobs according to the *shortest processing time (SPT)* rule. The SPT rule is a *local sequencing policy*, meaning that after the jobs have selected their machines, each machine independently orders its assigned jobs in nondecreasing order of their processing times. Local sequencing policies refer to the criteria that each machine uses to order the jobs for processing. For instance, a machine might sequence jobs based on the shortest processing time, highest weight, or a combination of factors to minimize its completion time [108]. On the other hand, *precedence constraints* are externally imposed rules that dictate a global ordering among jobs, which means certain jobs must be completed before others can begin, regardless of which machine processes them. Local sequencing policies provide flexibility and are used to optimize performance on individual machines; in contrast, precedence constraints are fixed requirements that must be met by any schedule, ensuring that inter-job dependencies are respected across the entire system.

This is a continuation of the research line of Hoeksma and Uetz [64], where they proved that the price of anarchy for this scheduling game lies in the interval of $[e/(e-1), 2]$. The upper bound of 2 on the price of anarchy was improved to $2 - 2/(n^2 + n + nm + m)$ by Zhang et al. [114], which asymptotically still leads to the same bound of 2 even for a fixed number of machines, and they provided a lower bound of 1.1875 for the case of two machines. In this chapter, we show an upper bound of $2 - 1/(4m - 2)$, which to the best of our knowledge, is the first bound for this problem that is lower than 2 that only depends on the number of machines and not on the number of jobs. Moreover, we further improve this bound to $2 - 1/(2m)$ when the machines have divisible speeds, i.e., if the speed of any machine is divisible by the speed of any slower machine. Lee et al. [77] established that in the context of two machines, the price of anarchy is within the range of $[(3 + \sqrt{3})/4, (1 + \sqrt{5})/2]$. Then, Brokkelkamp [20] improved the lower bound to 1.188. We improve the upper bound of the price of anarchy for this scenario to $3/2$.

It is shown by Heydenreich et al. [61] that, for scheduling games

Table 2.1: Summary of existing results.

Problem Setting	**Pure POA**		
	LB	**UB**	**Our UB**
$P\|\|\sum C_j$	1 [37]	1 [37]	-
$P\|\|\sum w_jC_j$	$(1+\sqrt{2})/2 \approx 1.2$ [71]	$(1+\sqrt{2})/2 \approx 1.2$ [71]	-
$Q\|\|\sum C_j$	$e/(e-1) \approx 1.58$ [64]	2 [64]	$2-1/(4m-2)$ [13]
$Q2\|\|\sum C_j$	1.188 [20]	$\frac{1+\sqrt{5}}{2} \approx 1.618$ [77]	3/2 [13]
$R\|\|\sum w_jC_j$	4 [38]	4 [35]	-

on related machines under the Local SPT policy, the set of solutions produced by the *Ibarra-Kim* algorithm, which was originally designed as an approximation algorithm for the unrelated machine scheduling problem [66], coincides exactly with the set of pure strategy Nash equilibria. In particular, every pure Nash equilibrium can be obtained by an appropriate tie-breaking rule in the Ibarra-Kim algorithm, and every schedule produced by the Ibarra-Kim algorithm is a pure Nash equilibrium. As a result, the price of anarchy of this problem is equal to the approximation ratio for this simple greedy algorithm.

Table 2.1 summarizes the existing results for the pure price of anarchy bounds in utilitarian machine scheduling games with the sum of (weighted) completion times as the objective. In the scheduling notation $\alpha|\beta|\gamma$ introduced by Graham et al. [55], the machine environment and objective considered in our setting are denoted by $Q||\sum C_j$, and in the context of two machines by $Q2||\sum C_j$. Additionally, $P||\sum C_j$ and $P||\sum w_jC_j$ describe scenarios where machines are identical, with the latter incorporating weights w_j associated with each job j. Finally, $R||\sum w_jC_j$ denotes unrelated machine scheduling with weighted jobs.

In Section 2.2, we will introduce the notation that will be used in

the paper, and we will formally define the problem. Additionally, we describe the mean flow algorithm and properties of the optimal solution. We then introduce the Ibarra-Kim algorithm and define pure Nash equilibria as well as the price of anarchy. Furthermore, we present an alternative perspective of the problem used in [20] by introducing linear programming and its dual to analyze the problem. In Section 2.3, we present our approach, starting with the case of two machines and deriving a bound of $3/2$. Subsequently, we extend our analysis to the general case and establish a bound of $2 - 1/(4m - 2)$. In Section 2.4, we consider the weighted case of the problem and provide a lower bound of 2. Finally, we conclude our findings, highlight our contribution, and provide future directions.

2.2 Preliminaries

Consider the scheduling problem where a collection of n jobs from the set J must be scheduled onto a set of m machines from the set M. Let p_j be the processing time of job $j \in J$. If job j is assigned to machine i, then the time needed to execute job j on machine i is equal to p_j/s_i, where s_i denotes the speed of machine i. The corresponding class of scheduling problems is denoted by $Q||\sum C_j$. To simplify the analysis, we normalize the speeds of the machines by dividing the speeds of all machines by the speed of the slowest machine. In this modified setting, we denote the speed of the fastest machine as s_m or s, while keeping the speed of the slowest machine fixed at 1. Without loss of generality, we assume an ordering of the jobs according to their processing times as $p_1 \leq p_2 \leq \ldots \leq p_n$ and a similar ordering of the machines with respect to their speeds as $s_1 = 1 \leq s_2 \leq \ldots \leq s_m = s$. In the case of ties on the ordering, we assume that these are broken consistently and that this is done based on the index. Moreover, if for every pair of machines i and i' belonging to M, where $s_i \geq s_{i'}$, the remainder of the division $s_i/s_{i'}$ is zero, then the configuration is referred to as having *divisible speeds*. Furthermore, for simplicity, we use the shorthand notation $[q] = \{1, \ldots, q\}$ for any $q \in \mathbb{N}$.

We are interested in finding an improved upper bound for the price of

anarchy of related machine scheduling games associated with $Q||\sum C_j$. We denote this family of games by Γ. For each instance I of $Q||\sum C_j$, let $\Gamma(I)$ denote the game in Γ induced by I. In the game $\Gamma(I)$, each job $j \in J$ is a player who acts selfishly and deterministically chooses a machine to minimize its own completion time while the social cost function is the sum of the completion times of all jobs. We consider the setting where all machines process jobs in the shortest processing time (SPT) order, where jobs with smaller processing times have higher *priority*. We assume that ties between jobs are broken consistently according to a fixed total order across all machines, hence, a pure strategy Nash equilibrium is guaranteed to exist [61, 67]. The strategy set for each job is equal to the set of machines M, indicating that the players can choose any machine. The strategies of the jobs are represented by a vector $\tau = (\tau_j)_{j \in J}$, where τ_j denotes the machine chosen by job j. The cost incurred by jobs in a strategy profile $\tau \in M^n$, where a job j has chosen machine τ_j, is determined by the completion times of the respective jobs $C_j(\tau)$. The completion time of job j on machine i is calculated as the sum of its processing time, p_j, and the processing times of all the jobs that select machine i and have a higher priority than j, divided by s_i. The social cost function based on a strategy profile τ is defined as the sum of the completion times for all jobs choosing their machines based on strategy τ

$$SC(\tau) = \sum_{j \in J} C_j(\tau) = \sum_{j \in J} \sum_{\substack{k \leq j \\ \tau_k = \tau_j}} \frac{p_k}{s_{\tau_j}}.$$

Let τ represent a strategy profile, and consider a machine i and a job j. Denote by

$$\psi_i(\tau, j) = |\{k > j \mid \tau_k = i\}|, \tag{2.1}$$

the number of jobs on machine i in strategy profile τ that have strictly lower priority than job j. It is worth noting that the sum of completion times can be reformulated as follows, because the processing time of each job appears in the completion time of itself and of every

lower-priority job on the same machine.

$$SC(\tau) = \sum_{j \in J} p_j \cdot \frac{\psi_{\tau_j}(\tau, j) + 1}{s_{\tau_j}}.$$

2.2.1 Optimal Schedule for Jobs

Consider the problem $1||\sum C_j$, where there is one machine with speed 1 and we aim to minimize the sum of the completion times. Note that the processing time of each job k contributes to its own completion time and to the completion times of all the jobs scheduled after k. Therefore, the social cost of strategy profile τ equals

$$SC(\tau) = \sum_{k=1}^{n} (n - k + 1) \cdot p_k.$$

Therefore, the optimal strategy profile is a schedule that matches longer jobs to smaller values of $(n - k + 1)$ in order to minimize the overall impact of these jobs on the social cost. The mean flow time (MFT) algorithm generalizes this idea to the case of m machines. The complexity of this algorithm is $O(n \log nm)$ [65].

Algorithm 1 MFT Algorithm for $Q||\sum C_j$

1: For each machine i, set $\psi_i = 0$.
2: **while** not all jobs are scheduled **do**
3: Let j be the unscheduled job with the largest index.
4: Schedule job j on a machine i with the smallest $(\psi_i + 1)/s_i$.
5: For that machine i, set $\psi_i = \psi_i + 1$.
6: **end while**
7: Sort the jobs on each machine in SPT order.

The MFT algorithm uses the *positional values* $(\psi_i + 1)/s_i$, which are the values for a job's possible position in the schedule, and assigns the currently unscheduled job with the longest processing time to the

machine with the currently smallest positional value. Note that ψ_i is different than $\psi_i(\tau, j)$ which shows the number of jobs on machine i in strategy profile τ that have strictly lower priority than job j.

Theorem 2.1 (Horowitz and Sahni [65])**.** *The MFT algorithm produces optimal schedules for $Q||\sum C_j$, and any optimal schedule for $Q||\sum C_j$ can be computed by the MFT algorithm with a proper tie-breaking rule.*

In this study, when encountering ties in choosing the smallest value of $(\psi_i + 1)/s_i$ during the execution of the MFT algorithm, we assume the ties are broken by selecting the machine with the largest index (and thus highest speed). We now introduce a lemma from Hoeksma and Uetz [64] that describes a useful characteristic of an optimal schedule and will be instrumental for our proofs. In other words, this lemma asserts that a job assigned to a machine cannot possess a better positional value than the jobs already allocated to any other machine.

Lemma 2.2 (Hoeksma and Uetz [64])**.** *A strategy profile τ^* is optimal for $Q||\sum C_j$ if and only if*

$$\frac{\psi_i(\tau^*, j) + 1}{s_i} \geq \frac{\psi_l(\tau^*, j)}{s_l}, \tag{2.2}$$

for all machines i and l, and for all jobs j.

Consider now the weighted variant of the related machine scheduling games, in which each job j is additionally assigned a positive weight $w_j > 0$ representing its "importance". The goal in the corresponding optimization problem $Q||\sum w_j C_j$, is to find a schedule that minimizes the sum of weighted completion times, or equivalently, weighted mean flow time (WMFT). The weighted scheduling game is defined analogously to the unweighted one. Each job is a player whose strategy set is the set of machines. Given a strategy profile τ, the individual cost of job j is its weighted completion time $w_j C_j(\tau)$, and job j aims to choose a machine deterministically that minimizes this cost. Since $w_j > 0$

is fixed, minimizing $w_j C_j(\tau)$ is equivalent to minimizing $C_j(\tau)$. We consider the WSPT order as the local sequencing policy, under which each machine processes its assigned jobs based on the non-increasing order of weight to the processing time ratios w_j/p_j, where ties are broken in favor of the job with the smaller index. The social cost function based on τ is defined as the sum of the weighted completion times for all jobs choosing their machines based on strategy τ

$$SC(\tau) = \sum_{j \in J} w_j C_j(\tau) = \sum_{j \in J} w_j \sum_{\substack{k \leq j \\ \tau_k = \tau_j}} \frac{p_k}{s_{\tau_j}}.$$

For the single machine case, we already know the following.

Lemma 2.3 (Conway [36])**.** *For a single processor, the weighted mean flow time is minimized if the jobs are processed in non-increasing order of w_j/p_j.*

Algorithm 2 Algorithm WMFT

1: $S^0 = (0, 0, 0); FSUM = 0;$
2: **for** $i = 1$ to n **do**
3: $\quad A = B = 0;$
4: $\quad$ **for** every $(F, f, e) \in S^{i-1}$ **do**
5: $\quad\quad A = A \cup (F + w_i(f + p_i), f + p_i, e + 2^i);$
6: $\quad\quad B = B \cup (F + w_i(FSUM - f + p_i)/s, f, e);$
7: $\quad$ **end for**
8: $\quad S^i =$ merge A, B into a single set ordered by the second coordinate, and for each distinct second coordinate, only the tuple with the smallest first coordinate is retained.
9: $\quad FSUM = FSUM + p_i;$
10: **end for**
11: In S^n choose the 3-tuple with minimum first coordinate.

Imagine we have two related machines with the speed of 1 and s. From this lemma, it follows that the WMFT of schedules corresponding to

each machine is minimized if the jobs in each machine are executed in non-increasing order of w_j/p_j. Consequently, given a partition, its minimum WMFT is easily computed. Thus, the major problem is determining a partition with the minimum WMFT. For this purpose, a dynamic programming approach is proposed [65]. Computing time is exponential in n, and space requirements are also exponential. In this algorithm (see Algorithm 2), sets $S^0, \ldots, S^n$ are computed where each set contains 3-tuples of the form (F, f, e) where F is the current WMFT, f is the finish time on machine 1, and e is an encoded bit string whose ones indicate jobs allocated to machine 1. From S^n we can choose the tuple with the minimum first coordinate, the corresponding third coordinate yielding the proper schedule. Algorithm 2 is an exact algorithm for computing the minimum WMFT. The inputs of this algorithm are n jobs with $p_1, \ldots, p_n$ and $w_1, \ldots, w_n$ ordered so that $w_i/p_i \geq w_{i+1}/p_{i+1}$, and the speed of the second machine s. The output is a schedule that assigns jobs to machines 1 and 2 such that the WMFT is minimized. It is worth mentioning that Algorithm 2 can be exponential in the worst case.

2.2.2 Nash Equilibria and Price of Anarchy

Consider an instance I of $Q||\sum C_j$ and its induced related machine scheduling game $\Gamma(I)$. As mentioned in Section 2.2, to define the scheduling game played by the jobs, we make the natural assumption that each machine sequences its jobs in the locally optimal order, which is the SPT rule. As it turns out, if the local sequencing policy is the SPT rule, the schedules resulting from (pure) Nash equilibria coincide with the outcome of the *Ibarra-Kim* algorithm (see Algorithm 3). As a result, one can exploit the results on the performance of this algorithm by analyzing the price of anarchy of the related machine scheduling game. Therefore, we consider the local SPT order as the sequencing policy for the objective $\sum_j C_j$, which is based on the non-decreasing order of processing times, and we consider the local WSPT order (a.k.a., Smith's rule) as the sequencing policy for the objective $\sum_j w_j C_j$, which is based on the non-increasing order of weight to processing time ratios

w_j/p_j. For both cases, we assume that ties are broken in consistently across all machines (in favor of the job with the smaller index); hence, a pure strategy Nash equilibrium is guaranteed to exist [61, 67].

A strategy profile $\tau = (\tau_{-j}, \tau_j) \in M^n$ for the game $\Gamma(I)$ is a *(pure) Nash equilibrium* if for all $i \in M$ and for all $j \in J$,

$$C_j(\tau) = C_j(\tau_{-j}, \tau_j) = \sum_{\substack{k \leq j \\ \tau_k = \tau_j}} \frac{p_k}{s_{\tau_j}} \leq \frac{p_j}{s_i} + \sum_{\substack{k < j \\ \tau_k = i}} \frac{p_k}{s_i} = C_j(\tau_{-j}, i).$$

It means that a Nash equilibrium is a strategy profile in which no job can improve its completion time by changing to a different machine when machines adhere to a local sequencing policy (which is publicly known).

Let $\chi(I)$ be the set of all possible pure Nash equilibria of the game $\Gamma(I)$. Then the *(pure) price of anarchy* of $\Gamma(I)$ is defined as

$$PoA(\Gamma(I)) = \max_{\tau \in \chi(I)} \frac{SC(\tau)}{SC(\tau^*)}, \tag{2.3}$$

where $\tau^* \in M^n$ is the strategy profile that minimizes the social cost for instance I. The pure price of anarchy for the family of related machine scheduling games Γ is then defined as the supremum over all instances, i.e.,

$$PoA(\Gamma) = \sup_{I \in Q||\sum C_j} PoA(\Gamma(I)).$$

Clearly, different local sequencing policies across machines lead to different Nash equilibria, and the resulting price of anarchy depends on the particular policy implemented. The analysis of these local sequencing policies is referred to as *coordination mechanisms* [29].

as a heuristic/approximation algorithm for scheduling on unrelated machines. The well-known *Ibarra-Kim* algorithm was first introduced by Ibarra and Kim [66] as an approximation algorithm for scheduling on unrelated machines. For the related machines, the algorithm is

Algorithm 3 Ibarra-Kim Algorithm for Problem $Q||\sum C_j$

1: Sort jobs based on non-decreasing order of p_j, breaking ties by smaller job index.
2: **while** not all jobs are scheduled **do**
3: Let j be the first unscheduled job from the sorted job list.
4: Schedule j on a machine i where job j has minimum completion time.
5: **end while**

described as in Algorithm 3. The following theorem shows that the set of strategy profiles corresponding to schedules produced by the Ibarra–Kim algorithm coincides exactly with the set of pure Nash equilibria.

Theorem 2.4 (Heydenreich et al. [61], Immorlica et al. [67])**.** *The set of pure Nash equilibria for the local SPT policy is precisely the set of solutions of the Ibarra-Kim algorithm.*

Hence, examining the pure PoA of the family Γ is equivalent to analyzing the approximation factor of the Ibarra-Kim algorithm for related machine scheduling.

Algorithm 4 WSPT Algorithm for Problem $Q||\sum w_j C_j$

1: Sort jobs based on non-increasing order of $\frac{w_j}{p_j}$, breaking ties by smaller job index.
2: **while** not all jobs are scheduled **do**
3: Let j be the first unscheduled job from the sorted job list.
4: Schedule j on a machine i where job j has minimum completion time.
5: **end while**

Similarly, one can show that the set of Nash equilibria of the local WSPT policy is equal to the set of all possible outputs of the WSPT algorithm (Algorithm 4).

Theorem 2.5 (Heydenreich et al. [61], Immorlica et al. [67])**.** *The set of pure Nash equilibria for the local WSPT policy is precisely the set of solutions of the WSPT algorithm.*

For identical machines, the worst-case behavior of the WSPT algorithm was analyzed by Kawaguchi and Kyan [71]. Thus, the price of anarchy of the Local WSPT policy follows from the analysis of that algorithm. For identical machines, the work of Kawaguchi and Kyan [71] thus yields the following.

Theorem 2.6 (Heydenreich et al. [61], Kawaguchi and Kyan [71])**.** *For parallel machines, the price of anarchy of the local WSPT policy for minimizing* $\sum w_j C_j$ *in the sequencing model is* $(1+\sqrt{2})/2 \approx 1.207$*.*

The primal-dual method, introduced by Bilò [17], is an approach for upper bounding the price of anarchy. We describe this method for the family Γ of related machine scheduling games.

Assume that τ^* is an optimal strategy profile and τ is a Nash equilibrium. The primal-dual method aims to construct a linear program $LP(s, \tau^*, \tau)$ that maximizes $SC(\tau)$ with respect to the processing times and subject to the constraints that $SC(\tau^*)$ is normalized to one and τ must be a Nash equilibrium. Observe that the constraint that $SC(\tau^*)$ is normalized to one is without loss of generality because processing times can be scaled down uniformly in a way that the social cost of the optimal strategy profile becomes equal to one. Based on weak duality, any feasible solution for the dual linear program provides an upper bound on the optimal primal solution and thus an upper bound on the price of anarchy.

Brokkelkamp [20] employed the primal-dual method using a weak linear programming formulation (see LP_{weak} below) for games in the family Γ to obtain the same bounds as Hoeksma and Uetz [64]. This version of the LP is called "weak" since it does not assume that τ must be a Nash equilibrium, but rather assumes that in τ, no job can improve

its completion time by deviating to its optimal machine (and not to all other machines as required in a Nash equilibrium). This idea follows from the smoothness concept [94]. The LP is obtained as follows.

$$LP_{weak}(s,\tau^*,\tau) = \max_{p} \sum_{j\in J} \frac{1}{s_{\tau_j}} \sum_{\substack{k\le j \\ \tau_k=\tau_j}} p_k \qquad s.t.$$

$$\sum_{j\in J} \frac{1}{s_{\tau_j^*}} \sum_{\substack{k\le j \\ \tau_k^*=\tau_j^*}} p_k = 1, \tag{2.4}$$

$$\frac{1}{s_{\tau_j}} \left[p_j + \sum_{\substack{k<j \\ \tau_k=\tau_j}} p_k \right] - \frac{1}{s_{\tau_j^*}} \left[p_j + \sum_{\substack{k<j \\ \tau_k=\tau_j^*}} p_k \right] \le 0 \quad (\forall\, j \in J), \tag{2.5}$$

$$p_j - p_{j+1} \le 0 \quad (\forall\, j \in [n-1]), \tag{2.6}$$

$$-p_1 \le 0.$$

The inputs of this LP are $s = (s_1, \ldots, s_m)$, τ^*, and τ, where s is the vector of machine speeds, τ^* is the optimal strategy profile, and τ is a strategy profile in which no job can improve its completion time by changing to its optimal machine. Observe that the optimal strategy profile can be deduced from the speeds by using the MFT algorithm. Also, notice that τ is allowed to be a Nash equilibrium, but need not be. The variables of the LP are the processing times, i.e., the vector $p = (p_1, \ldots, p_n)$. To obtain the dual of the linear program $LP_{weak}(s,\tau^*,\tau)$, the dual variables are defined as β for the normalization constraint, Constraint (2.4), y_j variables for each of the deviation constraints, Constraints (2.5), and z_1 to z_{n-1} for the SPT constraints, Constraints (2.6). Note that $z_n = z_0 = 0$.

The dual is obtained as follows.

$$\text{Dual}_{weak}(s, \tau^*, \tau) = \min\ \beta \quad s.t.$$

$$\beta \cdot \frac{1}{s_{\tau_k^*}} \sum_{\substack{j \geq k \\ \tau_k^* = \tau_j^*}} 1 + \frac{1}{s_{\tau_k}} \sum_{\substack{j \geq k \\ \tau_j = \tau_k}} y_j$$

$$- \frac{y_k}{s_{\tau_k^*}} - \frac{1}{s_{\tau_k}} \sum_{\substack{j > k \\ \tau_j^* = \tau_k}} y_j + z_k - z_{k-1} \geq \frac{1}{s_{\tau_k}} \sum_{\substack{j \geq k \\ \tau_j = \tau_k}} 1, \qquad (\forall k \in J)$$

$$y_k \geq 0, \qquad (\forall k \in J)$$

$$z_k \geq 0. \qquad (\forall k \in [n-1])$$

An upper bound of 2 can be easily obtained by a feasible solution of $Dual_{weak}$ given by $\beta = 2$, $y_j = 1$, and $z_j = 0$ for all $j \in J$ [20]. In this study, we use a more relaxed version of LP_{weak} to improve the existing upper bound of 2.

2.3 Upper Bounds on PoA under the SPT Rule

In this section, we improve the existing upper bound on the pure price of anarchy of the family Γ of related machine scheduling games. First, we analyze the case where the number of machines is equal to 2 and prove an upper bound of $3/2$ for the pure price of anarchy. Then we provide an example showing that our analysis is tight in this case. Next, we generalize the upper bound to $2 - 1/(4m - 2)$ for the case of m machines, and we also show that when the machines have divisible speeds, this bound can be further improved to $2 - 1/(2 \cdot m)$.

Given a speed vector s, an optimal strategy profile τ^*, and a strategy profile τ in which no job can improve its completion time by changing to a different machine suggested by τ^*, the optimal value of the following LP provides an upper bound to the sum of completion times of strategy profile τ. Since the total completion time of the optimal

strategy profile is constrained to be 1, and since every pure Nash equilibrium also satisfies the deviations constraints in τ (and therefore the aggregated deviation constraint), then the feasible set considered by LP_{sum} contains all pure Nash equilibria, and therefore, the optimal value of LP_{sum} provides an upper bound on the pure price of anarchy. Hence, we can use the dual of this LP to derive an upper bound for it. Define LP_{sum} as follows.

$$LP_{sum}(s, \tau^*, \tau) = \max_{p} \sum_{j \in J} \frac{1}{s_{\tau_j}} \sum_{\substack{k \leq j \\ \tau_k = \tau_j}} p_k \qquad s.t.$$

$$\sum_{j \in J} \frac{1}{s_{\tau_j^*}} \sum_{\substack{k \leq j \\ \tau_k^* = \tau_j^*}} p_k = 1, \tag{2.7}$$

$$\sum_{j \in J} \left[\frac{1}{s_{\tau_j}} \left(p_j + \sum_{\substack{k < j \\ \tau_k = \tau_j}} p_k \right) \right] - \sum_{j \in J} \left[\frac{1}{s_{\tau_j^*}} \left(p_j + \sum_{\substack{k < j \\ \tau_k = \tau_j^*}} p_k \right) \right] \leq 0, \tag{2.8}$$

$$p_j - p_{j+1} \leq 0 \quad (\forall\, j \in [n-1]), \tag{2.9}$$

$$-\, p_1 \leq 0.$$

Our analysis is based on a relaxation of the LP_{weak} formulation. Note that Equation (2.8) is obtained by summing all the deviation constraints in Equation (2.5). Instead of considering the individual deviation constraints for each job, we aggregate all deviation inequalities into a single constraint. This yields a weaker formulation with a larger feasible region, but simplifies the dual structure, which allows us to construct improved feasible dual solutions and derive stronger upper bounds. To obtain the dual linear program $Dual_{sum}$ of LP_{sum}, the dual variables are defined as β for the normalization constraint, Constraint (2.7), y for the deviation constraint, Constraint (2.8), and z_1 to z_{n-1} for the SPT constraints, Constraints (2.9). Note that $z_n = z_0 = 0$. The

$Dual_{sum}$ is defined as follows.

$$\text{Dual}_{sum}(s, \tau^*, \tau) = \min\ \beta \quad s.t.$$

$$\beta \cdot \frac{1}{s_{\tau_k^*}} \sum_{\substack{j \geq k \\ \tau_k^* = \tau_j^*}} 1 + \frac{y}{s_{\tau_k}} \sum_{\substack{j \geq k \\ \tau_j = \tau_k}} 1$$

$$- \frac{y}{s_{\tau_k^*}} - \frac{y}{s_{\tau_k}} \sum_{\substack{j > k \\ \tau_j^* = \tau_k}} 1 + z_k - z_{k-1} \geq \frac{1}{s_{\tau_k}} \sum_{\substack{j \geq k \\ \tau_j = \tau_k}} 1, \qquad (\forall k \in J)$$

$$z_k \geq 0, \qquad (\forall k \in [n-1])$$

$$y \geq 0.$$

Upon careful examination of $Dual_{sum}$, it becomes evident that $Dual_{sum}$ can be derived from $Dual_{weak}$ by uniformly setting all y_j values to y. This is in line with the dual solution proposed by Brokkelkamp [20]: $\beta = 2$, $y_j = 1$, and $z_j = 0$ for all $j \in J$. To improve this dual solution, we set $y = 1$, so that the second term on the left-hand side of the deviation constraint of $Dual_{sum}$ cancels with the term on the right-hand side of the inequality. Additionally, we will define some z_j variables to take non-zero values in order to attain a bound that is strictly below 2 for a fixed number of machines.

2.3.1 Two Machines

We first assume the number of machines is equal to 2 and provide a bound of $3/2$.

Theorem 2.7. *$PoA(\Gamma(I)) \leq 3/2$ for every instance I in class $Q2||\sum C_j$.*

Proof. Recall that we assume that $s_1 = 1$ and $s_2 = s \geq 1$. Without loss of generality, we can assume that $n > \lfloor s \rfloor$, because if $n \leq \lfloor s \rfloor$, the MFT

and Ibarra-Kim algorithm induce a strategy profile with the same cost, implying $PoA(\Gamma(I)) = 1$.

We will show that $\beta = 3/2$, $y = 1$, $z_{n-\lfloor s \rfloor} = \frac{\lfloor s \rfloor}{s} \cdot \frac{\lfloor s \rfloor}{s+\lfloor s \rfloor}$, and $z_j = 0$ for all $j \neq n - \lfloor s \rfloor$, constitutes a feasible solution for $Dual_{sum}$, valid for all possible values of τ, τ^* and s, and thus yields an upper bound for the price of anarchy. Recall that τ^* denotes the optimal strategy profile for the given speeds s and can be deduced from the speeds by using the MFT algorithm. By utilizing the definition introduced by Equation (2.1), we rewrite the first dual constraint in the following manner.

$$\beta \cdot \frac{\psi_{\tau_k^*}(\tau^*, k) + 1}{s_{\tau_k^*}} + y \cdot \frac{\psi_{\tau_k}(\tau, k) + 1}{s_{\tau_k}} - \frac{y}{s_{\tau_k^*}} - y \cdot \frac{\psi_{\tau_k}(\tau^*, k)}{s_{\tau_k}} + z_k - z_{k-1} \geq \frac{\psi_{\tau_k}(\tau, k) + 1}{s_{\tau_k}}. \quad (\forall k \in J)$$

By setting $y = 1$ and streamlining the dual constraint, we obtain a lower bound for β as follows.

$$\beta \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{s_{\tau_k}} \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1} + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1}. \quad (2.10)$$

In the remainder of the proof, we show that Inequality (2.10) is satisfied for all jobs when $\beta = 3/2$.

Based on the MFT algorithm, we can observe that the $\lfloor s \rfloor$ longest jobs are scheduled on the fast machine, and the job with the next highest processing time, job $n - \lfloor s \rfloor$, is scheduled on the slow machine. Let us refer to the job $n - \lfloor s \rfloor$ as the ***critical job*** (see Figure 2.1). Initially, we examine the feasibility of our dual solution for jobs $k \in J$, where $k < n - \lfloor s \rfloor$ (refer to the jobs scheduled in the shaded area of Figure 2.1

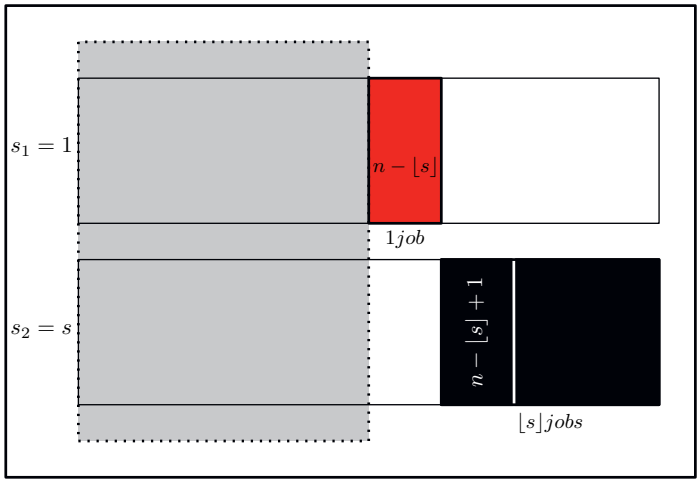


Figure 2.1: Critical job $n - \lfloor s \rfloor$ for any optimal schedule produced by the MFT algorithm for two machines.

for an illustration). In accordance with our dual solution, $z_k = 0$ for all $k < n - \lfloor s \rfloor$. Thus, we have

$$\beta = \frac{3}{2} \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + 1 \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k}},$$

where the first inequality follows from the fact that $\psi_{\tau_k^*}(\tau^*, k) \geq 1$ for these jobs (see Figure 2.1), and the second inequality from Inequality (2.2). Hence, Inequality (2.10) is satisfied for all $k < n - \lfloor s \rfloor$.

Now, it remains to extend our analysis for jobs $k \in J$, where $k \geq n - \lfloor s \rfloor$. For the critical job $n - \lfloor s \rfloor$ we know that $\psi_{\tau^*_{n-\lfloor s \rfloor}}(\tau^*, n - \lfloor s \rfloor) = 0$, $s_{\tau^*_{n-\lfloor s \rfloor}} = 1$, $z_{n-\lfloor s \rfloor - 1} = 0$, and $z_{n-\lfloor s \rfloor} = \frac{\lfloor s \rfloor}{s} \cdot \frac{\lfloor s \rfloor}{s + \lfloor s \rfloor}$. We also know that $1 \leq \frac{s}{\lfloor s \rfloor}$ for $s \geq 1$. Thus

$$1 \leq \frac{s}{\lfloor s \rfloor} \Leftrightarrow 1 + 1 \leq \frac{s + \lfloor s \rfloor}{\lfloor s \rfloor} \Leftrightarrow \frac{1}{2} \geq \frac{\lfloor s \rfloor}{s + \lfloor s \rfloor}. \tag{2.11}$$

Moreover, $\psi_{\tau_{n-\lfloor s \rfloor}}(\tau^*, n - \lfloor s \rfloor) = 0$ if the critical job is assigned to the slow machine in the Nash equilibrium, while $\psi_{\tau_{n-\lfloor s \rfloor}}(\tau^*, n - \lfloor s \rfloor) = \lfloor s \rfloor$ if the critical job is assigned to the fast machine in the Nash equilibrium. By substituting the relevant values into Inequality (2.10) and leveraging the previous inequality, we derive the following bound for the critical job.

$$\beta = \frac{3}{2} = 1 + \frac{1}{2} \geq 1 + \frac{\lfloor s \rfloor}{s + \lfloor s \rfloor} = 1 + \frac{\lfloor s \rfloor}{s} - \frac{\lfloor s \rfloor}{s} \cdot \frac{\lfloor s \rfloor}{s + \lfloor s \rfloor}$$
$$\geq 1 + \frac{\psi_{\tau_{n-\lfloor s \rfloor}}(\tau^*, n - \lfloor s \rfloor)}{s_{\tau_{n-\lfloor s \rfloor}}} - z_{n-\lfloor s \rfloor}.$$

Thus, Inequality (2.10) is satisfied. Lastly, let us focus on analyzing the jobs $k > n - \lfloor s \rfloor$ (see the jobs in the black area in Figure 2.1). These jobs are always assigned to the fastest machine in the optimal strategy profile. Considering all machines where a job $k > n - \lfloor s \rfloor$ could potentially be scheduled in a Nash equilibrium, we can simplify Inequality (2.10) and see which one results in a stronger bound for β. If $\tau_k = 1$, then, $\psi_{\tau_k}(\tau^*, k) = 0$ and Inequality (2.10) becomes

$$\beta \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1}.$$

If $\tau_k = 2$, Inequality (2.10) becomes

$$\beta \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k^*}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k^*}} + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1}$$
$$= 1 + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1}.$$

Hence, it becomes evident that we obtain a higher bound when a job $k > n - \lfloor s \rfloor$ is scheduled on the fast machine in the Nash equilibrium. Therefore, our focus is on the latter inequality. The dual variable $z_{n-\lfloor s \rfloor}$ is present in the Inequality (2.10) for job $j_{n-\lfloor s \rfloor+1}$. Based on what we discussed earlier and according to our dual solution, for this particular

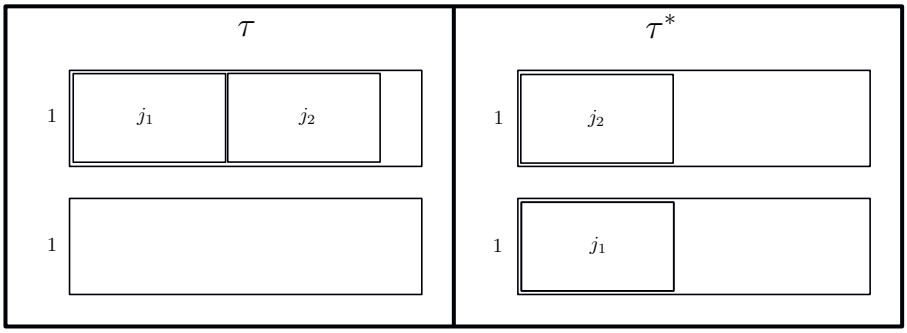


Figure 2.2: Schedules of Example 3.2.

job, we have that Inequality (2.10) is satisfied because

$$\begin{aligned}\beta = \frac{3}{2} = 1 + \frac{1}{2} \geq 1 + \frac{\lfloor s \rfloor}{s + \lfloor s \rfloor} &= 1 + z_{n-\lfloor s \rfloor} \cdot \frac{s}{\lfloor s \rfloor} \\ &= 1 + z_{n-\lfloor s \rfloor} \cdot \frac{s_{\tau^*_{n-\lfloor s \rfloor + 1}}}{\psi_{\tau^*_{n-\lfloor s \rfloor + 1}}(\tau^*, n - \lfloor s \rfloor + 1) + 1},\end{aligned}$$

where the first inequality follows from $\frac{\lfloor s \rfloor}{s} \leq 1$ and $1 + \frac{\lfloor s \rfloor}{s+\lfloor s \rfloor}$ being increasing in $\frac{\lfloor s \rfloor}{s}$ (see Inequality (2.11)). Similarly, according to our dual solution, the dual SPT constraints associated with all other jobs $k > n - \lfloor s \rfloor + 1$ are set to zero. Consequently, for these remaining jobs, we obtain the inequality $\beta \geq 1$, which is satisfied.

Summarizing, we conclude that the solution $\beta = 3/2$, $y = 1$, $z_{n-\lfloor s \rfloor} = \frac{\lfloor s \rfloor}{s} \cdot \frac{\lfloor s \rfloor}{s+\lfloor s \rfloor}$, and $z_j = 0$ for all $j \in J$ is a feasible solution for $Dual_{sum}$ that is valid for all possible values of s, τ^*, and τ. □

The previous theorem establishes an upper bound of $3/2$ for two machines. To demonstrate that we cannot improve this upper bound using $Dual_{sum}$, we present an example that illustrates the tightness of the bound for this method.

Example 2.1. *Consider the instance depicted in Figure 2.2. There are two jobs j_1 and j_2 with the processing times of p_1 and p_2, where $p_1 \leq p_2$. Moreover, there are two identical machines with a speed of 1. Observe that τ^* is an optimal strategy profile and τ is a strategy profile in which no job can improve its completion time by changing to its optimal machine.*

The corresponding LP_{sum} and $Dual_{sum}$ are presented as follows.

$$
\begin{aligned}
LP_{sum}(s,\tau^*,\tau) = \max \quad & 2\cdot p_1 + p_2 \\
s.t. \quad & p_1 + p_2 = 1, \\
& p_1 \leq p_1, \\
& p_1 - p_2 \leq 0, \\
& -p_1 \leq 0.
\end{aligned}
\qquad
\begin{aligned}
Dual_{sum}(s,\tau^*,\tau) = \min \quad & \beta \\
s.t. \quad & \beta + z_1 \geq 2, \\
& \beta - z_1 \geq 1, \\
& z_1 \geq 0.
\end{aligned}
$$

By adding the two dual constraints, it is clear that $\beta \geq 3/2$. By setting $\beta = 3/2$ and $z_1 = 1/2$, we have a feasible (and thus optimal) solution that attains $3/2$. Note that LP_{sum} does not impose a constraint for job j_2 and that is why τ is not a Nash equilibrium. That is why this method it is not able to prove the actual price of anarchy of the example, which is equal to 1, but only provides an upper bound of 3/2. We leave open the possibility of getting a better bound when we restrict τ in LP_{sum} to be a Nash equilibrium.

2.3.2 Three or more Machines

In the previous analysis, for two machines, we established an upper bound of $3/2$. Now, we extend the analysis for the general case of m machines and we obtain a slightly weaker bound of $2 - 1/(4m-2)$.

Before we proceed with the statement of the main result, we first provide two technical lemmas.

Lemma 2.8. *Let $a, b \in \mathbb{R}$ with $a \geq b \geq 1$. Then, we have $\frac{a}{2b} \leq \lfloor \frac{a}{b} \rfloor$.*

Proof. We can divide the analysis into two cases.

(1) $2b \leq a$.

$$\lfloor \frac{a}{b} \rfloor - \frac{a}{2b} = \frac{a}{b} - x - \frac{a}{2b} = \frac{a}{2b} - x \geq 0,$$

where $0 \leq x < 1$ is the decimal part of $\frac{a}{b}$. Additionally, building upon the given assumption, it is evident that $\frac{a}{2b} \geq 1$. Therefore, the last inequality holds.

(2) $b \leq a < 2b$.

$$1 \leq \frac{a}{b} < 2 \Rightarrow \frac{1}{2} \leq \frac{a}{2b} < 1 \Rightarrow \frac{a}{2b} < \lfloor \frac{a}{b} \rfloor,$$

where the last inequality holds since $\lfloor \frac{a}{b} \rfloor \geq 1$. □

Lemma 2.9. *Let $k-1, k \in J$ be two consecutive jobs with $\tau_k^* \neq m$. Then, we have $\tau_{k-1}^* \neq \tau_k^*$.*

Proof. Recall that the MFT algorithm first assigns job k and then assigns job $k-1$. Assume that the MFT algorithm assigns job k to machine i. As k stands as the most recently scheduled job, its positional value cannot be smaller than previously scheduled jobs. Hence,

$$\frac{\psi_l(\tau^*, k)}{s_l} \leq \frac{\psi_i(\tau^*, k) + 1}{s_i}. \qquad (\forall l \in M)$$

Thus, using the previous inequality, we observe that for all $l \geq i$,

$$\frac{\psi_l(\tau^*, k) + 1}{s_l} \leq \frac{\psi_l(\tau^*, k)}{s_l} + \frac{1}{s_i} \leq \frac{\psi_i(\tau^*, k) + 2}{s_i}.$$

The term on the left-hand side shows the positional value of job $k-1$ on machine ℓ, and the term on the right-hand side shows the positional value of job $k-1$ on machine i. Thus, we can infer that in the MFT algorithm, it is infeasible to position two successive jobs adjacent to each other on the same machine, except for the fastest one. □

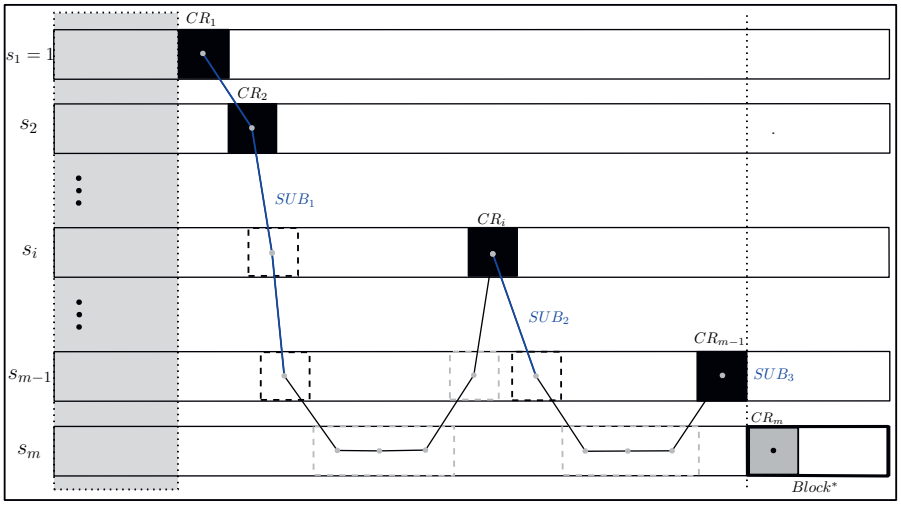


Figure 2.3: A potential chain of consecutive jobs from CR_1 to CR_m in an optimal schedule obtained from the MFT algorithm, together with its sub-chains. Only jobs that are part of a sub-chain (blue lines) have a positive z_j-value.

It is important to emphasize that the above proof exclusively establishes the impossibility of placing two consecutive jobs on a single machine, as a more favorable positional value can consistently be found for the subsequent job among the higher-speed machines. Nonetheless, it's worth noting that the subsequent job could potentially be accommodated on a machine with a lower speed.

Theorem 2.10. *$PoA(\Gamma(I)) \leq 2 - \frac{1}{4m-2}$ for every instance I in class $Q||\sum C_j$, where m is the number of machines.*

Proof. Define the set of ***critical jobs*** CR, one for each machine, where each $CR_i \in CR$ for $i \in [m-1]$ is the last job scheduled according to the optimal strategy profile on every machine i and $CR_m \in CR$ is the smallest job in $Block^*$ (see Figure 2.3 for an illustration), where

$Block^*$ contains all the consecutive jobs with the longest processing times before the first job is assigned to the second fastest machine by the MFT algorithm. The significance of the last job scheduled on each machine becomes crucial in our analysis.

We construct sub-chains as follows. Sub-chain 1, denoted by SUB_1, is the set of jobs $\{CR_1, CR_1+1, \ldots, j-1\}$, where $j = \min\{j' \geq CR_1 \mid \tau^*_{j'} = m\}$. Notice that a sub-chain might contain multiple critical jobs. Then, sub-chain i, denoted by SUB_i, for $i \geq 2$ is defined as the set of jobs $\{CR_{i'}, CR_{i'}+1, \ldots, j-1\}$, where $CR_{i'}$ is the smallest indexed critical job that is not part of a previous sub-chain and $j = \min\{j' \geq CR_{i'} \mid \tau^*_{j'} = m\}$. Let SUB denote the set of all sub-chains and $\mathcal{SUB} = \bigcup SUB_i$ for $i \geq 1$. For $j \in \mathcal{SUB}$ denote by r_j the machine index of the initial critical job within the sub-chain that includes job j. We will show that

$$\beta = 2 - \frac{1}{2 \cdot (2m-1)},$$
$$y = 1,$$
$$z_j = \begin{cases} i \cdot \frac{1}{2 \cdot s_{r_j} \cdot (2m-1)} & \text{, if } j \in \mathcal{SUB} \text{ and } CR_i \leq j < CR_{i+1}, i \in [m-1] \\ 0 & \text{, otherwise} \end{cases}$$

constitutes a feasible solution for $Dual_{sum}$, valid for all possible values of s, τ^*, and τ, and thus yields an upper bound for the price of anarchy. By this definition, only jobs j that are part of a sub-chain will have a positive z_j value, and the value depends on the sub-chain and on the position of that job between critical jobs within the sub-chain. Similar to the proof of Theorem 2.7, we will now show that Inequality (2.10) holds for all jobs $k \in J$.

For jobs $k \in J$ with $k < CR_1$ (all the jobs in the shaded area of Figure 2.3), we have that

$$\beta \geq \frac{3}{2} \geq \frac{1}{\psi_{\tau^*_k}(\tau^*, k) + 1} + 1 \geq \frac{1}{\psi_{\tau^*_k}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau^*_k}(\tau^*, k) + 1} \cdot \frac{s_{\tau^*_k}}{s_{\tau_k}},$$

where the second inequality follows from $\psi_{\tau_k^*}(\tau^*, k) \geq 1$, and the third inequality from Inequality (2.2). Since $z_{k-1} - z_k = 0$, Inequality (2.10) is satisfied for all $k < CR_1$.

For all k with $CR_1 < k < CR_m$, $\tau_k^* \neq m$, and $k \neq CR_i$ for $i = 2, \ldots, m-1$, we have that

$$\beta \geq \frac{3}{2} \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + 1 \geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k}},$$

where the second inequality follows from $\psi_{\tau_k^*}(\tau^*, k) \geq 1$, and the third inequality from Inequality (2.2). Since $z_{k-1} - z_k = 0$ for all these jobs, Inequality (2.10) is satisfied. It is important to note that the dual variables of the SPT constraints for these jobs are non-zero if they are in a sub-chain and zero otherwise.

Our focus now shifts to verifying the feasibility of our dual fitting concerning the dual constraints of the critical jobs that are not on the fastest machine. For $k \in CR$ with $k \neq CR_m$, we have that

$$\begin{aligned}\beta = 2 - \frac{1}{4m-2} &\geq 2 + (z_{k-1} - z_k) \cdot s_{\tau_k^*} \\ &\geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k}} \\ &\quad + (z_{k-1} - z_k) \cdot s_{\tau_k^*},\end{aligned}$$

using $z_{k-1} - z_k \leq -\frac{1}{2 \cdot s_{r_k} \cdot (2m-1)}$ (where we recall that each critical job can either serve as the starting point of a sub-chain or reside within a sub-chain originating from a prior critical job), $s_{\tau_k^*} \geq s_{r_k}$, $\psi_{\tau_k^*}(\tau^*, k) = 0$, and Inequality (2.2). Hence, Inequality (2.10) is satisfied for all $k \in CR$ with $k \neq CR_m$.

Now, let us focus on jobs k with $\tau_k^* = m$. We first prove a lemma that turns out to be useful in our remaining analysis.

Claim 2.11. *Consider job k with $\tau_k^* = m$ where $k-1$ is the last job of a sub-chain. Then, $\psi_m(\tau^*, k) + 1 = \lfloor \frac{s_m}{s_{r_{k-1}}} \rfloor$.*

Proof. Recall that r_{k-1} is the machine index of the initial critical job of the sub-chain that includes job $k-1$. Since $k-1$ is the last job of the sub-chain, we have $\psi_m(\tau^*, k)+1 = \psi_m(\tau^*, CR_{r_{k-1}})$. Based on the MFT algorithm we have for job k and $CR_{r_{k-1}}$, respectively,

$$\frac{\psi_m(\tau^*, CR_{r_{k-1}})}{s_m} \leq \frac{1}{s_{r_{k-1}}},$$
$$\frac{1}{s_{r_{k-1}}} < \frac{\psi_m(\tau^*, CR_{r_{k-1}}) + 1}{s_m}.$$

By using both inequalities we have

$$\frac{s_m}{s_{r_{k-1}}} - 1 < \psi_m(\tau^*, CR_{r_{k-1}}) \leq \frac{s_m}{s_{r_{k-1}}}.$$

Using $\frac{s_m}{s_{r_{k-1}}} - 1 < \lfloor \frac{s_m}{s_{r_{k-1}}} \rfloor \leq \frac{s_m}{s_{r_{k-1}}}$, and the fact that both $\psi_m(\tau^*, CR_{r_{k-1}})$ and $\lfloor \frac{s_m}{s_{r_{k-1}}} \rfloor$ are integers, we can conclude

$$\psi_m(\tau^*, CR_{r_{k-1}}) = \lfloor \frac{s_m}{s_{r_{k-1}}} \rfloor.$$

□

For all jobs k with $\tau_k^* = m$ and $k-1 \notin \mathcal{SUB}$, we have that $z_{k-1} - z_k = 0$ and thus Inequality (2.10) yields $\beta \geq 3/2$ (this can be shown by considering the two different cases: $k = n$ and thus $\psi_{\tau_k}(\tau^*, k) = 0$, or $k \neq n$ and thus $\psi_{\tau_k^*}(\tau^*, k) \geq 1$ and the result follows from Lemma 2.2), which is satisfied. For all jobs k with $\tau_k^* = m$ and $k-1 \in \mathcal{SUB}$, we have

that Inequality (2.10) is satisfied because

$$\begin{aligned}\beta = 2 - \frac{1}{4m-2} &= \frac{3}{2} + \frac{m-1}{2 \cdot s_{r_{k-1}} \cdot (2m-1)} \cdot (2 \cdot s_{r_{k-1}}) \\ &\geq \frac{3}{2} + z_{k-1} \cdot 2 \cdot s_{r_{k-1}} \\ &\geq \frac{3}{2} + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1} \\ &\geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k}} \\ &\qquad + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1},\end{aligned}$$

where the first inequality follows from $i \leq m-1$, the second inequality follows from Lemma 2.8 and Claim 2.11, and the third inequality follows from a case distinction between $k = CR_m$ and $k \neq CR_m$ (if $k = CR_m$, then either $\tau_k = m$ or $\psi_{\tau_k}(\tau^*, k) = 0$, and if $k \neq CR_m$, then $\psi_{\tau_k^*}(\tau^*, k) \geq 1$ and the result follows from Lemma 2.2).

We conclude that the introduced dual fitting is a feasible solution for the $Dual_{sum}$ that is valid for all possible values of s, τ^*, and τ. □

Theorem 2.12. *In the case of machines having divisible speeds, $PoA(\Gamma(I)) \leq 2 - \frac{1}{2 \cdot m}$ for every instance I in class $Q||\sum C_j$, where m is the number of machines.*

Proof. As mentioned earlier, a configuration is referred to as having divisible speeds if for every pair of machines $i, i' \in M$, where $s_i \geq s_{i'}$, the remainder of the division $\frac{s_i}{s_{i'}}$ is zero. We will show that the following dual fitting constitutes a feasible solution for $Dual_{sum}$, valid for all possible values of s, τ^*, and τ, and thus yields an upper bound of $2 - \frac{1}{2 \cdot m}$ for the price of anarchy. Here $\mathcal{SUB}$ and CR_i are defined as in

the proof of Theorem 2.10.

$$\beta = 2 - \frac{1}{2 \cdot m},$$
$$y = 1,$$
$$z_j = \begin{cases} i \cdot \frac{1}{2 \cdot s_{r_j} \cdot m} & \text{, if } j \in \mathcal{SUB} \text{ and } CR_i \leq j < CR_{i+1}, i \in [m-1] \\ 0 & \text{, otherwise} \end{cases}$$

For all jobs other than those satisfying $\tau_k^* = m$ and $k - 1 \in \mathcal{SUB}$, the verification of Inequality (2.10) is identical to that in the proof of Theorem 2.10. It remains to consider a job k with $\tau_k^* = m$ and $k - 1 \in \mathcal{SUB}$. For such a job, Inequality (2.10) is satisfied because

$$\begin{aligned} \beta = 2 - \frac{1}{2m} &= \frac{3}{2} + \frac{m-1}{2 \cdot s_{r_{k-1}} \cdot m} \cdot s_{r_{k-1}} \\ &\geq \frac{3}{2} + z_{k-1} \cdot s_{r_{k-1}} \\ &= \frac{3}{2} + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1} \\ &\geq \frac{1}{\psi_{\tau_k^*}(\tau^*, k) + 1} + \frac{\psi_{\tau_k}(\tau^*, k)}{\psi_{\tau_k^*}(\tau^*, k) + 1} \cdot \frac{s_{\tau_k^*}}{s_{\tau_k}} \\ &\qquad + (z_{k-1} - z_k) \cdot \frac{s_{\tau_k^*}}{\psi_{\tau_k^*}(\tau^*, k) + 1}, \end{aligned}$$

where the first inequality follows from $i \leq m-1$, the second inequality follows from Lemma 2.8, Claim 2.11 and divisibility of speeds, and the third inequality follows from a case distinction between $k = CR_m$ and $k \neq CR_m$ (if $k = CR_m$, then either $\tau_k = m$ or $\psi_{\tau_k}(\tau^*, k) = 0$, and if $k \neq CR_m$, then $\psi_{\tau_k^*}(\tau^*, k) \geq 1$ and the result follows based on Lemma 2.2). □

2.4 A Lower Bound on PoA under the WSPT Rule

Consider the family of weighted related machine scheduling games associated with $Q||\sum w_jC_j$ under the local WSPT rule. We denote this family by Γ_w. For each instance I_w of $Q||\sum w_jC_j$, let $\Gamma_w(I_w)$ denote the game in Γ_w induced by I_w, and let $\chi(I_w)$ denote the set of all possible pure Nash equilibria of the game $\Gamma_w(I_w)$. Then the *(pure) price of anarchy* of $\Gamma_w(I_w)$ is defined as

$$PoA(\Gamma_w(I_w)) = \max_{\tau\in\chi(I_w)} \frac{SC(\tau)}{SC(\tau^*)}, \tag{2.12}$$

where $\tau^* \in M^n$ is the strategy profile that minimizes the weighted social cost for instance I_w. The pure price of anarchy for the family of related machine scheduling games Γ_w is then defined as the supremum over all instances, as follows.

$$PoA(\Gamma_w) = \sup_{I_w\in Q||\sum w_jC_j} PoA(\Gamma_w(I_w)).$$

An upper bound of 4 on the pure price of anarchy of Γ_w follows immediately from [38, 35]. Let $s_{\max}$ and $s_{\min}$ denote the speed of the fastest and slowest machines, respectively. The upper bound of 4 can be improved for some specific instances where the ratio $s_{\max}/s_{\min}$ is small [82]. This upper bound depends on three parameters: the number of machines, the speed of the fastest machine, and the speed of the slowest machine. More precisely:

$$\min\left\{\frac{s_{\max}}{s_{\min}} + \frac{1}{4},\, 4\right\} \quad \text{for } m = 2,$$

$$\min\left\{\frac{3}{2}\left(1 - \frac{1}{m}\right)\frac{s_{\max}}{s_{\min}} + \frac{1}{4},\, 4\right\} \quad \text{for } m > 2.$$

Moreover, a lower bound of 2, based on parametric instances related to those introduced by Hoeksma and Uetz [64], is presented in [82].

Similarly, in what follows, we will provide an independent derivation of the same lower bound of 2 for the pure price of anarchy of the family of the related machine scheduling games Γ_w. A parametric instance similar to the one in [64] is used to prove this bound. For the sake of completeness, we begin by introducing these parametric instances.

Instance 2.13 (Hoeksma and Uetz [64])**.** Let $\mathcal{I}$ be the set of parametric instances $I(s, m)$ that satisfy the following. $I(s, m)$ has m machines, one with the speed $s \in \mathbb{N}$, with $s > 1$, and all other machines with speed 1. We assume that $m > s$. Furthermore, $I(s, m)$ has $n = m - 1 + s$ jobs with processing requirements

$$p_j = \begin{cases} 1, & \text{if } 1 \leq j \leq s-1 \\ x^{j-s}, & \text{if } s \leq j \leq n, \end{cases}$$

where $x \coloneqq s/(s-1)$.

We refer to the machine with speed s as the *fast machine* and to the machines with speed 1 as the *slow machines*. For $s \geq 2$ and $n \geq s$, let $I_w(s, n)$ be a weighted parametric instance that has n jobs whose processing requirements are defined by the same formula used for the instances $I(s, m) \in \mathcal{I}$, one machine with speed s, and n machines with speed 1. We set $w_j = p_j$ for every job j, and denote the resulting family of weighted instances by $\mathcal{I}_w$. Following a similar argument to [64, Lemma 4], we can show that assigning all jobs to the fast machine yields a Nash equilibrium.

Lemma 2.14. *Every instance $I_w(s, n) \in \mathcal{I}_w$ has a pure Nash equilibrium with all jobs on the fast machine.*

Proof. Consider the schedule with all the jobs on the fastest machine with an SPT order. Note that since $w_j = p_j$ for all jobs, the SPT order

is consistent with the WSPT order. For the completion time of a job $j \leq s-1$, we have

$$C_j = \sum_{k=1}^{j} \frac{1}{s} = \frac{j}{s} \leq 1.$$

Moreover, the completion time for a job $j \geq s$ is equal to

$$C_j = \sum_{k=1}^{j} \frac{p_k}{s} = \frac{1}{s}((s-1) + \sum_{k=0}^{j-s} x^k) = x^{j-s} = p_j. \tag{2.13}$$

Since all the other machines have speed 1, the Nash equilibrium condition holds. □

Although assigning all jobs to the fast machine yields a pure Nash equilibrium, this assignment does not necessarily minimize the weighted social cost. For an instance $I_w(s,n) \in \mathcal{I}_w$ and an integer c, consider the assignment in which jobs $1,\ldots,c$ are assigned to distinct slow machines, while jobs $c+1,\ldots,n$ are assigned to the fast machine. We refer to c as the *cut-off* point. Since τ^* minimizes the weighted social cost, its weighted social cost is no greater than the weighted social cost of this assignment. By the preceding lemma, assigning all jobs to the fast machine yields a pure Nash equilibrium. Moreover, since $I_w(s,n)$ is one of the weighted instances over which the supremum defining $PoA(\Gamma_w)$ is taken, the ratio between the weighted social cost when all jobs are assigned to the fast machine and the weighted social cost of the assignment described above provides a lower bound on $PoA(\Gamma_w)$. In the proof below, we choose c so that this lower bound approaches 2.

Theorem 2.15. *The pure price of anarchy of the family Γ_w is at least 2.*

Proof. We let the number of jobs to be a factor of speed, $n = as$, and we define the cut-off point as $c = \lfloor (a - \ln 2)s \rfloor$, where $a \in \mathbb{N}$. Consider the instance $I_w(s,n) \in \mathcal{I}_w$. Assign the $n-c$ longest jobs to the fast machine

and the remaining jobs each to a machine with the speed of 1 (slow machine). Hence, the weighted social cost of this assignment equals

$$\begin{aligned}
\sum_{j=1}^{n} w_j C_j &= \sum_{j=1}^{s-1} 1 + \sum_{j=s}^{c} x^{2(j-s)} + \frac{1}{s} \sum_{j=c+1}^{n} x^{j-s} \sum_{k=c+1}^{j} x^{k-s} \\
&= (s-1) + \sum_{j=0}^{c-s} x^{2j} + \frac{1}{s} \sum_{j=c+1}^{n} x^{j-s} (\sum_{k=0}^{j-s} x^k - \sum_{k=0}^{c-s} x^k) \\
&= (s-1) + \sum_{j=0}^{c-s} x^{2j} + \sum_{j=c+1}^{n} x^{j-s} (x^{j-s} - x^{c-s}) \\
&= (s-1) + \sum_{j=0}^{c-s} x^{2j} + \sum_{j=c+1}^{n} x^{2(j-s)} - \sum_{j=c+1}^{n} x^{j+c-2s} \\
&= (s-1) + \sum_{j=0}^{n-s} x^{2j} - (\sum_{j=0}^{n+c-2s} x^j - \sum_{j=0}^{2c-2s} x^j) \\
&= (s-1) - \frac{(s-1)^2}{2s-1} + \frac{s^2}{2s-1} x^{2(n-s)} - s(x^{n+c-2s} - x^{2c-2s}).
\end{aligned}$$

Next, we compute the weighted social cost when all jobs are assigned to the fast machine. By using Equation 2.13, we have

$$\begin{aligned}
\sum_{j=1}^{n} w_j C_j &= \sum_{j=1}^{s-1} \frac{j}{s} + \sum_{j=s}^{n} x^{2(j-s)} = \frac{(s-1)}{2} + \sum_{j=0}^{n-s} x^{2j} \\
&= \frac{(s-1)}{2} + (\frac{(s-1)^2}{2s-1})(((\frac{s}{s-1})^2)^{n-s+1} - 1) \\
&= \frac{(s-1)}{2(2s-1)} + (\frac{s^2}{2s-1}) x^{2(n-s)}.
\end{aligned}$$

Using the obtained social costs, we can get the following lower bound

$$PoA(\Gamma_w(I_w(s,n))) = \frac{\left(\frac{s^2}{2s-1}\right) x^{2(n-s)} + \frac{s-1}{2(2s-1)}}{\left(\frac{s^2}{2s-1}\right) x^{2(n-s)} - s\left(x^{n+c-2s} - x^{2c-2s}\right) - \frac{(s-1)^2}{2s-1} + (s-1)}.$$

By the definition of c, we can write, $c = (a - \ln 2)s + \epsilon$ with $\epsilon \in (-1, 0]$. Note that $\lim_{s\to\infty} x^s = e$. Moreover, since ϵ is bounded ($\epsilon \in (-1, 0]$) and $x \to 1$ as $s \to \infty$, we have $x^\epsilon \to 1$. Hence

$$\begin{aligned}
\lim_{s\to\infty} PoA(\Gamma_w(I_w(s,n))) &= \frac{e^{2(a-1)}}{e^{2(a-1)} - 2(e^{(2a-\ln 2-2)} - e^{(2a-2\ln 2-2)}) + 1} \\
&= \frac{e^{2a}}{e^{2a} - 2(e^{(2a-\ln 2)} - e^{(2a-2\ln 2)}) + e^2} \\
&= \frac{e^{2a}}{\frac{1}{2}e^{2a} + e^2}.
\end{aligned}$$

Now, by letting the number of jobs n (and consequently a) go to infinity we have

$$PoA(\Gamma_w) \geq \lim_{a\to\infty} PoA(\Gamma_w(I_w(s,n))) = 2.$$

□

2.5 Concluding Remarks

In this chapter, we derive an upper bound of $2 - (1/(4m-2))$ on the approximation factor of the Ibarra–Kim algorithm for scheduling on related machines with the objective of minimizing the total completion time. This bound is strictly below 2 for every fixed number of machines m, and we obtain a tighter bound of $3/2$ in the special case of two machines. These same bounds apply as improved upper bounds on the pure price of anarchy for the corresponding decentralized scheduling game, in which each job selfishly chooses a machine and machines schedule jobs according to the SPT rule. To our knowledge,

these are the first bounds for this problem that (i) are strictly below 2, and (ii) depend only on the number of machines, rather than also on the number of jobs.

Our analysis is motivated by the gap in the pure price of anarchy between the lower bound example, with a price of anarchy of $e\,/\,(e-1)$, and an upper bound of 2, as shown by Hoeksma and Uetz [64]. Their analysis uses *semi-smoothness*, a variant of the smoothness framework that yields bounds which hold not only for pure Nash equilibria, but also for mixed Nash equilibria and, more generally, for *coarse-correlated* equilibria [94]. A coarse-correlated equilibrium is a probability distribution over strategy profiles such that no player can reduce its expected completion time by committing to a fixed unilateral deviation before the realization of the random outcome. Consequently, the upper bound of 2 in [64] applies to both the pure and the mixed price of anarchy. For identical machines (i.e., all speeds equal), the situation is better understood. In that case, the pure price of anarchy is 1, since SPT yields an optimal schedule on each machine [36]. However, the "robust" price of anarchy, i.e., the worst-case ratio that holds for mixed Nash equilibria and coarse-correlated equilibria, equals $3/2 - 1/(2m)$, and this value is tight for mixed Nash equilibria [91, 64]. This shows that for identical machines, there is a separation between pure equilibria and more general equilibrium concepts. If a similar separation also occurs for related machines (i.e., machines with different speeds), then a smoothness or semi-smoothness argument, which is designed to be robust to randomization, cannot be expected to give an upper bound on the pure price of anarchy. This necessitates alternative techniques that explicitly exploit specific properties of pure Nash equilibria.

We address this using a primal-dual technique. We construct a feasible solution to the dual of a linear program that provides an upper bound for the worst-case ratio between the total completion time at a pure Nash equilibrium and the optimal total completion time. The main idea of our analysis is to consider a relaxation of the LP_{weak}. Rather than imposing deviation constraints for every individual job, we replace them with a single aggregated deviation inequality obtained by

summing all such constraints. Although this weakens the formulation, it simplifies the structure of the dual program and allows us to construct improved feasible dual solutions yielding better upper bounds. The construction relies on two ingredients. First, we work with an aggregated deviation inequality that captures, in a single constraint, that no job would strictly improve its completion time by moving to its "optimal" machine (the machine in the optimal strategy). Second, we assign dual multipliers in such a way (in particular, by fixing the deviation multiplier $y = 1$) that the equilibrium-dependent terms cancel, leaving constraints that can be verified using only structural properties of an optimal schedule. We believe that further progress on the pure price of anarchy for related machine scheduling can be made within the same primal-dual paradigm along several directions:

- **Stronger deviations.** The linear program we analyze does not encode the full set of best-response conditions of a Nash equilibrium. Instead, it uses a weaker deviation condition: in the strategy profile τ, no job can reduce its completion time by moving to its "optimal" machine. This is why we refer to the underlying formulation as LP_{weak}. A natural strengthening is to impose a richer family of deviation constraints, for example, that no job can reduce its completion time by deviating to any machine, or at least to a particular set of machines. Such a strengthening would yield a stronger primal upper bound on the pure price of anarchy, and therefore, in principle, can only improve the dual-fitting bound.

- **Alternative aggregation of deviation constraints.** In our analysis, all deviation constraints from LP_{weak} are summed to obtain a single aggregated inequality, yielding LP_{sum}, and we then construct a feasible dual solution for the dual of that aggregated formulation.

  The choice of this aggregation is motivated by the semi-smoothness analysis of this scheduling game by Hoeksma and Uetz [64]. Every Nash equilibrium satisfies one inequality for each job and each unilateral deviation. In the

semi-smoothness analysis of this game, the deviation used for each job is its "optimal" machine. The costs resulting from these deviations are then summed, and their total is bounded using structural properties of an optimal schedule. This motivates us to sum the corresponding individual deviation constraints and work with a single aggregated constraint. Although this gives a weaker primal formulation, it simplifies the corresponding dual by replacing the individual deviation multipliers with a single multiplier.

As a result of this aggregation, $Dual_{sum}$ contains only one dual variable associated with the aggregated deviation constraint. This makes the dual fitting tractable, but it also discards the distinction between the individual deviations. This motivates considering other aggregations of the deviation constraints, for example, per-machine or per-prefix aggregations. Such aggregations would give separate dual variables for the corresponding aggregated inequalities and may allow the equilibrium-dependent terms to be canceled or bounded more tightly, thereby improving the upper bound.

- **Generalizing the dual fitting beyond y = 1.** In our construction, the dual variable corresponding to the deviation constraint is fixed to $y = 1$. This choice is crucial because it causes the equilibrium-dependent terms on both sides of the dual constraint to cancel, so that feasibility can be certified using only properties of the optimal schedule and not of the equilibrium schedule. This makes the proof tractable. A possible direction for improvement is to allow non-uniform or more refined multipliers instead of the single scalar $y = 1$, with the aim of extracting more information from the equilibrium structure without losing feasibility. Such refinements could potentially lower the resulting value of β and thus reduce the remaining gap $[e/(e-1), 2]$.

Overall, this chapter shows that the primal-dual technique can use structural information that is not retained in the semi-smoothness

framework. The semi-smoothness proof of Hoeksma and Uetz [64] also uses properties of optimal schedules, but only through the positional value comparison derived from the MFT algorithm. After the deviation inequalities are summed, the analysis no longer distinguishes how the individual jobs appear in the optimal schedule. In contrast, our LP formulation and the corresponding dual fitting keep this information available through the constraints and the dual variables. This allows the proof to use more of the structure of the optimal schedule and yields a better non-asymptotic guarantee on the pure price of anarchy in decentralized related machine scheduling. We believe that the upper bound of two can be asymptotically improved to $e/(e-1)$.

Chapter 3

Fixed Order Scheduling with Deadlines

3.1 Introduction

In scheduling problems, given a set of jobs J, where each job $j \in J$ has a release date r_j, a processing time p_j, and a deadline d_j, along with a set of machines, the objective is to determine a feasible assignment of jobs to machines that optimizes the objective of the problem while respecting the constraints on the details of the job and the environment of the machines. In the literature, various objectives, such as makespan, completion time, and the number of late jobs are examined. Among these objectives, *minimizing the number of machines* is very important in practice.

Consider a scenario in which multiple users use parallel processors with shared memory. By scheduling jobs on a certain subset of processors, other processors can be put to sleep, thereby reducing energy consumption. The goal is to minimize the cost, specifically the number of processors in use. Another application arises in the management of energy storage systems for microgrids under emergency conditions. In such scenarios, backup power sources such

Parts of this chapter are adapted from: Andre Berger, Arman Rouhani, and Marc Schröder. "Fixed order scheduling with deadlines". In: *Operations Research Letters* 61 (2025). DOI: 10.1016/j.orl.2025.107306.

as diesel generators are responsible for supplying energy to critical operations. Each generator has a limited capacity and is only activated when needed. Scheduling emergency jobs efficiently minimizes the number of active generators, reducing resource consumption and extending operational availability during crises [86].

In numerous practical applications such as airport gate management [76], train scheduling [46], the transportation of navy fuel oil tankers [40], developing distributed memory architectures [81], runway scheduling, and scheduling of maintenance work for trains in a service station, the processing of a set of jobs on a minimum number of machines is desired while the schedule respects the time intervals in which jobs have to be scheduled. These problems are referred to as *SRDM* (scheduling with release times and deadlines on a minimum number of machines [31]), and in the scheduling notation $\alpha|\beta|\gamma$ introduced by [55], it is represented as $P|r_j, p_j, d_j|m$. For the special case of the problem where the jobs have a common release time and equal deadlines, the problem is reduced to the well-known Bin Packing Problem (BPP), which is known to be NP-hard [34]. The general problem can be considered a BPP with additional constraints. Specifically, when $d_j = r_j + p_j$ for each job $j \in J$, each job occupies its entire time window, i.e., the interval $[r_j, d_j]$. This is exactly the Interval Partition Problem (also referred to as the Interval Coloring Problem), which can be solved in polynomial time [53]. This problem is a special case of the BPP with conflicts [43, 88, 98]. It is shown that when $d_j \leq r_j + p_j + 1$, the SRDM problem can be solved in polynomial time; otherwise, it remains NP-complete [31]. For the common release time variant of the problem denoted as $P|r_j = r, p_j, d_j|m$, a 2-approximation algorithm is provided [111]. Also, it is proved that the problem of $P|r_j, p_j = p, d_j|m$ admits a 6-approximation algorithm [111]. This result was later improved, proving that the equal processing times variant of the problem is polynomially solvable [75].

In various practical applications such as CPU task management, perishable goods handling, airline boarding, and healthcare systems, there is often a requirement to follow an *order* on each machine

when scheduling tasks. This fixed order is distinct from precedence constraints, as it requires tasks to be processed in a specific sequence on each processor, without dependencies between the tasks themselves. For example, in CPU scheduling with static priorities, tasks must be executed in a pre-established sequence across processors, following global priority rules [23]. Similarly, in industries dealing with perishable goods, warehouses or delivery systems must follow a predefined order to ensure items are processed and delivered before their expiration [83]. In airline operations, passengers are boarded in a fixed sequence across gates, ensuring smooth and efficient boarding procedures [12]. In healthcare systems, patient appointments are frequently managed in a fixed sequence, with doctors treating patients in the order determined by their schedule, independent of other appointments [57]. These examples demonstrate the importance of efficient scheduling under a fixed global order. This variant of the problem recently gained attention for different objectives [19, 107].

In this paper, we address the SRDM problem with common release times and a fixed processing order, denoted as $P|r_j = r, p_j, d_j, \pi^*|m$. Since the release times are identical, an appropriate translation of the time axis yields an equivalent formulation with zero release time. We investigate the impact of the imposed processing order on job scheduling on a minimum number of machines. We first study the approximation factors of two simple greedy heuristics for the problem: first-fit and next-fit. Then, we provide a greedy approximation algorithm for the problem that fits the largest number of unscheduled jobs on a machine using a dynamic programming approach.

Section 3.2 formally defines the problem and introduces the notation used in the paper. In Section 3.3, we show that the first-fit algorithm optimally solves the problem for the case of unit processing times, and it is a 2-approximation for several variants of the problem. Finally, we give an $O(\log n)$ approximation algorithm for the general problem.

3.2 Preliminaries

In this study, we focus on determining the minimum number of machines required to schedule a given set of jobs. The input to the problem consists of n identical machines and a set of n jobs denoted as $J = \{1, \ldots, n\}$. Each job $j \in J$ is characterized by a pair $(p_j, d_j) \in \mathbb{N}^2$ with processing time $p_j > 0$, reflecting the time required for completion, and deadline $d_j \geq p_j$. For each job $j \in J$, we define $\lambda_j = d_j - p_j$ as the *slack* representing the difference between the job's deadline and its processing time, which is non-negative. Let $T = \{(p_j, d_j) \mid j \in J\}$ be the set of tuples representing the processing time and deadline for each job $j \in J$. The jobs follow a predetermined fixed order for processing, and each machine schedules its assigned jobs according to this order. W.l.o.g., we assume that job 1 has the highest priority, job 2 has the second-highest priority, and job n has the lowest priority, i.e., $j < k$ means that j has a higher priority than k. The objective of the problem is to minimize the number of machines required to schedule the jobs. A feasible solution corresponds to an assignment of jobs to machines such that the prescribed order is maintained on each machine, jobs are processed non-preemptively, i.e., if the processing of a job j starts at time t on a machine, it completes at time $t + p_j$ on the same machine, and every job is completed within its stipulated deadline. Since all jobs have a common release time, we assume, without loss of generality, that this release time is 0. Furthermore, schedules contain no idle times between consecutive jobs on the same machine, as delaying a job cannot improve feasibility. In addition, each machine can process at most one job at a time. For simplicity, we use the notation $[q] = \{1, \ldots, q\}$ for any $q \in \mathbb{N}$.

An optimal solution (*OPT*) is a schedule $\tau^* : J \to [n]$ that is feasible, i.e., $\sum_{k \leq j : \tau^*(k) = \tau^*(j)} p_k \leq d_j$ for all $j \in J$, and minimizes the number of *open* machines. A machine is open if it has been assigned at least one job. Let $\mathcal{M}^*$ denote the set of open machines in τ^*. For an optimal schedule, define $J_i^* = \{j \in J \mid \tau^*(j) = i\}$ as the set of jobs assigned to machine $i \in \mathcal{M}^*$, and $L_i^* = \sum_{j \in J_i^*} p_j$ as the load of machine $i \in \mathcal{M}^*$.

Moreover, let l_i^* denote the last job scheduled on machine $i \in \mathcal{M}^*$. We will study two simple greedy algorithms. The *first-fit (FF)* algorithm (Algorithm 5) schedules jobs based on the fixed order and assigns each job to the first feasible machine.

Algorithm 5 First-fit Algorithm

1: **for** $j = 1$ to n **do**
2: Assign job j to the smallest-indexed machine i with $L_i + p_j \leq d_j$.
3: **end for**

An even simpler greedy algorithm *next-fit* (NF) operates similarly, with the exception that NF (Algorithm 6) only considers the last open machine (while FF considers all open machines) for scheduling a job and, if that job fails to fit in the last machine, NF uses a new empty machine for scheduling. Given some approximation algorithm with

Algorithm 6 Next-fit Algorithm

1: Set $i = 1$ and $L_i = 0$
2: **for** $j = 1$ to n **do**
3: **if** $L_i + p_j \leq d_j$ **then**
4: Assign job j to machine i
5: $L_i = L_i + p_j$
6: **else**
7: Assign job j to machine $i + 1$
8: $i = i + 1$, $L_i = p_j$
9: **end if**
10: **end for**

schedule $\tau : J \to [n]$, we define $J_i = \{j \in J \mid \tau(j) = i\}$ as the set of jobs assigned to machine i. Furthermore, let $L_i = \sum_{j \in J_i} p_j$ represent the load of machine i. Let $OPT(I)$ denote the number of open machines of an optimal solution and $ALG(I)$ denote the number of open machines of the algorithm for a given instance I. The approximation ratio of an algorithm for a class of instances is defined as $\sup_I ALG(I)/OPT(I)$.

In this investigation, we will study the approximation ratio of first-fit and next-fit for the fixed order scheduling problem.

3.2.1 Optimal Schedule

To gain deeper insights into the problem, we present an integer linear program (ILP) formulation for the optimal schedule. The ILP takes as input the set T, which comprises tuples representing the processing time and deadline for each job $j \in J$. Additionally, let M represent the set of n identical machines, where $|M| = |T|$. The ILP can be represented as follows

$$
\begin{aligned}
ILP(T) = \min \sum_{i \in M} y_i & \\
\sum_{i \in M} x_{ij} = 1, & \qquad \forall j \in J, \qquad (3.1)\\
\sum_{j \in J} x_{ij} \leq n \cdot y_i, & \qquad \forall i \in M, \qquad (3.2)\\
\sum_{k \leq j} p_k \cdot z_{kj} \leq d_j, & \qquad \forall j \in J, \qquad (3.3)\\
x_{ik} + x_{ij} - 1 \leq z_{kj}, & \qquad \forall i \in M,\ \forall j, k \in J. \qquad (3.4)
\end{aligned}
$$

where

$$
y_i = \begin{cases} 1, & \text{if machine } i \text{ is open,} \\ 0, & \text{otherwise} \end{cases}
$$

$$
x_{ij} = \begin{cases} 1, & \text{if job } j \text{ is scheduled on machine } i, \\ 0, & \text{otherwise} \end{cases}
$$

$$
z_{kj} = \begin{cases} 1, & \text{if } k \text{ and } j \text{ are scheduled on the same machine,} \\ 0, & \text{otherwise} \end{cases}
$$

Equation (3.1) ensures that each job is assigned to exactly one machine. Equation (3.2) limits job assignments so that only active machines can receive jobs, with the limit n large enough to avoid restricting capacity. Equation (3.3) ensures that the completion time of each job j is no greater than its deadline, where the completion time of j includes the processing times of all jobs on the same machine that have a priority equal to or higher than j. Equation (3.4) ensures that if two jobs k and j are assigned to the same machine, this is recorded in z_{kj} so that Equation (3.3) can correctly calculate the total processing time according to the fixed global order. It should be noted that this ILP formulation has $2n^2+n$ variables and n^3+3n constraints. The problem we are studying is NP-hard since it is at least as hard as BPP. This means no polynomial-time algorithm can solve this problem optimally unless $P = NP$. We aim to find how well the optimal answer can be approximated using simple polynomial-time algorithms.

3.3 Analysis of Approximation Factors

In this section, we provide a comprehensive analysis of our problem. Initially, we show that next-fit does not admit a constant approximation factor, neither for the general setting nor for the special case of unit processing times. However, we establish that first-fit is actually optimal in the case of unit processing times. Then, we prove that first-fit has an approximation factor of 2 in four different classes of instances: (1) if the fixed order aligns with slacks that are in a non-increasing order, i.e., $d_j - p_j \geq d_{j+1} - p_{j+1}$ for $j \in [n-1]$, (2) if the fixed order aligns with slacks that are in a non-decreasing order, i.e., $d_j - p_j \leq d_{j+1} - p_{j+1}$ for $j \in [n-1]$, (3) if the fixed order aligns with deadlines that are in a non-increasing order, i.e., $d_j \geq d_{j+1}$ for $j \in [n-1]$, and (4) if the optimal solution uses at most 3 machines, i.e., $|\mathcal{M}^*| \leq 3$. Finally, using a dynamic programming approach, we prove that there is an $O(\log n)$ approximation factor for the general fixed order.

Instance 3.1. We construct a parametric instance I_n with $n > 2$ jobs as follows. Assume that $(p_1, d_1) = (1, 1)$ and $(p_2, d_2) = (2, 2)$. For $j > 2$,

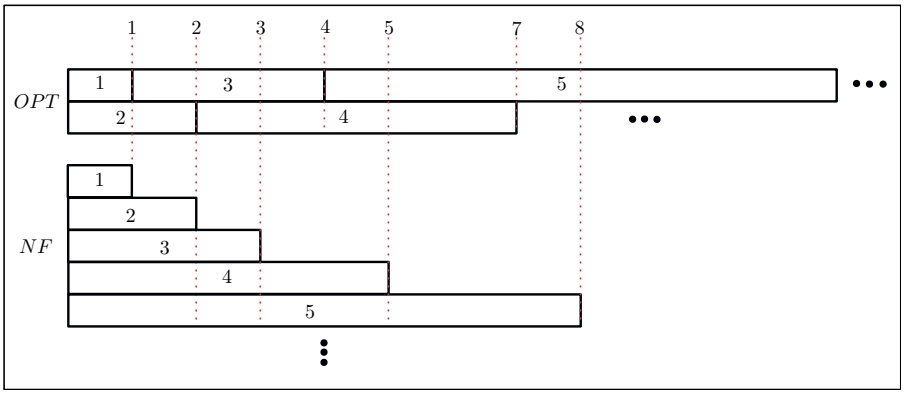


Figure 3.1: The optimal schedule and the schedule by NF for I_n where $n \geq 5$.

the pair (p_j, d_j) is obtained by $(p_{j-1} + p_{j-2}, p_j + p_{j-1} - 1)$.

Lemma 3.2. *The approximation ratio of the next-fit algorithm is at least $n/2$.*

Proof. Consider Instance 3.1. The NF algorithm consistently selects the machine that has opened most recently to place a new job j. When applied to I_n, the NF algorithm initiates the opening of a new machine for each job $j \in J$ since the deadline of the job j is equal to $p_j + p_{j-1} - 1$. However, the optimal schedule requires only two machines to schedule all the jobs. Jobs with odd indices are scheduled on one machine, and jobs with even indices are scheduled on the other machine (see Figure 3.1 for an illustration). Hence, the number of machines in OPT is always 2, and the number of machines in the NF algorithm is equal to n. Thus, $NF(I_n)/OPT(I_n) = n/2$, and the lemma holds. □

Lemma 3.3. *The next-fit algorithm is an $n/2$-approximation algorithm.*

Proof. Let I be an arbitrary instance with n jobs. Consider two cases. If $OPT(I) = 1$, all jobs can be feasibly scheduled on a single machine in the given fixed order. Since NF always attempts to place a job j on the

most recently opened machine before opening a new one, it will also schedule all jobs on the first machine. Thus, $NF(I) = OPT(I) = 1$. If $OPT(I) \geq 2$, then NF opens at most one new machine per job, so $NF(I) \leq n$. So, $NF(I)/OPT(I) \leq n/2$. In both cases, the ratio is at most $n/2$. Thus, the lemma holds. □

By Lemmas 3.3 and 3.2, we conclude that the approximation ratio of the next-fit algorithm on instances with $n > 2$ jobs is exactly $n/2$.

3.3.1 Unit Processing Times

In this subsection, we consider the case of unit processing times. In the scheduling notation, this problem is represented as $P|r_j = r, p_j = 1, d_j, \pi^*|m$. We first construct an instance that shows the NF algorithm does not admit a constant approximation factor, and then we prove that the FF algorithm solves $P|r_j = r, p_j = 1, d_j, \pi^*|m$ optimally.

Instance 3.4. Let I'_n be a parametric instance consisting of $n \geq 2$ jobs. For all $j \in [n]$, let $p_j = 1$, $r_j = 0$, and $d_j = \lceil j/2 \rceil$.

Lemma 3.5. *The approximation ratio of the next-fit algorithm for the case of unit processing times is at least* $(\lfloor \log_2 n \rfloor + 1)/2$.

Proof. Consider Instance 3.4. We first show that OPT$(I'_n) = 2$. Jobs 1 and 2 have deadlines $d_1 = d_2 = 1$. Since fixed order requires job 2 to follow job 1 on any machine they share, placing both on the same machine results in job 2 being completed at time $2 > d_2$. Hence, there is no feasible single-machine schedule. So, OPT$(I'_n) \geq 2$. On the other hand, two machines suffice: schedule all odd-indexed jobs on machine 1 and all even-indexed jobs on machine 2. Job j is the $\lceil j/2 \rceil$-th job on its machine and completes at time $\lceil j/2 \rceil = d_j$, so this schedule is feasible. Hence OPT$(I'_n) = 2$.

We now analyze the schedule produced by NF. Let $q = \lfloor \log_2 n \rfloor$, so that $2^q \leq n < 2^{q+1}$. We prove by induction on r that NF opens a new machine when it schedules job 2^r, for every $r \in \{0, \ldots, q\}$.

Base Case: NF opens the first machine for job $1 = 2^0$.

Induction Hypothesis: Assume that NF opens a new machine when it schedules job 2^r, for some $r \in \{0, \ldots, q-1\}$.

Induction Step: We show that NF opens a new machine when it schedules job 2^{r+1}. Since $r < q$, we have $2^{r+1} \leq n$, so job 2^{r+1} exists in the instance. After NF opens a machine for job 2^r, consider any job $j = 2^r + t$ with $0 \leq t \leq 2^r - 1$. Before NF processes job j, the most recently opened machine contains exactly t jobs (namely jobs $2^r, \ldots, 2^r + t - 1$), so assigning job j to this machine would result in a completion time of $t + 1$. Since $t \leq 2^r - 1$, we have $2^r + t \geq 2t + 1$, and therefore $\lceil (2^r + t)/2 \rceil \geq t + 1 = d_j$. Hence, job j fits on the most recently opened machine. Consequently, before NF processes job 2^{r+1}, the most recently opened machine contains exactly 2^r jobs. Assigning job 2^{r+1} to this machine would result in a completion time of $2^r + 1$. Since $d_{2^{r+1}} = \lceil 2^{r+1}/2 \rceil = 2^r$, job 2^{r+1} does not fit, and NF opens a new machine for it. By induction, NF opens a new machine for each job in $\{2^0, 2^1, \ldots, 2^q\}$. Therefore, $\mathrm{NF}(I'_n) \geq \lfloor \log_2 n \rfloor + 1$, and together with $\mathrm{OPT}(I'_n) = 2$ this gives

$$\frac{\mathrm{NF}(I'_n)}{\mathrm{OPT}(I'_n)} \geq \frac{\lfloor \log_2 n \rfloor + 1}{2}. \qquad \square$$

Theorem 3.6. *The FF algorithm solves the problem $P|r_j = r, p_j = 1, d_j, \pi^*|m$ optimally in time $O(n^2)$.*

Proof. It is easy to verify that the worst-case running time of the FF algorithm is $O(n^2)$. Therefore, it only remains to prove that the FF algorithm provides an optimal schedule for $P|r_j = r, p_j = 1, d_j, \pi^*|m$.

Denote by τ the schedule produced by the FF algorithm. Also, define $J_k^{\pi^*} = \{1, \ldots, k\}$ and let τ_k be the schedule produced by the FF algorithm for jobs $J_k^{\pi^*}$. Moreover, we call the position of a job j on a machine its *spot* s, i.e., the spot of job $j \in J$ is $s(j) = |\{k \in J \mid k \leq j \text{ and } \tau(k) = \tau(j)\}|$.

Claim 3.7. *If the jobs have unit processing times, then applying the FF algorithm yields*

$$L_1 \geq L_2 \geq \ldots \geq L_n. \tag{3.5}$$

Proof. Assume by contradiction that the load of a machine i is less than the load of a machine $i^{'}$, where $i < i^{'}$. Now, consider the iteration of the FF algorithm when $L_{i'}$ becomes larger than L_i which means $L_{i'} = L_i + 1$. Observe that since the processing times are unit, before the iteration, both of the machines have equal loads. This means that since the job fits in $i^{'}$, it could have fit in machine i. This is a contradiction to the procedure of the FF algorithm since it fits a job in a machine with the lowest possible index. □

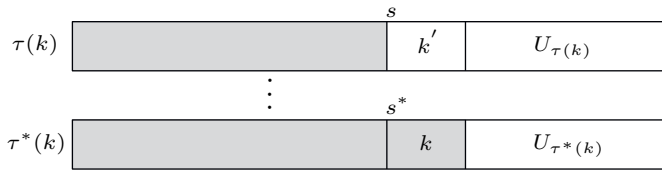
Claim 3.8. *There exists an optimal solution for the problem $P|r_j = r, p_j = 1, d_j, \pi^*|m$ where the assignment of jobs of $J_k^{\pi^*}$ are the same as their assignment in τ_k, for all $1 \leq k \leq n$.*

Proof. The proof is done by induction on k.

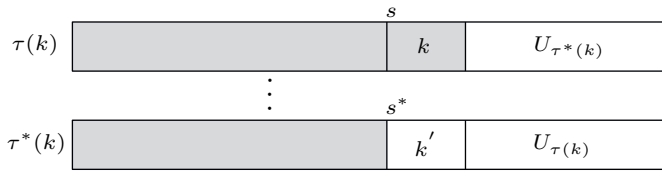
Base Case: By the assignment of the first job, the load of one machine becomes one, and the rest have a load of zero, regardless of which machine the job is assigned to. Therefore, it is easy to verify that the base case is valid.

Induction Hypothesis: We assume there exists an optimal schedule τ^* such that the assignment of each $j \in J_{k-1}^{\pi^*}$ is the same as in τ.

Induction Step: We prove that there exists an optimal schedule τ^{**} such that the assignment of job k is the same as in τ. Therefore, we modify the schedule of τ^* that agrees with the schedule of τ for every $j \in J_{k-1}^{\pi^*}$ in a way that both τ and τ^{**} have the same placement for job k. Let $\tau(k)$ and $\tau^*(k)$ be the machines that job k is scheduled on in the FF algorithm and the optimal schedule, respectively. If $\tau(k) = \tau^*(k)$, then we are done because we can define $\tau^{**} = \tau^*$. So, we can assume that $\tau(k) \neq \tau^*(k)$. Because the FF algorithm assigns the jobs to the first open machine, we have $\tau(k) < \tau^*(k)$. Assume that job k is assigned



(a) Assignment of jobs before modification.



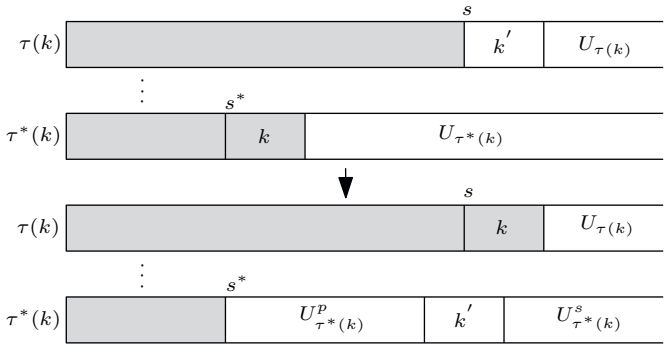
(b) Assignment of jobs after modification.

Figure 3.2: Exchanging the jobs between two machines of τ^* when $s = s^*$.

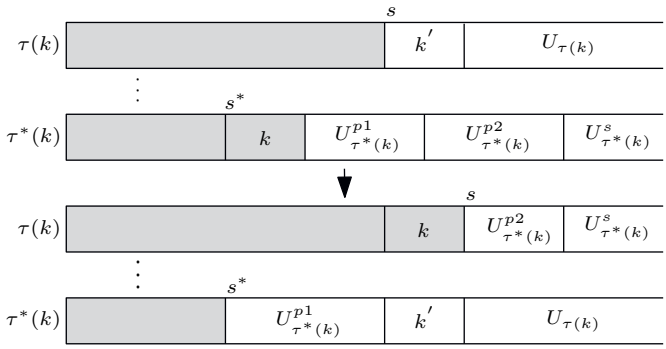
to spot s on machine $\tau(k)$, and to spot s^* on machine $\tau^*(k)$. Since the assignment of each $j \in J^{\pi^*}_{k-1}$ is the same as in τ and $\tau(k) < \tau^*(k)$, we have by Claim 3.7 that $s \geq s^*$. Moreover, let $U_{\tau(k)}$ be the set of jobs $j \in J$ where $\tau^*(j) = \tau(k)$ and $s(j) > s$ (note that these jobs have lower priority than job k), and denote by $U_{\tau^*(k)}$ as the set of jobs $j \in J$ with $\tau^*(j) = \tau^*(k)$ and $s(j) > s^*$. Let also k' be the job that is assigned to the spot s of the machine $\tau(k)$ in τ^* (note that this job need not exist).

First, assume that $s^* = s$. In this case, define τ^{**} as τ^* subject to the following change: process jobs in $U_{\tau(k)}$ on $\tau^*(k)$ and jobs in $U_{\tau^*(k)}$ on $\tau(k)$. By doing so, each job keeps its previous spot in the new machine, and the order is not violated. Then, swap jobs k and k' so that job k is assigned to the machine $\tau(k)$ in τ^{**} (see Figure 3.2 for an illustration). This means there exists an optimal schedule τ^{**} that has the same assignment for job k as in τ.

Next, assume that $s^* < s$. We define τ^{**} by modifying τ^* in a way that it has the same placement for job k as in τ. If no job is scheduled in spot s of $\tau(k)$, then job k can be relocated to this machine in spot s, and we are done. Denote by $U^p_{\tau^*(k)}$ and $U^s_{\tau^*(k)}$ the set of jobs in $U_{\tau^*(k)}$ that



(a) Assignment of jobs for $|U^p_{\tau^*(k)}| \leq s - s^*$.



(b) Assignment of jobs for $|U^p_{\tau^*(k)}| > s - s^*$.

Figure 3.3: Exchanging the jobs between two machines of τ^* when $s > s^*$.

precede and succeed job k' in π^*. Since job k can be assigned to machine $\tau(k)$ in spot s in τ, and based on the induction hypothesis, changing the machine of job k in τ^* does not violate its deadline. Moreover, the completion times of jobs in $U_{\tau^*(k)}$ are reduced. Based on the difference between s and s^*, we analyze two different cases

Case (I) **($|U^p_{\tau^*(k)}| \leq s - s^*$):** In this case, define τ^{**} so that the assignments of jobs k and k' are switched in a way that job k is scheduled in spot s of $\tau(k)$ and job k' is scheduled after $U^p_{\tau^*(k)}$ in the machine $\tau^*(k)$ (see Figure

3.3a for an illustration). It is easy to confirm that this modification violates no deadlines and preserves the order. Only the completion time of job k increases, and since it is assigned to spot s of machine $\tau(k)$ in τ, this swap does not violate its deadline. It is also worth mentioning that since $|U^p_{\tau^*(k)}| \leq s - s^*$, the deadline of job k' is not violated either. Thus, the proof holds.

Case (II) **($|U^p_{\tau^*(k)}| > s - s^*$):** In previous case, since $|U^p_{\tau^*(k)}| \leq s - s^*$, switching the machine of k' to $\tau^*(k)$ does not violate the deadline of k'. However, this becomes an issue in the current case. To solve this issue, we divide the jobs in $U^p_{\tau^*(k)}$ into two sets $U^{p1}_{\tau^*(k)}$ and $U^{p2}_{\tau^*(k)}$ where the former set consists of the first $s - s^*$ jobs of $U^p_{\tau^*(k)}$ and the latter set contains the rest. In the modified schedule τ^{**} (refer to Figure 3.3b for an illustration) job k is assigned to spot s of $\tau(k)$ and job k' is assigned to spot s of $\tau^*(k)$ while their orders and completion times are preserved. Jobs of $U^{p1}_{\tau^*(k)}$ are placed in the spots s^* to $s-1$ of machine $\tau^*(k)$, where their completion times are reduced, and their orders are preserved. Jobs belonging to $U^{p2}_{\tau^*(k)}$ and $U_{\tau(k)}$ are assigned after spot s in $\tau(k)$, as in the original configuration (before modification). Also, it is easy to verify that jobs $U^s_{\tau^*(k)}$ preserve their order after modification and their deadlines are not violated. □

Hence, FF provides an optimal schedule for $P|r_j = r, p_j = 1, d_j, \pi^*|m$. □

Remark 3.9. In the special case of unit processing times, one might be tempted to determine the minimum number of machines based on

$$\left\lceil \max_{k \geq 1} \frac{|D_k|}{k} \right\rceil, \tag{3.6}$$

where $D_k = \{j \in J : d_j \leq k\}$. This is solely based on the deadlines and yields the optimal number of machines when jobs can be scheduled arbitrarily. However, the presence of a fixed order may force a

scheduling algorithm to use more machines. Example 3.1 shows that the expression 3.6 does not give the optimal number of machines, given that the fixed order has to be taken into account.

Example 3.1. *Consider the following instance with 6 jobs*

$$(p_1, d_1) = (p_2, d_2) = (1, 3),$$
$$(p_3, d_3) = (p_4, d_4) = (p_5, d_5) = (p_6, d_6) = (1, 2),$$

and assume the fixed order is $1 < 2 < 3 < 4 < 5 < 6$*. We have*

$$\left\lceil \max_{k \geq 1} \frac{|D_k|}{k} \right\rceil = \left\lceil \frac{|\{1, 2, 3, 4, 5, 6\}|}{3} \right\rceil = 2.$$

In contrast, FF (the optimal algorithm based on Theorem 3.6) produces the assignment of $\tau^*(1) = \tau^*(2) = 1, \tau^*(3) = \tau^*(4) = 2, \tau^*(5) = \tau^*(6) = 3$ *which uses 3 machines. This shows that when the fixed processing order is enforced, the simple deadline-based measure underestimates the number of machines required.*

3.3.2 Non-Increasing Slacks

In this section, we first prove that when the fixed order aligns with non-increasing slacks, the output of the next-fit algorithm is identical to the first-fit algorithm. In this case, the approximation ratio of both algorithms is equal to 2. Furthermore, we substantiate this claim by illustrating an example that confirms the 2-approximation ratio and shows the tightness of the result.

Lemma 3.10. *For the instances with non-increasing slacks, the output of the next-fit algorithm is identical to that of the first-fit algorithm.*

Proof. Consider an iteration of the first-fit algorithm where k machines are already opened, and let $j \in J$ be the job at the beginning of the

sorted list of jobs. Then, if job j cannot fit in machine i, for $i \in [k]$ we have

$$L_i + p_j > d_j \Rightarrow L_i > d_j - p_j. \tag{3.7}$$

Given the assumption that the order is established by non-increasing slacks, for all $q \in J$ with $j < q$, the inequality $d_j - p_j \geq d_q - p_q$ holds. Utilizing this inequality together with Inequality (3.7), we have $L_i + p_q > d_q$, for all $i \in [k]$. This implies that while the FF algorithm considers machines 1 to k, if job j cannot fit into any of those machines, then none of the jobs q with $j < q$ can be accommodated in those machines either. This scenario is equivalent to exclusively considering the last machine. Therefore, the lemma holds. □

Remark 3.11. The problem $P|r_j = r, p_j, d_j, \pi^*|m$ is NP-hard when the fixed order aligns with slacks that are in a non-increasing order, i.e., $d_j - p_j \geq d_{j+1} - p_{j+1}$ for $j \in [n-1]$.

Proof. Consider an instance of the decision version of Bin Packing Problem as given in Definition 1.1. Change the label of the items so that $s(a_1) \leq \ldots \leq s(a_n)$. Construct a scheduling instance with n jobs each with $p_j = s(a_j)$ and deadline $d_j = B$. Since $\lambda_j = B - s(a_j)$, the slacks satisfy $\lambda_1 \geq \cdots \geq \lambda_n$, so the order matches with the non-increasing order of slacks. It remains to show that the scheduling instance admits a feasible schedule on at most K machines if and only if the bin-packing instance can be packed into at most K bins of capacity B. Packing the items into at most K bins of capacity B yields a schedule on at most K machines by assigning the items of each bean to a single machine and scheduling them in ascending label order. Because the sum of processing times on any machine is at most B and each job has a deadline of B, no job misses its deadline. Thus, the jobs can be scheduled on at most K machines. Conversely, a feasible schedule on at most K machines must not exceed B in total processing time on any machine. Hence, each machine corresponds to a bin of capacity B. This is a polynomial-time reduction. □

In the following theorem, we establish that when the order is aligned with non-increasing slacks, both the FF algorithm and, equivalently, the NF algorithm (as shown in Lemma 3.10) attain an approximation ratio of at most 2.

Theorem 3.12. *First-fit is a 2-approximation algorithm for instances with non-increasing slacks, and this bound is tight.*

Proof. For any instance I with $d_j - p_j \geq d_{j+1} - p_{j+1}$ for $j \in [n-1]$, we prove by induction on k that for all $k \in \mathbb{N}$, if $OPT(I) = k$ then, $FF(I) < 2k$.

Base Case: If OPT fits all of the jobs in one machine, then FF can do the same. So, if $OPT(I) = 1$, then $FF(I) = 1$.

Induction Hypothesis: Assume that for all $k^{'} \leq k$, if $OPT(I) = k^{'}$ then, $FF(I) < 2k^{'}$.

Induction Step: Assume that $OPT(I) = k + 1$. We use proof by contradiction. Let us assume, to the contrary, that $FF(I) \geq 2k + 2$. We assume that machines $i \in \mathcal{M}^*$ are ordered in such a way that

$$l_1^* < l_2^* < \ldots < l_{k+1}^*.$$

Thus, the last job in the order (the job with the smallest slack) is scheduled on machine $k + 1$ in OPT. Moreover, w.l.o.g., we further assume that I is the smallest such instance in terms of the number of jobs. This means that l_{k+1}^* is the only job on machine $2k + 2$ in FF, as otherwise there is a smaller instance by removing all lower priority jobs. Note that by removing such jobs, the number of machines in the optimal schedule does not change because it contradicts the induction hypothesis, and FF uses $2k + 2$ machines. Thus, $l_{k+1}^* = n$.

Claim 3.13. *For $i \in [2k + 1]$,*

$$L_{k+1}^* < L_i + p_{l_{k+1}^*}.$$

Proof. The assignment of job l^*_{k+1} on machine $k+1$ in OPT implies that $L^*_{k+1} \leq d_{l^*_{k+1}}$. Given that l^*_{k+1} is scheduled on $2k+2$ in FF, it means that it is rejected on all machines $1, \ldots, 2k+1$, which implies that $L_i + p_{l^*_{k+1}} > d_{l^*_{k+1}}$ for $i \in [2k+1]$. Using these two inequalities we conclude that, $L^*_{k+1} < L_i + p_{l^*_{k+1}}$ for $i \in [2k+1]$. □

Claim 3.14. *For $i < k+1$ and $q \in [2i]$, we have*

$$L^*_i < L_q + p_{l^*_i}.$$

Proof. Consider τ as the schedule generated by the FF algorithm, where $\tau(j)$ denotes the machine on which job $j \in J$ is scheduled. Assume that $i < k+1$. We will show that $\tau(l^*_i) \geq 2i+1$ in FF. The result then follows because $L^*_i \leq d_{l^*_i} < L_q + p_{l^*_i}$ as job l^*_i is rejected on machines $q \in [2i]$. Suppose, by contradiction, that job l^*_i is scheduled on machine $\tau(l^*_i) \leq 2i$ in FF. Let job j be the first job on machine $\tau(l^*_i)+1$ in FF. Observe from Lemma 3.10 that $j > l^*_i$. Consider the instance I' defined by jobs $j, \ldots, n$. Given that in OPT for I, all these jobs are assigned to machines $i+1, \ldots, k+1$, we know that $OPT(I') \leq k+1-i$. Moreover, applying FF to I' yields the same number of machines as FF needed for these jobs in I, because of Lemma 3.10. Hence, $FF(I') = 2k+2-\tau(l^*_i) \geq 2(k+1-i)$. This, however contradicts the induction hypothesis of Theorem 3.12. □

We conclude the proof by using the above two claims to derive the contradictory statement $\sum_{i=1}^{k+1} L^*_i < \sum_{q=1}^{2k+2} L_q$, where both sides of the inequality should indeed be equal to the sum of processing times of all jobs. Define $M_l = \{q \in [2k+2] \mid l^*_i \in J_q \text{ for some } i \in [k+1]\}$. For all $q \in M_l$, we have that $L_q \geq \sum_{l^*_i \in J_q} p_{l^*_i}$ and hence

$$\sum_{i=1}^{k+1} p_{l^*_i} \leq \sum_{q \in M_l} L_q.$$

As we proved earlier in Claim 3.14, for each l_i^* where $i \in [k]$, we have $\tau(l_i^*) \geq 2i+1$. To each l_i^* with $i \in [k]$, we assign a machine $q(l_i^*) \in M \setminus M_l$ with $q(l_i^*) \leq 2i$ and $q(l_i^*) \neq q(l_v^*)$ for all $v \in [i-1]$. Such an assignment is feasible, because, for each l_i^* with $i \in [k]$, there are $2i$ potential machines. Out of these $2i$ machines, at most $i-1$ machines are in M_l (all jobs v with $v \geq l_i^*$ have $\tau(v) \geq 2i+1$) and at most $i-1$ machines are used for the assignment of a job l_v^* with $v \in [i-1]$. This means that at least two candidates are available for $q(l_i^*)$. Moreover, we assign job l_{k+1}^* to a machine $q(l_{k+1}^*) \in M \setminus M_l$ with $q(l_{k+1}^*) \leq 2k+1$ and $q(l_{k+1}^*) \neq q(l_v^*)$ for all $v \in [k]$. Such an assignment is feasible because there are $2k+1$ potential machines. Out of these $2k+1$ machines, at most k machines are in M_l, and at most k machines are used for the assignment of a job l_v^* with $v \in [k]$. This means that at least one candidate is available for $q(l_{k+1}^*)$. Now, for $i \in [k+1]$, we have by either Claim 3.13 or Claim 3.14 that $L_i^* < L_{q(l_i^*)} + p_{l_i^*}$. Summing over all $i \in [k+1]$

$$\sum_{i=1}^{k+1} L_i^* < \sum_{i=1}^{k+1} \left[L_{q(l_i^*)} + p_{l_i^*} \right] \leq \sum_{q \in M \setminus M_l} L_q + \sum_{q \in M_l} L_q = \sum_{q=1}^{2k+2} L_q,$$

which is a contradiction. By the principle of induction, the claim holds for all $k \in \mathbb{N}$. □

Example 3.2. *Consider the following example where $n = 3k+1$ for some $k \geq 1$, featuring three distinct types of jobs (see Figure 3.4 for an illustration). Specifically, $\mathcal{A} = \{a_1, \ldots, a_k\}$, $\mathcal{B} = \{b_1, \ldots, b_k\}$, and $\mathcal{C} = \{c_1, \ldots, c_{k+1}\}$. In this context, the pair $(k, 2k) = (p_{a_i}, d_{a_i})$ for $a_i \in \mathcal{A}$, the pair $(1, k+1) = (p_{b_i}, d_{b_i})$ for $b_i \in \mathcal{B}$, and the pair $(k+1, 2k+1) = (p_{c_i}, d_{c_i})$ for $c_i \in \mathcal{C}$. Assume the fixed order is given by $a_1 < b_1 < a_2 < b_2 < \cdots < a_k < b_k < c_1 < c_2 < \cdots < c_{k+1}$. Then, the approximation ratio is obtained as $\frac{2k+1}{k+1} < 2$.*

In Example 3.2, we present an example that illustrates the tightness of the bound of 2.

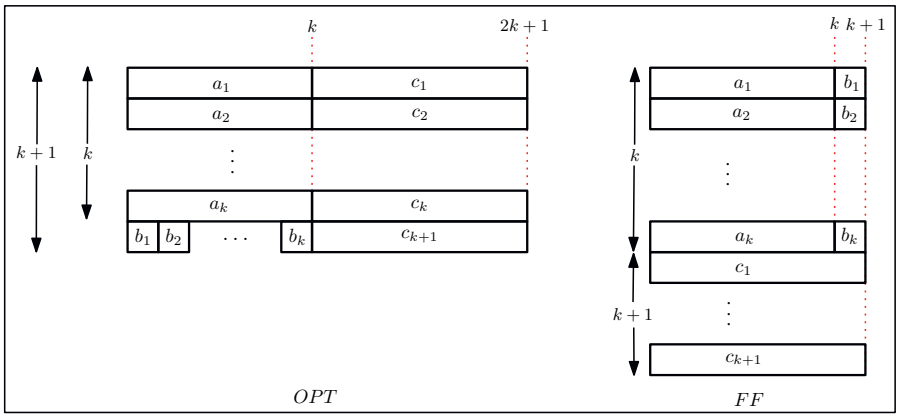


Figure 3.4: An example with the approximation ratio of $(2k+1)/(k+1)$.

3.3.3 Non-Decreasing Slacks

In this subsection, we prove that the approximation ratio of the first-fit algorithm is 2 when the fixed order is aligned with non-decreasing slacks.

Lemma 3.15. *The problem $P|r_j = r, p_j, d_j, \pi^*|m$ is NP-hard when the fixed order aligns with slacks that are in a non-decreasing order, i.e., $d_j - p_j \leq d_{j+1} - p_{j+1}$ for $j \in [n-1]$.*

Proof. Consider an instance of the decision version of the Bin Packing Problem as given in Definition 1.1. Change the label of the items so that $s(a_1) \geq \ldots \geq s(a_n)$. Construct a scheduling instance with n jobs each with $p_j = s(a_j)$ and deadline $d_j = B$. Since $\lambda_j = B - s(a_j)$, the slacks satisfy $\lambda_1 \leq \cdots \leq \lambda_n$, so the order matches with the non-decreasing order of slacks. It remains to show that the scheduling instance admits a feasible schedule on at most K machines if and only if the bin-packing instance can be packed into at most K bins of capacity B. Packing the items into at most K bins of capacity B yields a schedule on at most K machines by assigning the items of each bean to a single

machine and scheduling them in descending label order. Because the sum of processing times on any machine is at most B and each job has a deadline of B, no job misses its deadline. Thus, the jobs can be scheduled on at most K machines. Conversely, a feasible schedule on at most K machines must not exceed B in total processing time on any machine. Hence, each machine corresponds to a bin of capacity B. This is a polynomial-time reduction. □

Theorem 3.16. *First-fit is a 2-approximation algorithm for instances with non-decreasing slacks. Furthermore, this bound is tight.*

Proof. For any instance I with $d_j - p_j \leq d_{j+1} - p_{j+1}$ for $j \in [n-1]$, we prove by induction on k that for all $k \in \mathbb{N}$, if $OPT(I) = k$ then, $FF(I) < 2k$.

Base Case: If OPT fits all of the jobs in one machine, then FF can do the same. So, if $OPT(I) = 1$, then $FF(I) = 1$.

Induction Hypothesis: Assume that for all $k^{'} \leq k$, if $OPT(I) = k^{'}$ then, $FF(I) < 2k^{'}$.

Induction Step: Assume that $OPT(I) = k + 1$. We use a proof by contradiction. To the contrary, let us assume that $FF(I) \geq 2k + 2$. Likewise, in Theorem 3.12, we assume that I is the smallest such instance by means of the number of jobs and that machines $i \in \mathcal{M}^*$ are ordered in such a way that $l_1^* < l_2^* < \ldots < l_{k+1}^*$. Thus, the last job in the order (the job with the highest slack value) is scheduled on machine $k + 1$ in OPT.

Claim 3.17. *For $i \leq k + 1$ and $q \in [2k + 1]$, we have*

$$L_i^* < L_q + p_{l_i^*}.$$

Proof. Since l_{k+1}^* is rejected on machines $1, \ldots, 2k + 1$, we know that for all $i \in [2k + 1]$, we have $d_{l_{k+1}^*} - p_{l_{k+1}^*} < L_i$. Moreover, based on the assumption we know that l_{k+1}^* has the largest slack, i.e., $d_j - p_j \leq$

$d_{l^*_{k+1}} - p_{l^*_{k+1}}$ for all $j \in J$. Furthermore, we have that $L^*_i - p_{l^*_i} \leq d_{l^*_i} - p_{l^*_i}$. Utilizing the previous inequalities for $i \in [k+1]$ and $q \in [2k+1]$, provides us with $L^*_i < L_q + p_{l^*_i}$. □

We can now use the above claim to derive the contradictory statement that $\sum_{i=1}^{k+1} L^*_i < \sum_{q=1}^{2k+2} L_q$, where both sides of the inequality should indeed be equal to the sum of the processing times of all jobs. We define $M_l = \{q \in [2k+2] \mid l^*_i \in J_q \text{ for some } i \in [k+1]$. For all $q \in M_l$, we have that $L_q \geq \sum_{l^*_i \in J_q} p_{l^*_i}$ and hence, we have that

$$\sum_{i=1}^{k+1} p_{l^*_i} \leq \sum_{q \in M_l} L_q.$$

To each l^*_i with $i \in [k+1]$, we assign a machine $q(l^*_i) \in M \setminus M_l$ with $q(l^*_i) \neq q(l^*_v)$ for all $v \in [k+1]$ and $l^*_i \neq l^*_v$. Such an assignment is feasible, because based on Claim 3.17, for each l^*_i with $i \in [k+1]$, there are $2k+1$ potential machines. Out of these $2k+1$ machines, at most k machines are in M_l (note that job l^*_{k+1} is alone in machine $2k+2$, thus $2k+2 \in M_l$), and at most $k+1$ machines are used for the assignment of a job l^*_v with $v \in [k+1]$. This means that enough candidates are available for $q(l^*_i)$.

Now, for each $i \in [k+1]$, we have by Claim 3.17 that $L^*_i < L_{q(l^*_i)} + p_{l^*_i}$. Summing over all $i \in [k+1]$

$$\sum_{i=1}^{k+1} L^*_i < \sum_{i=1}^{k+1} \left[L_{q(l^*_i)} + p_{l^*_i} \right] \leq \sum_{q \in M \setminus M_l} L_q + \sum_{q \in M_l} L_q = \sum_{q=1}^{2k+2} L_q,$$

which is a contradiction. By the principle of induction, the claim holds for all $k \in \mathbb{N}$. □

Remark 3.18. In Example 3.2, the slack of all the jobs is equal. Therefore, it can be used as a lower bound example to show the tightness of ratio 2 in the case of non-decreasing slacks.

3.3.4 Non-Increasing Deadlines

In this subsection, we prove that the approximation ratio of the first-fit algorithm is 2 when the fixed order is aligned with non-increasing deadlines.

Lemma 3.19. *The problem $P|r_j = r, p_j, d_j, \pi^*|m$ is NP-hard when the fixed order aligns with deadlines that are in a non-increasing order, i.e., $d_j \geq d_{j+1}$ for $j \in [n-1]$.*

Proof. Consider an instance of the decision version of the Bin Packing Problem as given in Definition 1.1. Construct a scheduling instance with n jobs each with $p_j = s(a_j)$ and deadline $d_j = B$. Since all deadlines are B, we have $d_1 = B \geq \ldots \geq d_n = B$, so the deadlines form a non-increasing sequence. If the Bin Packing instance can be packed into at most K bins of capacity B, assign the corresponding jobs to at most K machines. The total load per machine does not exceed B, so no job misses its deadline B. Conversely, in any feasible schedule on at most K machines, each machine's total processing time is at most B. Hence we obtain a packing of $\{a_1, \ldots, a_n\}$ into at most K bins of capacity B. This is a polynomial-time reduction. □

Theorem 3.20. *First-fit is a 2-approximation algorithm for instances with non-increasing deadlines.*

Proof. For any instance I with $d_j \geq d_{j+1}$ for $j \in [n-1]$, we prove by induction on k that for all $k \in \mathbb{N}$, if $OPT(I) = k$ then, $FF(I) < 2k$.

Base Case: If OPT fits all of the jobs in one machine then FF can do the same. So, if $OPT(I) = 1$, then $FF(I) = 1$.

Induction Hypothesis: Assume that for all $k' \leq k$, if $OPT(I) = k'$ then, $FF(I) < 2k'$.

Induction Step: Assume that $OPT(I) = k + 1$. We use a proof by contradiction. Let us assume, to the contrary, that $FF(I) \geq 2k + 2$.

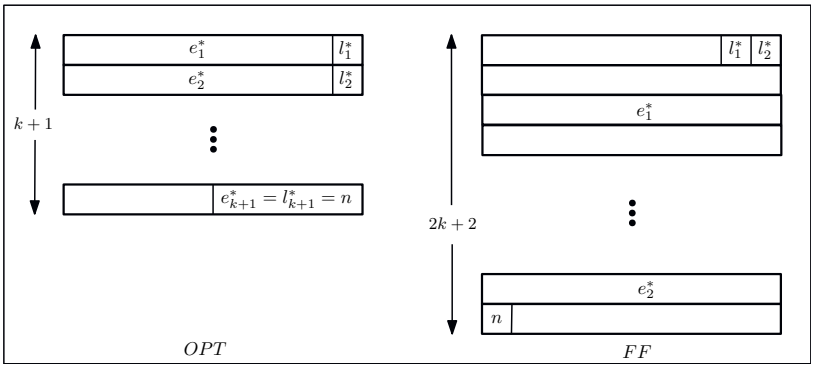


Figure 3.5: An example of the placement of jobs e_i^* in the OPT and FF schedules.

Moreover, as in Theorem 3.12, we assume that I is the smallest such instance by means of the number of jobs.

For all $i \in [k+1]$, define $E_i = \{j \in J_i^* \mid \tau(j) = \max(\tau(k)) \text{ for all } k \in J_i^*\}$ and let $e_i^* \in E_i$ be the job with $e_i^* \geq j$ for all $j \in E_i$ (see Figure 3.5 for an example). We further assume that machines $i \in \mathcal{M}^*$ are ordered in such a way that

$$\tau(e_1^*) \leq \tau(e_2^*) \leq \ldots \leq \tau(e_{k+1}^*).$$

Hence, the last job in the order (with the smallest deadline) is scheduled on machine $k+1 \in \mathcal{M}^*$. Notice that $e_{k+1}^* = n$.

Claim 3.21. *For $i < k+1$ and $q \in 2i$, we have*

$$L_i^* < L_q + p_{e_i^*}.$$

Proof. Assume that $i < k+1$. We will show that $\tau(e_i^*) \geq 2i+1$ in FF. The result then follows because $L_i^* \leq d_{l_i^*} \leq d_{e_i^*} < L_q + p_{e_i^*}$ as job e_i^* is rejected on machines $q \in [2i]$. Suppose, by contradiction, that job e_i^* is scheduled on a machine $q \leq 2i$ in FF. Based on the definition of job e_i^*, we know that $\tau(j) \leq q$ for all jobs $j \in J_i^*$. Moreover, based

on our earlier assumption on the order of the machines in the optimal schedule, we know that $\tau(j) \leq q$ for all $j \in \cup_{i-1}^{s=1} J_s^*$. Consider the instance I' defined by jobs $j \in \cup_{k+1}^{s=i+1} J_s^*$. Given that in the optimal schedule for I, all these jobs are assigned to machines $i+1, \ldots k+1$, we know that $OPT(I') = k+1-i$. Moreover, applying FF to I' yields at least the same number of machines as FF needed for these jobs in I. Hence, $FF(I') = 2k+2-q \geq 2(k+1-i)$. This however contradicts the induction hypothesis. □

Also, using the same argument as in Claim 3.13, it can be proved that for all $i \in [2k+1]$, we have $L_{k+1}^* < L_i + p_{e_{k+1}^*}$ (note that $e_{k+1}^* = l_{k+1}^* = n$). By utilizing this inequality and Claim 3.21, we can conclude the proof by driving the contradictory statement of $\sum_{i=1}^{k+1} L_i^* < \sum_{q=1}^{2k+2} L_q$, where both sides of the inequality should indeed be equal to the sum of processing times of all jobs.

Define $M_e = \{q \in [2k+2] \mid e_i^* \in J_q \text{ for some } i \in [k+1]\}$. For all $q \in M_e$, we have that $L_q \geq \sum_{e_i^* \in J_q} p_{e_i^*}$ and hence, we have that

$$\sum_{i=1}^{k+1} p_{e_i^*} \leq \sum_{q \in M_e} L_q.$$

As we proved earlier in Claim 3.21, for each e_i^* where $i \in [k]$, we have $\tau(e_i^*) \geq 2i+1$. To each e_i^* with $i \in [k]$, we assign a machine $q(e_i^*) \in M \setminus M_e$ with $q(e_i^*) \leq 2i$ and $q(e_i^*) \neq q(e_v^*)$ for $v \in [i-1]$. Such an assignment is feasible because, for each e_i^* with $i \in [k]$, there are $2i$ potential machines. Out of these $2i$ machines, at most $i-1$ machines are in M_e (all jobs e_v^* with $v > i$ have $\tau(e^*) \geq 2i+1$) and at most $i-1$ machines are used for the assignment of a job e_v^* with $v \in [i-1]$. This means at least two candidates are available for $q(e_i^*)$. Moreover, we assign job e_{k+1}^* to a machine $q(e_{k+1}^*) \in M \setminus M_e$ with $q(e_{k+1}^*) \leq 2k+1$ and $q(e_{k+1}^*) \neq q(e_v^*)$ for all $v \in [k]$. Such an assignment is feasible because there are $2k+1$ potential machines. Out of these $2k+1$ machines, at most k machines are in M_e, and at most k machines are used for the

assignment of a job e_v^* with $v \in [k]$. Hence, at least one candidate is available for $q(e_{k+1}^*)$.

As mentioned before, using Claim 3.13, it can be proved that for all $i \in [2k+1]$, we have $L_{k+1}^* < L_i + p_{e_{k+1}^*}$. Now, for each $i \in [k+1]$, we have by either the previous inequality or by Claim 3.21 that $L_i^* < L_{q(e_i^*)} + p_{e_i^*}$. Summing over all $i \in [k+1]$

$$\sum_{i=1}^{k+1} L_i^* < \sum_{i=1}^{k+1} \left[L_{q(e_i^*)} + p_{e_i^*} \right] \leq \sum_{q \in M \setminus M_e} L_q + \sum_{q \in M_e} L_q = \sum_{q=1}^{2k+2} L_q,$$

which is a contradiction. By the principle of induction, the claim holds for all $k \in \mathbb{N}$. □

3.3.5 Approximation Factors with Few Machines

In this subsection, we allow for an arbitrary fixed order. We demonstrate constant approximation factors of 3/2 and 2 when the number of machines used in OPT is 2 and 3, respectively. These analyses highlight how the complexity increases when proving a constant approximation factor for the first-fit algorithm.

Theorem 3.22. *First-fit is a 3/2-approximation when the optimal schedule opens at most two machines.*

Proof. Consider any instance I. Observe that if $OPT(I) = 1$, then $FF(I) = 1$.

In the following, we prove that if $OPT(I) = 2$ then, $FF(I) < 4$. Assume by contradiction that for some instance I', $OPT(I') = 2$ and $FF(I') = 4$ (as before we consider I' to be the smallest such instance in terms of the number of jobs).

Assume w.l.o.g. that job n is scheduled in the second machine in the optimal schedule. We have that $L_2^* \leq d_n$. We also know that job n

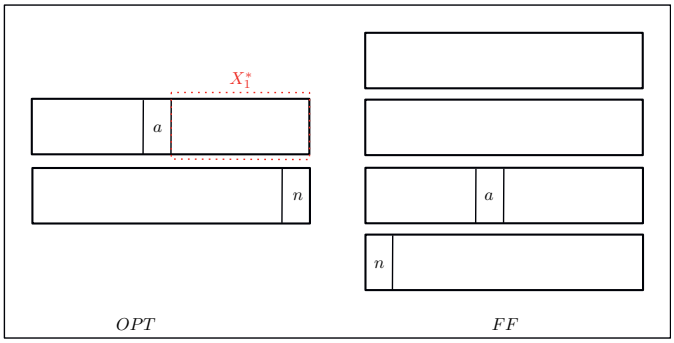


Figure 3.6: Instance $I^{'}$ where $OPT(I^{'}) = 2$ and $FF(I^{'}) = 4$.

is rejected from the first three machines in the FF algorithm. Thus, $d_n < L_i + p_n$ with $i = 1, 2, 3$. Hence, for $i \in \{1, 2, 3\}$ we have

$$L_2^* < L_i + p_n. \tag{3.8}$$

Observe that $J_3 \setminus J_2^* \neq \emptyset$, as otherwise, the last job could have fit in the third machine. Therefore, there should be a job $a \in J_1^* \cap J_3$. Let $a \in J_1^*$ be such a job with the lowest possible priority. Denote also by X_1^* the set of all jobs $j \in J_1^*$ with $j > a$ (see Figure 3.6 for an illustration). Note that the set X_1^* can be empty. Below, we demonstrate that all elements $b \in X_1^*$ are allocated to the first machine by the FF algorithm.

Lemma 3.23. *First-fit schedules all the jobs $b \in X_1^*$ in the first machine.*

Proof. We prove by contradiction. Consider Inequality (3.8) with $i = 3$

$$\begin{aligned} L_2^* < L_3 + p_n &\Leftrightarrow \sum_{\substack{j \in J_2^* \\ j \in J_1}} p_j + \sum_{\substack{j \in J_2^* \\ j \in J_2}} p_j + \sum_{\substack{j \in J_2^* \\ j \in J_3}} p_j < \sum_{\substack{j \in J_3 \\ j \in J_1^*}} p_j + \sum_{\substack{j \in J_3 \\ j \in J_2^*}} p_j \\ &\Leftrightarrow \sum_{\substack{j \in J_2^* \\ j \in J_1}} p_j + \sum_{\substack{j \in J_2^* \\ j \in J_2}} p_j < \sum_{\substack{j \in J_3 \\ j \in J_1^* \\ j \neq a}} p_j + p_a. \end{aligned} \tag{3.9}$$

Now, assume by contradiction that a job $b \in X_1^*$ is not scheduled in the first machine by the FF algorithm. Note that job b cannot reside in the third machine due to $b > a$ (as per the definition, job a should hold the lowest priority in the third machine, belonging to J_1^*). Thus, job b should be scheduled on the second machine. The optimal schedule fits job b in the first machine. We get

$$\sum_{\substack{j \in J_1^* \\ j<b \\ j \neq a}} p_j + p_a + p_b \leq d_b. \tag{3.10}$$

Also, job b is rejected by the FF algorithm from the first machine. Hence

$$\sum_{\substack{j \in J_1^* \\ j \in J_1 \\ j<b}} p_j + \sum_{\substack{j \in J_2^* \\ j \in J_1 \\ j<b}} p_j + p_b > d_b. \tag{3.11}$$

By combining Inequalities (3.10) and (3.11), we have

$$\begin{aligned} \sum_{\substack{j \in J_1^* \\ j<b \\ j \neq a}} p_j + p_a < \sum_{\substack{j \in J_1^* \\ j \in J_1 \\ j<b}} p_j + \sum_{\substack{j \in J_2^* \\ j \in J_1 \\ j<b}} p_j \Leftrightarrow \sum_{\substack{j \in J_1^* \\ j \in J_2 \\ j<b}} p_j + \sum_{\substack{j \in J_1^* \\ j \in J_3 \\ j<b \\ j \neq a}} p_j + p_a < \sum_{\substack{j \in J_2^* \\ j \in J_1 \\ j<b}} p_j \\ \Leftrightarrow \sum_{\substack{j \in J_1^* \\ j \in J_2 \\ j<b}} p_j + \sum_{\substack{j \in J_1^* \\ j \in J_3 \\ j \neq a}} p_j + p_a < \sum_{\substack{j \in J_2^* \\ j \in J_1 \\ j<b}} p_j. \end{aligned} \tag{3.12}$$

Note that the last equivalence follows because all jobs in X_1^* are in J_1 or J_2. Since the processing times are positive, it is easy to verify that Inequalities (3.9) and (3.12) contradict. Therefore, the proof holds. □

The optimal schedule fits job a in the first machine. We get

$$L_1^* - \sum_{j \in X_1^*} p_j \leq d_a \tag{3.13}$$

Also, job a is rejected from the first machine by the FF algorithm. We get

$$L_1 - \sum_{\substack{j \in J_1 \\ j > a}} p_j + p_a > d_a \tag{3.14}$$

By combining Inequalities (3.13) and (3.14), we have

$$L_1^* - \sum_{j \in X_1^*} p_j < L_1 - \sum_{\substack{j \in J_1 \\ j > a}} p_j + p_a. \tag{3.15}$$

Based on Lemma 3.23, we know that all jobs $b \in X_1^*$ belong to J_1. This means that $\sum_{\substack{j \in J_1 \\ j > a}} p_j$ contains $\sum_{j \in X_1^*} p_j$. Thus

$$L_1^* < L_1 + p_a. \tag{3.16}$$

Consider Inequality (3.8) with $i = 2$. By summing up Inequalities (3.8) and (3.16), we obtain

$$L_1^* + L_2^* < L_1 + p_a + L_2 + p_n < L_1 + L_2 + L_3 + L4,$$

where second inequality is derived from the fact that jobs a and n belong to J_3 and J_4, respectively. Thus, their processing times cannot exceed the load of their respective machines. This contradicts the fact that the total loads in both the optimal schedule and the FF algorithm schedule must be equal. □

Now, we demonstrate that the FF algorithm achieves a 2-approximation factor when the optimal schedule employs three machines.

Theorem 3.24. *First-fit is a 2-approximation when the optimal schedule uses at most three machines.*

Proof. For any instance I, we prove that if $OPT(I) = 3$ then, $FF(I) < 7$. Observe that if $OPT(I)$ = 1, then $FF(I) = 1$, and when $OPT(I)$ = 2, based on Theorem 3.22, $FF(I) < 4$. Assume by contradiction that there is an instance I with $OPT(I) = 3$ and $FF(I) = 7$. As before we assume that I is an instance with those properties having the fewest number of jobs. In particular, the lowest priority job n is the only job assigned to the last opened machine when the FF algorithm terminates.

For each machine $i \in \{1, 2, 3\}$ in the optimal solution, we define a critical job c_i^* to be the lowest priority job which is scheduled on machine i in OPT and on machines 4 to 7 by the FF algorithm, $c_i^* = \max\{j : \tau^*(j) = i, \tau(j) \geq 4\}$.

We assume that the machines in OPT are ordered such that $c_1^* < c_2^* < c_3^*$. Clearly, $c_3^* = n$, and for $i = 1, 2$ it needs to be shown that c_i^* exists. Assume to the contrary that $\tau(j) \leq 3$ for all j with $\tau^*(j) = 1$. This means that for any job j with $\tau(j) \in \{4, \ldots, 7\}$, we have $\tau^*(j) \in \{2, 3\}$. If we now consider the instance with jobs only in the set $J' = \{j : 4 \leq \tau(j) \leq 7\}$, then FF will schedule the jobs in J' just like it would for the instance I on four machines, while those jobs are scheduled on two machines in the optimal solution for I. This contradicts Theorem 3.22. The same argument shows that c_2^* exists.

We will now derive the following inequality

$$\sum_{i=1}^{3} L_i^* < \sum_{i=1}^{7} L_i, \tag{3.17}$$

where L_i^* is the completion time of machine i in the optimal solution and L_i is the completion time of machine i in the FF solution. Both sides of the inequality are equal to the sum of processing times of all jobs and hence this contradicts our initial assumption.

First of all, since job $c_3^* = n$ is scheduled on machine 3 in the optimal solution and rejected from the first six machines by the FF algorithm, we have that

$$L_3^* \leq d_n < L_i + p_n \tag{3.18}$$

for every $i \in \{1, \ldots, 6\}$. We now consider two cases.

Case (I): there exists a job j such that $c_1^* < j < c_2^*$, $\tau^*(j) = 1$ and $\tau(j) \in \{2, 3\}$. If this is the case, we replace c_1^* with the highest indexed, i.e. lowest priority, job j that satisfies the above three properties.

Let us assume that $\tau(c_1^*) = 2$ and that $\tau(c_2^*) = 4$. The cases when $\tau(c_1^*) = 3$ or when $\tau(c_2^*) \in \{5, 6\}$ can be handled similarly by renaming. Since c_1^* is scheduled on machine 1 in the optimal solution and rejected from machine 1 by the first algorithm, we obtain

$$L_1^* - \sum_{\substack{j > c_1^* \\ \tau^*(j)=1}} p_j \leq d_{c_1^*} < L_1 - \sum_{\substack{j > c_1^* \\ \tau(j)=1}} p_j + p_{c_1^*}. \tag{3.19}$$

Similarly, since c_2^* is scheduled on machine 2 in the optimal solution and rejected from machine 3 by the first algorithm, we obtain

$$L_2^* - \sum_{\substack{j > c_2^* \\ \tau^*(j)=2}} p_j \leq d_{c_2^*} < L_3 - \sum_{\substack{j > c_2^* \\ \tau(j)=3}} p_j + p_{c_2^*}. \tag{3.20}$$

When we combine Inequalities (3.18) (with $i = 5$), (3.19) and (3.20), we find that

$$\begin{aligned} L_1^* + L_2^* + L_3^* < L_1 + L_3 + L_5 + p_{c_1^*} + p_{c_2^*} + p_{c_3^*} \\ + \sum_{\substack{j > c_1^* \\ \tau^*(j)=1}} p_j + \sum_{\substack{j > c_2^* \\ \tau^*(j)=2}} p_j - \sum_{\substack{j > c_1^* \\ \tau(j)=1}} p_j - \sum_{\substack{j > c_2^* \\ \tau(j)=3}} p_j. \end{aligned}$$

Since $\tau(c_2^*) = 4$ and $\tau(c_3^*) = 7$, we also have that $p_{c_2^*} \leq L_4$ and $p_{c_3^*} \leq L_7$. Now consider the positive p_j terms in the first sum on the right-hand side of the above inequality. By the choice of c_1^*, if $j > c_1^*$ and $\tau^*(j) = 1$, then either $\tau(j) = 1$, in which case there is a corresponding negative term in the third sum, or $j > c_2^*$. In that case, there is a corresponding negative p_j term in the last sum, or $\tau(j) = 2$. Similarly, for the positive p_j terms in the second sum it holds that $\tau(j) \leq 3$, and there are either corresponding negative terms in the third or last sum or $\tau(j) = 2$. Hence, $p_{c_1^*}$ plus the last four sums can be bounded from above by L_2. Hence, we obtain the desired contradiction of Inequality (3.17) for the first case.

$$L_1^* + L_2^* + L_3^* < L_1 + L_3 + L_5 + L_4 + L_7 + L_2 \leq \sum_{i=1}^{7} L_i.$$

Case (II): In this case there is no j such that $c_1^* < j < c_2^*$, $\tau^*(j) = 1$ and $\tau(j) \in \{2, 3\}$, i.e. $\tau(j) = 1$ for all such j. As before, we have that

$$L_1^* - \sum_{\substack{j > c_1^* \\ \tau^*(j)=1}} p_j \leq d_{c_1^*} < L_1 - \sum_{\substack{j > c_1^* \\ \tau(j)=1}} p_j + p_{c_1^*} \tag{3.21}$$

and

$$L_2^* - \sum_{\substack{j > c_2^* \\ \tau^*(j)=2}} p_j \leq d_{c_2^*} < L_2 - \sum_{\substack{j > c_2^* \\ \tau(j)=2}} p_j + p_{c_2^*}, \tag{3.22}$$

since c_2^* is rejected from machine 2 by the FF algorithm.

We now use (3.18) with that value of $4 \leq i \leq 6$ such that $\tau(c_1^*) \neq i$ and $\tau(c_2^*) \neq i$, e.g. if $\tau(c_1^*) = 4$ and $\tau(c_2^*) = 5$ we use $i = 6$. Together with (3.21) and (3.22), we get that

$$L_1^* + L_2^* + L_3^* < L_1 + L_2 + L_6 + p_{c_1^*} + p_{c_2^*} + p_{c_3^*} + \sum_{\substack{j > c_1^* \\ \tau^*(j)=1}} p_j + \sum_{\substack{j > c_2^* \\ \tau^*(j)=2}} p_j - \sum_{\substack{j > c_1^* \\ \tau(j)=1}} p_j - \sum_{\substack{j > c_2^* \\ \tau(j)=2}} p_j.$$

Again, the positive terms p_j in the first two sums either have a corresponding negative term or $\tau(j) = 3$ due to the choice of c_1^* and c_2^*. Since $p_{c_1^*} + p_{c_2^*} + p_{c_3^*} \leq L_4 + L_5 + L_7$, we finally get

$$L_1^* + L_2^* + L_3^* < L_1 + L_2 + L_6 + L_4 + L_5 + L_7 + L_3 \leq \sum_{i=1}^{7} L_i.$$

This contradicts (3.17) also in this case and completes the proof. □

3.3.6 Logarithmic Approximation for Arbitrary Orders

In this subsection, we establish that the problem with an arbitrary fixed order can be approximated within a factor of $O(\log n)$. This result is based on the Set Cover problem as described in [106].

Definition 3.25. *The input in the Set Cover problem is a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, and a cost function $c : \mathcal{S} \to \mathbb{Q}^+$. The goal is to find a minimum cost subcollection of $\mathcal{S}$ that covers all elements of U.*

Based on the Definition 3.25, each subset $S_i \in \mathcal{S}$ is associated with a cost $c(S_i) \in \mathbb{Q}^+$. The objective is to find a subcollection of $\mathcal{S}$ that covers all elements of U while minimizing the total cost. The greedy method described in Algorithm 7 provides an $O(\log n)$-approximation for this problem [106]. This algorithm iteratively selects the most *cost-effective* subset, which is the subset that minimizes the cost per uncovered

element, removes the elements covered by the selected subset, and repeats the process until all elements in U are covered.

Algorithm 7 Greedy Set Cover Algorithm

1: $C \leftarrow \varnothing$
2: **while** $C \neq U$ **do**
3: **Find** the most cost-effective set in the current iteration, say S.
4: Let $\alpha = \frac{c(S)}{|S \setminus C|}$. *(cost-effectiveness of S)*
5: Pick S, and for each $e \in S \setminus C$, set $\text{price}(e) = \alpha$.
6: $C \leftarrow C \cup S$
7: **end while**
8: Output the picked sets.

Theorem 3.26. *The problem $P|r_j = r, p_j, d_j, \pi^*|m$ can be approximated within a factor of $O(\log n)$.*

Proof. We define $\mathcal{S}$ as the collection of subsets of jobs S_i, where each subset contains the jobs that can be scheduled on a single machine in the given fixed order without violating any deadlines. Let $\bar{J}$ represent the set of unscheduled jobs, initially equal to J. While $\bar{J} \neq \emptyset$, perform the following steps

- Identify a subset S_i, where $i \in [|\mathcal{S}|]$, that contains the largest number of unscheduled jobs. The subset S_i must satisfy the condition that all jobs in S_i can be executed on a single machine in the given fixed order without violating any deadlines.
- Remove all jobs $j \in S_i$ from $\bar{J}$.

When all sets have the same cost, the greedy set cover algorithm chooses, in each iteration, the set covering the largest number of currently uncovered jobs. According to [106], the described greedy method results in a $O(\log n)$ approximation factor for the problem $P|r_j = r, p_j, d_j, \pi^*|m$. Given that there are an exponential number of subsets, we present a dynamic programming approach that runs in

polynomial time to identify a subset $S_i \subseteq \mathcal{S}$ that contains the maximum number of unscheduled jobs that can be successfully scheduled on a single machine without violating their respective deadlines. Let $D(i,k)$ denote the smallest possible completion time for scheduling k jobs selected from the set $\{1,\ldots,i\}$ on a single machine, such that no job violates its deadline. The recurrence relation for $D(i,k)$ is defined as follows

Case (I): Job i is not included in the solution. If job i is not a part of the solution, the shortest completion time for k jobs remains the same as for k jobs chosen from the subset $\{1,\ldots,i-1\}$. So, $D(i,k) = D(i-1,k)$.

Case (II): Job i is included in the solution. If job i is included in the solution, then $k-1$ jobs must be selected from the subset $\{1,\ldots,i-1\}$. The completion time for these $k-1$ jobs must allow job i to be scheduled within its deadline d_i. In this case, if $D(i-1,k-1)+p_i \leq d_i$ we have

$$D(i,k) = D(i-1,k-1) + p_i.$$

Therefore,

$$D(i,k) = \begin{cases} \min\Big(D(i-1,k),\ D(i-1,k-1)+p_i\Big), & \text{if } D(i-1,k-1)+p_i \leq d_i, \\ D(i-1,k), & \text{otherwise.} \end{cases}$$

The boundary conditions are $D(0,0) = 0$, $D(i,0) = 0$, for all i, and $D(0,k) = +\infty$, for all $k > 0$. Using this dynamic programming approach, we search for the maximum k that $D(n,k) < +\infty$, which can be done in polynomial time $O(n^2)$. We must run the dynamic program at most $O(n)$ times. Hence, the total running time is $O(n^3)$. □

3.4 Concluding Remarks

In this study, we tackled the problem of fixed order scheduling with deadlines on a minimum number of machines. This setting models capacity planning situations in which the global execution order is externally imposed and cannot be rearranged on machines. We

investigated several algorithmic approaches. First, we showed that simple one-pass rules, such as next-fit, can perform arbitrarily poorly under the fixed-order and deadline constraints. Second, we analyzed first-fit. While first-fit is not always optimal, it is optimal in certain restricted cases (for example, when all processing times are equal), and it can be shown to use at most twice as many machines as an optimal solution in several structured subclasses, including instances with monotone slack, instances with monotone deadlines, and instances in which the optimal schedule uses at most three machines. Third, we described an iterative approach in which the maximum number of unscheduled jobs that can be successfully scheduled on a single machine without violating their respective deadlines, and the global order is selected in each iteration. With the same argument as the set cover problem, this approach yields an $O(\log n)$-approximation for the general problem. The analysis in this chapter leaves several open questions

- **Worst-case guarantees for greedy policies.** It is currently not known whether first-fit admits a constant worst-case performance guarantee for the general problem. One possibility is that first-fit always uses at most a constant factor more machines than an optimal schedule (for example, a factor of 2) on every instance. Another possibility is that there exist explicit families of instances that force first-fit to exceed any fixed constant factor. Resolving this question would clarify whether simple greedy rules can be used with theoretical guarantees in full generality, or whether their guarantees are limited to structured subclasses. Our analysis of special cases shows that first-fit has a constant approximation factor when the global order satisfies additional regularity, such as monotone slack or monotone deadlines. This suggests a possible approach toward the general question: identify and classify further structural patterns in the job order for which first-fit can be proved to be near-optimal. Examples include orders in which the slack sequence is single-peaked or single-caved (i.e., increases and then decreases, or decreases and

then increases), or orders with non-decreasing deadlines. A systematic understanding of such intermediate cases may either lead to prove that first-fit has a constant approximation factor (preferably 2). Another interesting future direction might be to examine other algorithms as potential approximation methods to determine if they offer better bounds.

- **Linear programming relaxations and rounding.** We formulated the problem as an ILP. However, solving this ILP becomes computationally intractable for large instances. Since the underlying problem is NP-hard, generic ILP solvers may exhibit exponential growth in runtime and memory consumption as the number of jobs increases. So, the ILP is exact but not tractable in general for large instances. A natural next step is to construct an LP relaxation of that ILP. Such an LP can be solved in polynomial time. If one can design a rounding procedure that converts any fractional LP solution into a feasible schedule and prove that this rounding increases the machine count by at most a factor α, then this would yield an α-approximation algorithm for the general problem. Here α would be determined by the worst-case loss in the rounding step (equivalently, by the integrality gap of the LP relaxation).
- **Model extensions.** In practice, additional features commonly arise: heterogeneous machine speeds, machine eligibility restrictions, different release times, and different orders per machine. Extending approximation guarantees to such settings is not straightforward.

Chapter 4

Cost Preserving Dependent Rounding for Allocation Problems

4.1 Introduction

A successful paradigm in the design of approximation algorithms is to first solve a continuous relaxation, which can typically be done efficiently using linear programming, and then to round the fractional solution $x \in [0,1]^d$ to an integer solution $X \in \{0,1\}^d$. Careful choices need to be made in the rounding step so that the error introduced is low. Independent randomized rounding is one of the most natural rounding schemes. In a simple variant, we independently set each variable X_i to 1 with probability x_i and to 0 with probability $1 - x_i$. The advantage is that the value of every linear function (over the d variables) is maintained in expectation. Moreover, for linear functions with small coefficients, a Chernoff bound yields strong concentration guarantees for the value. Hence, if the initial solution x satisfies some linear constraints from the continuous relaxation, we can often argue

This chapter is based on: Lars Rohwedder, Arman Rouhani, and Leo Wennmann. "Cost Preserving Dependent Rounding for Allocation Problems". In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2025, pp. 127-1.

with several Chernoff bounds combined by a union bound that they are still satisfied up to a small error by the rounded solution X.

In some cases, however, the problem dictates structures or hard constraints on the solution. For example, we might require X to be (the incidence vector of) a perfect matching in a given graph or the basis of a given matroid. Perfect matchings or bases are objects that are quite simple in many computational aspects, but it is typically very unlikely that independent randomized rounding on a fractional object, that is, a point in the convex hull of the objects we want, results in one of these objects. This motivates so-called dependent randomized rounding. Here, the goal is to achieve similar guarantees as independent randomized rounding, but with a distribution over a restricted set of objects, which necessarily introduces some dependency between variables. These rounding schemes are typically tailored to specific object structures, and achieving comparable goals is challenging for very simple structures. For bipartite perfect matchings, a fundamental structure in combinatorial optimization, one cannot hope to achieve concentration guarantees similar to independent randomized rounding, due to the well-known example introduced in Example 4.1 [26].

Example 4.1. *Let C_n be a cycle of length $n \in 2\mathbb{N}$. There are exactly two perfect matchings, $M^0 = \{e_1, e_3, \dots, e_{n-1}\}$ and $M^1 = \{e_2, e_4, \dots, e_n\}$. Consider the values of a linear function $a \in \{0,1\}^E$ over the edges alternate between 0 and 1, e.g., $a_e = 1$ for $e \in M^0$ and $a_e = 0$ for $e \in M^1$. The fractional matching $x \in [0,1]^E$, which takes every edge e with $x_e = 1/2$ will have a function value of $a^\mathsf{T} x = n/4$, but each of the integral matchings incurs an additive distortion of*

$$|a^\mathsf{T}\mathbf{1}_{M^0} - a^\mathsf{T}x| = |a^\mathsf{T}\mathbf{1}_{M^1} - a^\mathsf{T}x| = \frac{n}{4} = \Omega(n).$$

This is much higher than the bound of $O(\sqrt{n})$ that holds with high probability if each edge is picked independently with probability $1/2$.

If one considers b-matchings or other more general assignment problems, however, then there are non-trivial guarantees that can be achieved with dependent rounding. Gandhi et al. [50] show that between any two edges incident to the same vertex, they can establish *negative correlation*. Also, their algorithm has the natural property of *marginal preservation*, which means that the probability of $X_e = 1$ is equal to the fractional value x_e for each variable X_e. Together, this implies strong concentration guarantees at least for linear functions on the incident edges of each vertex. Consider a bipartite graph $G = (A \cup B, E)$. Let $\text{supp}(a) = \{e \in E \mid a_e \neq 0\}$, for any vector $a \in [0,1]^E$. Also, $\delta(v) = \{e \in E \mid v \text{ is incident to } e\}$ for any $v \in A \cup B$. The following proposition is a consequence of their result.

Proposition 4.1. *Let $G = (A \cup B, E)$ be a bipartite graph and $x \in [0,1]^E$ represent a fractional many-to-many assignment. Furthermore, let $c \in \mathbb{R}^E_{\geq 0}$, and $a_1, \ldots, a_k \in [0,1]^E$ such that for each $j \in \{1, \ldots, k\}$ there is some $v \in A \cup B$ with $\text{supp}(a_j) \subseteq \delta(v)$. Then, in randomized polynomial time, one can compute $X \in \{0,1\}^E$ satisfying with constant probability*

Cost Preservation. $c^\intercal X \leq (1+\epsilon) \cdot c^\intercal x$,

Concentration. $|a_j^\intercal x - a_j^\intercal X| \leq O(\max\{\log k, \sqrt{a_j^\intercal x \cdot \log k}\})$ *for* $j \in \{1, \ldots, k\}$,

Degree Preservation. $X(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v)) \rceil\}$.

Here, $\epsilon > 0$ is an arbitrarily small constant that influences the success probability.

A generalization to matroid intersection with a similar restriction was shown by Chekuri et al. [26]. The same work also presents a dependent rounding scheme for a single matroid that outputs a basis satisfying similar concentration bounds on linear functions without a restriction on the support. In this study, we ask the following question

Can we avoid an error in the cost for dependent rounding while maintaining comparable other guarantees?

We call a rounding algorithm *cost preserving* if it does not exceed the cost of the fractional solution we start with. Here, we focus on the stronger variant where distributions are only over objects that are cost preserving, although one might be satisfied with a sufficiently high probability of cost preservation in some cases. We have no evidence that such a relaxation would make the task significantly easier.

There are several situations where even the seemingly small cost approximation of $(1 + \epsilon)$, as derived from marginal preservation and Markov's inequality in the previously mentioned result, is unacceptable. For example, the cost of the fractional solution might come from a hard budget constraint $c^\mathsf{T} x \leq C$ in the problem. Another situation is an extension of the objective function to potentially negative values, representing, for example, the task of maximizing profit = revenue - cost. Here, Markov's inequality cannot be applied at all. Finally, an algorithm that preserves the cost provides polyhedral insights: every fractional object is in the convex hull of integer objects that marginally deviate in the considered linear functions. Similarly, the (non-integral) polytope of a relaxation is contained in an approximate integral polytope. It is easy to see that cost preservation is incompatible with marginal preservation and hence cannot be satisfied by the dependent rounding schemes above. Consider the following.

Example 4.2. *Consider $d + 1$ variables $x_0, x_1, \ldots, x_d$ of which exactly one is selected, then this can be modeled by the bases of a uniform matroid or a degree constraint in the assignment problem in Proposition 4.1. Suppose that $c_1 = \cdots = c_d = 1/(1 - 2^{-d}) > 1$ and $c_0 = 0$ where the fractional solution is given by $x_1 = \cdots = x_n = (1 - 2^{-d})/d$ and $x_0 = 2^{-d}$, leading to a cost of 1. Since marginal preservation implies $\mathbb{P}[X_0 = 1] = x_0 = 2^{-d}$, and $X_0 = 1$ is the only integral solution with cost smaller than 1, the probability that the integral solution X has a cost less than 1, i.e., $c^\top X < 1$, is 2^{-d}, which is exponentially small in d.*

Note, however, that the previous example is not an immediate counter-example to our stated goal: in this example, deterministically

choosing $X_0 = 1$ (and $X_1 = \cdots = X_d = 0$) still maintains $|a^\mathsf{T} x - a^\mathsf{T} X| \leq 1$ for every $a \in [0,1]^{d+1}$.

Our contributions

Our results are twofold. First, we show that one can obtain comparable guarantees to Proposition 4.1 while preserving costs. Note that throughout this chapter, by a randomized polynomial time algorithm we mean a probabilistic algorithm whose running time is bounded by a polynomial in the input size, independently of the outcomes of its random choices.

Theorem 4.2. *Let $G = (A \cup B, E)$ be a bipartite graph and $x \in [0,1]^E$ be a fractional many-to-many assignment. Furthermore, let $c \in \mathbb{R}^E$ and $a_1, \ldots, a_k \in [0,1]^E$ such that for each $j \in \{1, \ldots, k\}$ there is some $v \in A \cup B$ with $\operatorname{supp}(a_j) \subseteq \delta(v)$. Then, in randomized polynomial time, one can compute $X \in \{0,1\}^E$ satisfying with constant probability*

Cost Preservation. $c^\mathsf{T} X \leq c^\mathsf{T} x$,

Concentration. $|a_j^\mathsf{T} x - a_j^\mathsf{T} X| \leq O(\max\{\log k, \sqrt{a_j^\mathsf{T} x \cdot \log k}\})$ *for* $j \in \{1, \ldots, k\}$,

Degree Preservation. $X(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v)) \rceil\}$.

Note that in contrast to the previous result, we allow for negative components in the cost function c.

Second, we present a non-trivial application of our theorem to an allocation problem we call the *Budgeted Santa Claus Problem (with identical valuations)*. This is often described as Santa Claus distributing gifts to children on Christmas. Formally, there are n resources $\mathcal{R}$ (gifts) to be distributed among m players $\mathcal{P}$ (children). Each resource j has a specific value $v_j \geq 0$. Additionally, there is a total budget of $C \geq 0$, and assigning a resource j to a player i incurs a cost denoted by $c_{ij} \geq 0$. The goal is a distribution of resources among the players where the least happy player is as happy as possible, and the total cost does not exceed the budget C. Formally, we aim to find disjoint

sets $R_i \subseteq \mathcal{R}$, $i \in \mathcal{P}$, maximizing $\min_{i\in\mathcal{P}} \sum_{j\in R_i} v_j$, while ensuring that $\sum_{i\in\mathcal{P}} \sum_{j\in R_i} c_{ij} \leq C$. Note that not all resources need to be assigned. However, in the following, we show that the Budgeted Santa Claus Problem with the requirement that all items need to be assigned is not harder than the variant we study, provided that it is feasible, i.e., $\sum_{j\in\mathcal{R}} \min_{i\in\mathcal{P}} c_{ij} \leq C$.

For each resource j, adjust the cost of assigning j to any player i to $c'_{ij} = c_{ij} - \min_{i\in\mathcal{P}} c_{ij}$. This reduces the total budget by the sum of these minimum values across all resources, $C' = C - \sum_{j\in\mathcal{R}} \min_{i\in\mathcal{P}} c_{ij}$. Then we solve the reduced problem without enforcing that all the resources have to be assigned. If some resources remain unassigned, we allocate them to the players with zero cost (i.e., players $i \in \mathcal{P}$ with $c'_{ij} = 0$). This ensures that all resources are assigned without exceeding the original budget, as the reduced budget C' already accounts for these assignments. This reduction maintains the approximation ratio of the solution, as values remain the same and costs are not approximated.

It is possible to consider an even more general variant where each value v_{ij} depends on both player i and resource j, which we call the *unrelated valuations*. Mainly, we restrict ourselves to identical valuations because the understanding of unrelated valuations in the literature is rather poor, even without considering costs. In fact, much of the recent literature is focused on the so-called restricted assignment case of unrelated valuations (without costs), where $v_{ij} \in \{0, v_j\}$, meaning each resource is either desired with a value of v_j or worthless to a player. Among players who desire a particular resource, its value is the same. Our budgeted variant generalizes the restricted assignment case: observe that by setting costs $c_{ij} \in \{0, 1\}$ and $C = 0$, we can restrict the set of players to which a resource can be assigned. In a non-trivial framework, we apply our dependent rounding theorem to obtain the following approximation guarantee.

Theorem 4.3. *There is a randomized polynomial time $O(\log n)$-approximation algorithm for the Budgeted Santa Claus problem.*

Other related work for dependent rounding

Saha and Srinivasan [99] also provide a dependent rounding scheme for allocation problems, focusing on combinations of dependent and iterative rounding. Bansal and Nagarajan [10] combine dependent rounding with techniques from discrepancy theory, known as the Lovett-Meka algorithm [78]. They prove that one can round a fractional independent set (or basis) of a matroid to an integral one, while maintaining comparable concentration guarantees to both Lovett-Meka and Chernoff-type bounds. We note that Bansal and Nagarajan also integrate costs in their framework, but they make the assumption that the costs are polynomially bounded, which is inherently different from our setting (apart from the fact that they consider matroids).

Another well-known dependent rounding scheme is the maximum entropy rounding developed by Asadpour, Goemans, Mądry, Gharan and Saberi [5]. This is used to sample a spanning tree, i.e., a basis of a particular matroid, while guaranteeing negative correlation properties and therefore Chernoff-type concentration. This result led to the first improvement over the longstanding approximation rate of $\Theta(\log n)$ for the asymmetric traveling salesman problem (ATSP). However, all algorithms above guarantee marginal preservation, which means they cannot guarantee cost preservation.

At least superficially related to our work is the literature on multi-budgeted independence systems [27, 56]. Here, the goal is to find objects of certain structures, e.g., matchings or independent sets of matroids, subject to several (potentially hard) packing constraints of the form $a^\intercal x \leq b$ for some $a \in \mathbb{R}^n_{\geq 0}$, $b \in \mathbb{R}_{\geq 0}$. This can also be used to model cost preservation alike to our results. Chekuri, Vondrak, and Zenklusen [27] and Gradoni and Zenklusen [56] show various positive results in a similar spirit to ours. These results, however, are restricted to downward-closed structures where, for a given solution formed by a set of elements, all subsets are valid solutions as well. For example, Chekuri, Vondrak, and Zenklusen achieve strong concentration results for randomized rounding on

matchings, but this relies on dropping edges in long augmenting paths or cycles in order to reduce dependencies. Gradoni and Zenklusen [56] give a rounding algorithm for a constant number of hard budgets, but this requires rounding down all components of a fractional solution. Hence, these results are unable to handle instances like matroid basis constraints, perfect matching constraints, or degree preservation as in Theorem 4.2.

Other related work for the Santa Claus problem

Omitting the costs in the variant we study, the problem becomes significantly easier and admits an EPTAS, see e.g. [68], which relies on techniques that contrast with the ones that are relevant to us. As mentioned before, the problem with costs generalizes the restricted assignment variant and therefore inherits the approximation hardness of $2 - \epsilon$ due to [11]. Here and in the following, we use restricted assignment synonymous with the variant without costs, but $v_{ij} \in \{0, v_j\}$. Bansal and Srividenko [11] developed a randomized rounding algorithm for the restricted assignment. Normally, this would lead to a similar logarithmic approximation rate as ours (for the problem without costs), but they show that combining it with the Lovász Local Lemma yields an even better rate of $O(\log\log m/\log\log\log m)$. Using similar techniques, the rate was improved to a constant by Feige [47]. Note that this randomized rounding uses intricate preprocessing that violates the marginal preserving property and thus cannot even maintain the cost of a solution in expectation. Based on local search, there is also a combinatorial approach, see e.g., [9, 4, 2], which yields a (better) constant approximation for restricted assignment. However, it is not at all clear how costs could be integrated in this framework.

Finally, a classical algorithm by Shmoys and Tardos [101] gives an additive guarantee, where the rounded integral solution is only worse by the maximum value $v_{\max} = \max_{ij} v_{ij}$. Therefore, it even works in the unrelated case without increasing the cost. Notably, they state this result for the dual of minimizing the maximum value, namely the

Generalized Assignment Problem. The mentioned guarantee for Santa Claus is followed by a trivial adaption, see Lemma 4.14. Although very influential, this is the only technique we are aware of which considers the problem with costs. Unfortunately, this additive guarantee does not lead to a multiplicative guarantee, since the optimum may be lower than $v_{\max}$. In fact, it is well known that the linear programming relaxation used in [101] has an unbounded integrality gap even for restricted assignment [11]. Hence, one cannot hope to improve this by a simple modification. Nevertheless, this algorithm forms an important subprocedure in our result.

Notation

First, we introduce some necessary notation. Let $S \in \{0,1\}^E$ be an edge set in a bipartite graph $G = (A \cup B, E)$, where for every edge $e \in E$, $S_e = 1$ if e belongs to S, and $S_e = 0$, otherwise. For any edge subset $F \subseteq E$, we define $S(F) = \sum_{e \in F} S_e$. We use the same notation for fractional vectors. For any $x \in [0,1]^E$ and any edge subset $F \subseteq E$, we define $x(F) = \sum_{e \in F} x_e$. Let P be the convex hull of degree preserving edge sets $S \in [0,1]^E$. Moreover, for any $v \in A \cup B$ define $\delta(v) = \{e \in E \mid v \text{ is incident to } e\}$. For the sake of simplicity, we use the shorthand notation $[q] = \{1, \dots, q\}$ for any $q \in \mathbb{N}$. Furthermore, for any vector $a \in [0,1]^E$, the support of a is denoted by $\operatorname{supp}(a) = \{e \in E \mid a_e \neq 0\}$.

4.2 Budgeted Dependent Rounding

This section introduces a dependent randomized rounding procedure, which produces an integral solution satisfying certain concentration guarantees, while preserving the cost and the degree of the fractional solution. The following theorem summarizes the formal properties.

Theorem 4.2. *Let $G = (A \cup B, E)$ be a bipartite graph and $x \in [0,1]^E$ be a fractional many-to-many assignment. Furthermore, let $c \in \mathbb{R}^E$ and $a_1, \dots, a_k \in [0,1]^E$ such that for each $j \in \{1, \dots, k\}$ there is some $v \in A \cup B$ with $\operatorname{supp}(a_j) \subseteq \delta(v)$. Then, in randomized polynomial time, one can compute $X \in \{0,1\}^E$ satisfying with constant probability*

Cost Preservation. $c^\mathsf{T} X \leq c^\mathsf{T} x$,

Concentration. $|a_j^\mathsf{T} x - a_j^\mathsf{T} X| \leq O(\max\{\log k, \sqrt{a_j^\mathsf{T} x \cdot \log k}\})$ *for* $j \in \{1, \ldots, k\}$,

Degree Preservation. $X(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v)) \rceil\}$.

Throughout this section, the proofs of the technical lemmas are deferred to Subsection 4.2.1. An oversimplified outline of our algorithm is as follows: imagine x is the average of two integral edge sets, then the result can be shown by decomposing the symmetric difference of both edge sets into cycles and paths. We reduce to this case by starting with many edge sets and iteratively merging pairs of them in a tree-like manner.

In order to find the initial integral edge sets, we compute a representation of x (or rather another similar assignment y') that is a convex combination of degree preserving edge sets such that its scalars satisfy a certain level of discreteness. Let P be the convex hull of degree preserving edge sets $S \in \{0,1\}^E$, that is, those S that satisfy for all $v \in A \cup B$

$$S(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v) \rceil\}.$$

It can be shown that x is contained in P and, in particular, x is a convex combination of degree preserving sets. In the following lemma, we show something even stronger: there exists a fractional assignment y at least as good as x, which is the convex combination of only a few edge sets and has a few fractional variables in the support of each constraint.

Lemma 4.4. *There exists a convex combination* $y = \sum_{i \in [k]} \lambda_i S_i$ *where* $\lambda_i \in [0,1]$ *and* $\sum_{i \in [k]} \lambda_i = 1$ *and* $S_i \in P \cap \{0,1\}^E$ *with*

$$c^\mathsf{T} y \leq c^\mathsf{T} x \tag{4.1}$$

$$a_j^\mathsf{T} y = a_j^\mathsf{T} x \qquad \forall j \in \{1, \ldots, k\} \tag{4.2}$$

$$|\{e \in \delta(v) \mid y_e \notin \{0,1\}\}| \leq 2k \qquad \forall v \in A \cup B \tag{4.3}$$

Considering y as a vertex solution of a linear program, the proof follows from analyzing the structure of polytope P. For our algorithm, however, the scalars λ_i are not discrete enough. Hence, we use the following lemma to round y to a more discrete assignment y'.

Lemma 4.5. *Let $\ell \in \mathbb{N}_{\geq 0}$ and $y = \sum_{i\in[k]} \lambda_i S_i$ where $\lambda_i \in [0,1]$, $\sum_{i\in[k]} \lambda_i = 1$, and $S_i \in \{0,1\}^E$. In polynomial time, we can compute $y' = \sum_{i\in[k]} \lambda'_i S_i$ where $\sum_{i\in[k]} \lambda'_i = 1$ and*

$$\lambda'_i \in \frac{1}{2^\ell} \cdot \mathbb{Z}_{\geq 0}, \qquad \forall i \in \{1,\dots,k\} \tag{4.4}$$

$$\lambda'_i = \lambda_i, \qquad \forall i \in \{1,\dots,k\}, \lambda_i \in \{0,1\} \tag{4.5}$$

$$|y_e - y'_e| \leq k \cdot \frac{1}{2^\ell}, \qquad \forall e \in E \tag{4.6}$$

$$c^\mathsf{T} y' \leq c^\mathsf{T} y. \tag{4.7}$$

We prove this lemma by constructing a flow network and the standard argument that integral capacities imply the existence of an integral min-cost circulation.

Notably, this is the first time we incur a small error for the linear functions a_j while the cost is preserved. More precisely, we use the lemma with $\ell := \lceil 2\log(2k) \rceil$. For all $e \in E$, from Equation (4.6) follows that

$$|y_e - y'_e| \leq k \cdot \frac{1}{2^\ell} \leq \frac{1}{2k}. \tag{4.8}$$

Thus, the linear functions also slightly change. Using Equations (4.3) and (4.8), it holds that for $j \in [k]$

$$|a_j^\mathsf{T} x - a_j^\mathsf{T} y'| = |a_j^\mathsf{T} y - a_j^\mathsf{T} y'| \leq \sum_{e\in E} (a_j)_e |y'_e - y_e| \leq 1. \tag{4.9}$$

Since y' is a convex combination of (integral) degree preserving sets in P, we have $y' \in P$. In other words, the scalar rounding in Lemma 4.5 does in fact preserve the degree of y.

Next, we construct a complete binary tree $\mathcal{T}$ with levels $0, 1, \ldots, \ell$, where each node will be labeled with an edge set. When the algorithm finishes, the label of the root will be $X \in \{0,1\}^E$ and satisfy the properties stated in Theorem 4.2. In the following, we describe how the algorithm TreeMerge creates the labels on $\mathcal{T}$. The lowest level ℓ represents the fractional assignment $y' = \sum_{i \in [k]} \lambda'_i S_i$ where $S_i \in P \cap \{0,1\}^E$ are degree preserving edge sets. As we can write $\lambda'_i = h_i / 2^\ell$ for some $h_i \in \mathbb{Z}_{\geq 0}$ and all h_i sum to 2^ℓ, we can naturally label h_i leaves of level ℓ with S_i for all $i \in [k]$. Thus, y' is the average of all labels of level ℓ. For all $j \in \{0, \ldots, \ell - 1\}$, the labels of level j are derived from those in level $j+1$ such that each node's label is closely related to those of its two children. Similar to the last level, each level j represents a (fractional) edge set y'_j by taking the average of all labels in this level.

One of the central goals in the construction of y'_j is to guarantee $c^\mathsf{T} y'_j \leq c^\mathsf{T} y'_{j+1}$. We achieve this by creating two complementary labelings of level j and selecting the better of the two. Denote the labels of level $j+1$ by S_{2i-1}, S_{2i} for all $i \in \{1, \ldots, 2^j\}$. Here, each pair S_{2i-1}, S_{2i} represents the children of the i-th node in level j. For node i, we construct two potential labels $T_i, T'_i \in \{0,1\}^E$ using a random procedure with the following guarantees.

Lemma 4.6. *Let $S_1, S_2 \in \{0,1\}^E$. There exists a random polynomial time procedure that constructs two random edge sets $T, T' \in \{0,1\}^E$ with the following properties.*

- *It holds that $T + T' = S_1 + S_2$ and $\mathbb{E}(T) = \mathbb{E}(T') = (S_1 + S_2)/2$.*
- *For all $v \in A \cup B$ it holds that $T(\delta(v)), T'(\delta(v)) \in \{\lfloor (S_1(\delta(v)) + S_2(\delta(v)))/2 \rfloor, \lceil (S_1(\delta(v)) + S_2(\delta(v)))/2 \rceil\}$.*
- *For all $v \in A \cup B$ and all $e \in \delta(v)$, there is at most one edge $e' \in \delta(v) \setminus \{e\}$ such that T_e depends on $T_{e'}$. Likewise, there is at most one edge $e' \in \delta(v) \setminus \{e\}$ such that T'_e depends on $T'_{e'}$.*

The lemma can be derived in two steps. First, decompose the symmetric difference of S_1 and S_2 into cycles and paths. Second, for each of the cycles and paths, randomly select one of the alternating edge sets for T with probability $1/2$, and the other for T'. This random process is similar to other dependent rounding approaches, e.g. [26, 50], except that we also store T' that contains the "opposite" to every decision in T.

We create one fractional assignment from the random edge sets T_i, $i \in \{1, \dots, 2^j\}$, and one from T'_i, $i \in \{1, \dots, 2^j\}$, and pick the lower cost assignment for y'_j. Formally, let

$$z_j = \frac{1}{2^j} \sum_{i \in [2^j]} T_i \qquad \text{and} \qquad z'_j = \frac{1}{2^j} \sum_{i \in [2^j]} T'_i$$

From the fact that $T_i + T'_i = S_{2i-1} + S_{2i}$, it immediately follows that $(z_j + z'_j)/2 = y'_{j+1}$. If $c^\mathsf{T} z_j \leq c^\mathsf{T} z'_j$, set $y'_j = z_j$. Otherwise, $y'_j = z'_j$. Consequently, we have that

$$c^\mathsf{T} y'_j \leq (c^\mathsf{T} z_j + c^\mathsf{T} z'_j)/2 = c^\mathsf{T} y'_{j+1}. \tag{4.10}$$

We determine the labels of level j by picking either T_i (if z_j was chosen) or T'_i (if z'_j was chosen). Repeating the procedure for all $j \in \{\ell-1, \dots, 0\}$ results in a label for the root node that is identical to y'_0. We conclude by setting $X = y'_0$.

As a last step before proving the main theorem, we bound how much the linear functions a_j can change in each level.

Lemma 4.7. *Let $v \in A \cup B$ and $a \in [0,1]^E$ with $\operatorname{supp}(a) \subseteq \delta(v)$. Let $y'_{j+1} \in [0,1]^E$ be the fractional solution of the $(j+1)$-th level of* TreeMerge *and $y'_j \in [0,1]^E$ that of the j-th level. Let $t = 132 \ln k$. Then with probability at least $1 - 1/k^{10}$, it holds that*

$$|a^\mathsf{T} y'_j - a^\mathsf{T} y'_{j+1}| \leq 2^{-j/2} \left(t + \sqrt{a^\mathsf{T} y'_{j+1} \cdot t} \right). \tag{4.11}$$

This lemma follows from a standard Chernoff bound. We are now in the position to prove Theorem 4.2 using the lemmas above.

Proof of Theorem 4.2. Let $\ell = \lceil 2\log(2k) \rceil$. As explained throughout Section 4.2, we use Lemmas 4.4 and 4.5 to obtain a fractional degree preserving assignment $y' \in P$ with $c^\mathsf{T} y' \leq c^\mathsf{T} x$. Note that the rounding of x to y' marginally changes the linear function values, but we are able to maintain $|a_j^\mathsf{T} y' - a_j^\mathsf{T} x| \leq 1$, see Equation (4.9). Afterwards, we use the algorithm TreeMerge to construct a complete binary tree $\mathcal{T}$ with $\ell + 1$ levels and corresponding fractional solutions $y'_\ell = y', y'_{\ell-1}, \ldots, y'_0 = X$. It remains to show that X satisfies all three properties from the theorem. By construction, more precisely Equation (4.10), it holds that

$$c^\mathsf{T} X = c^\mathsf{T} y'_0 \leq \cdots \leq y'_\ell = c^\mathsf{T} y' \leq c^\mathsf{T} x.$$

Thus, the cost is preserved. Next, we will show that the rounding of x to X also preserves the degree. Due to Lemmas 4.4, 4.5 and 4.8, we have that $y' = \sum_{i \in [k]} \lambda'_i S_i$, where $S_i \in P \cap \{0,1\}^E$ form the labels of level ℓ of $\mathcal{T}$. For all $i \in \{1, \ldots, k\}$, the fact that $S_i \in P$ implies

$$S_i(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v)) \rceil\}. \tag{4.12}$$

By induction over the tree $\mathcal{T}$, we show that all labels and, in particular, X are indeed degree preserving. Equation (4.12) proves the base case. Let S_1, S_2 be the labels for two children of some node in $\mathcal{T}$ and T, T' be the two potential labels for the said node (derived using Lemma 4.6). From the second property of Lemma 4.6 directly follows that for all $v \in A \cup B$

$$T(\delta(v)), T'(\delta(v)) \in \{\lfloor \frac{1}{2} S_1(\delta(v)) + \frac{1}{2} S_2(\delta(v)) \rfloor, \lceil \frac{1}{2} S_1(\delta(v)) + \frac{1}{2} S_2(\delta(v)) \rceil\}. \tag{4.13}$$

By induction hypothesis, S_1 and S_2 are degree preserving, so

$$\lfloor \frac{1}{2} S_1(\delta(v)) + \frac{1}{2} S_2(\delta(v)) \rfloor \geq \lfloor x(\delta(v)) \rfloor \text{ and similarly}$$
$$\lceil \frac{1}{2} S_1(\delta(v)) + \frac{1}{2} S_2(\delta(v)) \rceil \leq \lceil x(\delta(v)) \rceil.$$

This concludes the induction step. It remains to prove the concentration, i.e., that X marginally deviates from x in each of the given linear functions a_j. We apply Lemma 4.7 together with a union bound over all k linear functions and all $\ell = \lceil 2\log(2k) \rceil$ levels of $\mathcal{T}$. For sufficiently large k, we have $\ell \leq k$. Let $t = 132 \ln k$. As a consequence, with probability at least $1 - 1/k^8$, it holds for all levels $i \in \{0, \dots, \ell - 1\}$ and linear functions $a_j, j \in [k]$ that

$$|a_j^\mathsf{T} y'_{i+1} - a_j^\mathsf{T} y'_i| \leq 2^{-i/2} \Big(t + \sqrt{a_j^\mathsf{T} y'_{i+1} \cdot t} \Big). \tag{4.14}$$

For some universal constant d, we prove that for all $j \in \{1, \dots, k\}$ and all $i \in \{0, \dots, \ell\}$

$$a_j^\mathsf{T} y'_i \leq d(1 + t + a_j^\mathsf{T} x). \tag{4.15}$$

Let us first argue that this, in fact, implies the last part of the theorem.

$$\begin{aligned}
|a_j^\mathsf{T} X - a_j^\mathsf{T} x| &= |a_j^\mathsf{T} y'_0 - a_j^\mathsf{T} y'_\ell + a_j^\mathsf{T} y'_\ell - a_j^\mathsf{T} x| \\
&\leq |a_j^\mathsf{T} y'_0 - a_j^\mathsf{T} y'_\ell| + 1 \leq \sum_{i \in \{0,\dots,\ell-1\}} |a_j^\mathsf{T} y'_{i+1} - a_j^\mathsf{T} y'_i| + 1 \\
&\leq \sum_{i \in \{0,\dots,\ell-1\}} \frac{1}{(\sqrt{2})^i} \Big(t + \sqrt{a_j^\mathsf{T} y'_{i+1} \cdot t} \Big) + 1 \\
&\leq \sum_{i \in \{0,\dots,\ell-1\}} \frac{1}{(\sqrt{2})^i} \Big(t + \sqrt{d(1 + t + a_j^\mathsf{T} x) \cdot t} \Big) + 1 \\
&= O(\max\{\log k, \sqrt{a_j^\mathsf{T} x \cdot \log k}\}).
\end{aligned}$$

This holds using the triangle inequality, geometric series, and Equations (4.9), (4.14) and (4.15).

Finally, we prove Equation (4.15). Let $j \in \{1, \dots, k\}$ and $i \in \{0, \dots, \ell - 1\}$. Let $i' \geq i$ be the minimal index such that $a_j^\mathsf{T} y'_{i'} \leq 1 + t + a_j^\mathsf{T} x$. As $a_j^\mathsf{T} y'_\ell = a_j^\mathsf{T} y' \leq 1 + a_j^\mathsf{T} x$, such i' must exist. If $i' = i$, we are done. If

$i' \neq i$, then it follows from Equation (4.14) that

$$a_j^\mathsf{T} y'_{i'-1} \leq (1 + t + a_j^\mathsf{T} x) + 2^{-(i'-1)/2}(t + (1 + t + a_j^\mathsf{T} x)) \leq 3(1 + t + a_j^\mathsf{T} x) .$$

For all $i'' \in \{i' - 1, \ldots, i\}$, we have $a_j^\mathsf{T} y'_{i''} > t$. Thus, the same equation implies

$$a_j^\mathsf{T} y'_{i''-1} \leq a_j^\mathsf{T} y'_{i''} + \frac{2}{2^{(i''-1)/2}} a_j^\mathsf{T} y'_{i''} = \left(1 + \frac{1}{2^{(i''-3)/2}}\right) a_j^\mathsf{T} y'_{i''} .$$

Using the inequality $1 + z \leq \exp(z)$ for all $z \in \mathbb{R}$, we have

$$\begin{aligned} a_j^\mathsf{T} y'_i &\leq a_j^\mathsf{T} y'_{i'-1} \cdot \prod_{i''=i'-1}^{i+1} \left(1 + \frac{1}{2^{(i''-3)/2}}\right) \\ &\leq a_j^\mathsf{T} y'_{i'-1} \cdot \exp\left(\sum_{i''=i'-1}^{i+1} \frac{1}{2^{(i''-3)/2}}\right) \\ &\leq 3e^{O(1)} \cdot (1 + t + a_j^\mathsf{T} x). \end{aligned}$$

This shows Equation (4.15) and thereby concludes the proof. □

4.2.1 Omitted Proofs of Dependent Rounding

In this subsection, we provide the proofs of the previously stated lemmas. Recall that P is the convex hull of integral degree preserving edge sets for $x \in [0,1]^E$. Let the operator $\oplus$ denote the symmetric difference (i.e., XOR) of two edge sets. We start by showing that P indeed contains x.

Lemma 4.8. *Let $q = |E|$. There exists a convex representation $x = \sum_{i\in[q]} \lambda_i S_i \in P$ where $S_i \in P \cap \{0,1\}^E$ and $\lambda_i \in [0,1]$ and $\sum_{i\in[q]} \lambda_i = 1$ such that for all $v \in A \cup B$ and $i \in \{1, \ldots, q\}$ holds that $S_i(\delta(v)) \in \{\lfloor x(\delta(v)) \rfloor, \lceil x(\delta(v)) \rceil\}$.*

Proof. We rely on a standard flow argument. To this end, construct a digraph $D_f = (V_f, A_f)$ as follows. Let $V_f = \{s, t\} \cup V$ be the set of vertices, A_s be a set of arcs directed from s to each $a \in A$, A_t be the set of arcs directed from each $b \in B$ to t, and A_E be a directed variant of E from A to B. Let $A_f = A_s \cup A_E \cup A_t \cup \{(t, s)\}$ be the set of arcs in D_f. Set the capacity interval for arc $(s, a) \in A_s$ as $[\lfloor x(\delta(a)) \rfloor, \lceil x(\delta(a)) \rceil]$ and similarly for each arc $(b, t) \in A_t$ as $[\lfloor x(\delta(b)) \rfloor, \lceil x(\delta(b)) \rceil]$. Moreover, set the capacity interval for each arc $e \in E$ to $[0, 1]$ and for the arc (t, s) to $[0, \infty]$. A function $f : A_f \to \mathbb{R}$ is called a circulation if $f(\delta^{\text{in}}(v)) = f(\delta^{\text{out}}(v))$ for each vertex $v \in V_f$.

One can naturally derive a feasible fractional circulation from the vector $x \in [0, 1]^E$: the flow from s to $a \in A$ is $x(\delta(a))$, which lies in $[\lfloor x(\delta(a)) \rfloor, \lceil x(\delta(a)) \rceil]$, the circulation from $b \in B$ to t is $x(\delta(b))$, which lies in $[\lfloor x(\delta(b)) \rfloor, \lceil x(\delta(b)) \rceil]$, the circulation on each arc $(a, b) \in A_E$ is x_{ab}, and the circulation on arc (t, s) is $x(E)$. Hence, x satisfies the capacity and flow conservation constraints. It is well known, see e.g. [100, Corollary 13.10b], that for integral capacities the set of all feasible circulations (including x) forms an integral polytope. Thus, we can rewrite the circulation above as a convex combination of integral circulations. Each of these integral circulations corresponds to a degree preserving edge set. □

Further, there always exists a comparable convex representation which has only few fractional variables in the support of each constraint.

Lemma 4.4. *There exists a convex combination $y = \sum_{i \in [k]} \lambda_i S_i$ where $\lambda_i \in [0, 1]$ and $\sum_{i \in [k]} \lambda_i = 1$ and $S_i \in P \cap \{0, 1\}^E$ with*

$$c^{\mathsf{T}} y \leq c^{\mathsf{T}} x \tag{4.1}$$

$$a_j^{\mathsf{T}} y = a_j^{\mathsf{T}} x \qquad \forall j \in \{1, \ldots, k\} \tag{4.2}$$

$$|\{e \in \delta(v) \mid y_e \notin \{0, 1\}\}| \leq 2k \qquad \forall v \in A \cup B \tag{4.3}$$

Proof. First, we argue about the structure of the edges of polytope P. Let $S, T \in \{0,1\}^E$ be the two vertices at the two ends of some edge of P. We claim that $S \oplus T$ is a simple cycle or a simple path. Suppose not. If $S \oplus T$ is acyclic, let D be a maximal path in $S \oplus T$; otherwise let D be a simple cycle contained in $S \oplus T$. Note that $S \oplus D \in P$ and $T \oplus D \in P$ and both points do not lie on the edge between S and T. However, $(S+T)/2 = (T \oplus D + S \oplus D)/2$ contradicts that S and T are connected by an edge.

Let y be an optimal vertex solution of the linear program

$$\begin{aligned} \min\, & c^\mathsf{T} y \\ & a_j^\mathsf{T} y = a_j^\mathsf{T} x && \forall j \in \{1,\dots,k\} \\ & y \in P \end{aligned}$$

Due to lemma 4.8, the linear program is feasible. The solution y must lie on a face F of P with dimension at most k. Consider an arbitrary vertex S of F. Furthermore, let $T_1, \dots, T_h$, $h \leq k$, be the vertices of F such that there is an edge between each T_i and S. Thus, we can write $y = S + \sum_{i\in[h]} \lambda_i (T_i - S)$ where $\lambda_1, \dots, \lambda_h \geq 0$. By our previous argument, $T_i \oplus S$ is a simple cycle or path for each $i \in \{1,\dots,h\}$. In particular, $|(T_i \oplus S) \cap \delta(v)| \leq 2$ for each $v \in A \cup B$. This implies that the last property holds for y. □

Allowing a small rounding error, there always exists a convex representation where the scalars are integer multiples of a power of two.

Lemma 4.5. *Let $\ell \in \mathbb{N}_{\geq 0}$ and $y = \sum_{i\in[k]} \lambda_i S_i$ where $\lambda_i \in [0,1]$, $\sum_{i\in[k]} \lambda_i = 1$, and $S_i \in \{0,1\}^E$. In polynomial time, we can compute $y' = \sum_{i\in[k]} \lambda'_i S_i$*

where $\sum_{i\in[k]} \lambda'_i = 1$ *and*

$$\lambda'_i \in \frac{1}{2^\ell} \cdot \mathbb{Z}_{\geq 0}, \qquad \forall i \in \{1, \ldots, k\} \tag{4.4}$$

$$\lambda'_i = \lambda_i, \qquad \forall i \in \{1, \ldots, k\}, \lambda_i \in \{0, 1\} \tag{4.5}$$

$$|y_e - y'_e| \leq k \cdot \frac{1}{2^\ell}, \qquad \forall e \in E \tag{4.6}$$

$$c^\mathsf{T} y' \leq c^\mathsf{T} y. \tag{4.7}$$

Proof. Using a standard flow argument, we construct a digraph $D_f = (V_f, A_f)$ that represents a circulation network. For each arc $a \in A_f$, we adjust a capacity interval such that every feasible circulation corresponds to a solution that is "close" to x and preserves the cost. Let $V_f = \{s, t\} \cup \{v_i : i \in [k]\}$ where nodes v_i correspond to each S_i. The set of arcs A_f contains arcs (t, s), (s, v_i), and (v_i, t) for $i \in [k]$. A function $f : A_f \to \mathbb{R}$ is called a circulation if $f(\delta^{\text{in}}(v)) = f(\delta^{\text{out}}(v))$ for each vertex $v \in V_f$. Set the capacity interval of arc (t, s) to $[2^\ell, 2^\ell]$ and the one of arcs (s, v_i) and (v_i, t) to $[\lfloor \lambda_i 2^\ell \rfloor, \lceil \lambda_i 2^\ell \rceil]$ for $i \in [k]$. Moreover, define a linear cost function $\text{cost}(s, v_i) = \sum_{e\in E} c_e(S_i)_e$, the contribution of S_i to the total cost. Set $\text{cost}(t, s) = \text{cost}(v_i, t) = 0$. The scalars λ_i induce a natural circulation f: on arcs (s, v_i) and (v_i, t), we send a flow of $\lambda_i 2^\ell$ and on arc (t, s), we send a flow of 2^ℓ.

For integral capacities, the set of feasible circulations forms an integral polytope, see e.g. [100, Corollary 13.10b]. Thus, there exists a minimum cost integral circulation. Let $\bar{f}$ be this integral circulation. We define $\lambda'_i = \bar{f}(s, v_i)/2^\ell$ for all $i \in [k]$, then

$$\sum_{i\in[k]} \lambda'_i = \sum_{i\in[k]} \frac{\bar{f}(s, v_i)}{2^\ell} = \frac{1}{2^\ell} \cdot \bar{f}(t, s) = \frac{1}{2^\ell} \cdot 2^\ell = 1.$$

Consequently, λ'_i creates a valid convex combination for y' where each

λ'_i is a multiple of $1/2^\ell$. For all $e \in E$

$$\begin{aligned}|y_e - y'_e| &= \left|\sum_{i\in[k]} \lambda_i (S_i)_e - \sum_{i\in[k]} \lambda'_i (S_i)_e\right| \\ &= \sum_{i\in[k]} |\lambda_i - \lambda'_i| \cdot (S_i)_e \leq \sum_{i\in[k]} \frac{1}{2^\ell} = k \cdot \frac{1}{2^\ell}.\end{aligned}$$

Since the integer circulation minimizes the total cost, we have

$$\begin{aligned}c^\mathsf{T} y' &= \sum_{e\in E} c_e \sum_{i\in[k]} \lambda'_i (S_i)_e \\ &= \frac{1}{2^\ell} \sum_{e\in E} \bar{f}(e) \cdot \mathrm{cost}(e) \leq \frac{1}{2^\ell} \sum_{e\in E} f(e) \cdot \mathrm{cost}(e) = c^\mathsf{T} x.\end{aligned} \tag{4.16}$$

Moreover, constructing D_f and solving minimum cost flow can be done in polynomial time [100]. □

The TreeMerge algorithm described in Section 4.2 uses the following lemma to construct new random edge sets while preserving the degree of the former sets.

Lemma 4.6. *Let $S_1, S_2 \in \{0,1\}^E$. There exists a random polynomial time procedure that constructs two random edge sets $T, T' \in \{0,1\}^E$ with the following properties.*

- *It holds that $T + T' = S_1 + S_2$ and $\mathbb{E}(T) = \mathbb{E}(T') = (S_1 + S_2)/2$.*
- *For all $v \in A \cup B$ it holds that $T(\delta(v)), T'(\delta(v)) \in \{\lfloor (S_1(\delta(v)) + S_2(\delta(v)))/2 \rfloor, \lceil (S_1(\delta(v)) + S_2(\delta(v)))/2 \rceil\}$.*
- *For all $v \in A \cup B$ and all $e \in \delta(v)$, there is at most one edge $e' \in \delta(v) \setminus \{e\}$ such that T_e depends on $T_{e'}$. Likewise, there is at most one edge $e' \in \delta(v) \setminus \{e\}$ such that T'_e depends on $T'_{e'}$.*

Proof. For the randomized construction of T and T', we distinguish whether an edge is present in the symmetric difference $S_1 \oplus S_2$ or not. For any edge $e \in E$ where $(S_1)_e = (S_2)_e$, we set $T_e = ((S_1)_e + (S_2)_e)/2$ and $T'_e = ((S_1)_e + (S_2)_e)/2$. Next, partition the edges in $S_1 \oplus S_2$ into simple paths and cycles with $E_1 \dot\cup \cdots \dot\cup E_k = S_1 \oplus S_2$ with the following property: each odd-degree vertex is the endpoint of exactly one path and no path ends in an even-degree vertex. This easily follows from iteratively removing cycles and maximal paths.

Let $i \in \{1, \ldots, k\}$. Choose $C_i \dot\cup D_i = E_i$ such that no two edges in C_i or in D_i are adjacent. This is possible by alternatingly assigning edges to C_i and D_i, as E_i is an even length cycle or a path. We make a uniform binary random decision $R_i \in \{0, 1\}$, which is independent of all $R_{i'}$, $i' \neq i$. This random variable indicates whether C_i is assigned to T or T' (D_i is then assigned to the other edge set). More precisely, for each $e \in E_i$ we set

$$T_e = \begin{cases} (C_i)_e & \text{if } R_i = 0, \\ (D_i)_e & \text{if } R_i = 1, \end{cases} \qquad \text{and} \qquad T'_e = \begin{cases} (D_i)_e & \text{if } R_i = 0, \\ (C_i)_e & \text{if } R_i = 1. \end{cases} \tag{4.17}$$

In particular, $T_e + T'_e = (C_i)_e + (D_i)_e = 1 = (S_1)_e + (S_2)_e$. Since both outcomes $R_i = 1$ and $R_i = 0$ have probability $1/2$, it holds that $\mathbb{E}(T_e) = \mathbb{E}(T_e) = ((C_i)_e + (D_i)_e)/2 = ((S_1)_e + (S_2)_e)/2$ for all $e \in S \oplus T$. Thus, it follows that $\mathbb{E}(T) = \mathbb{E}(T') = (S_1 + S_2)/2$. Let $\mathcal{F} = \{\delta(v) \mid v \in A \cup B\}$. Considering each vertex in a simple path or cycle is incident to at most two edges, we have $|F \cap E_i| \leq 2$ for all $F \in \mathcal{F}$ and $i \in \{1, \ldots, k\}$. As the random choice for each path and cycle is independent, it holds that for all $F \in \mathcal{F}$ and $e \in F$, there is at most one edge $e' \in F$ with $e \neq e'$ such that T_e depends on $T_{e'}$ (and the same for T').

It remains to show that $T(F), T'(F) \in \{\lfloor (S_1(F) + S_2(F))/2 \rfloor, \lceil (S_1(F) + S_2(F))(2) \rceil\}$. To this end, we distinguish whether a vertex $v \in A \cup B$ has even or odd degree. If $|\delta(v)|$ is even, then v is always incident to exactly one edge in C_i and one in D_i, for each $i \in [k]$. Hence, $T(F) = T'(F) = |\delta(v)|/2 = (S_1(F) + S_2(F))/2$. If $|\delta(v)|$ is odd, then there is exactly one path that ends in v due to the choice of decomposition into

cycles and paths. Consequently, the arguments above apply to all but one edge in $\delta(v)$, and therefore we have $|T(F) - T'(F)| = 1$. This implies the claim. □

The last lemma bounds the change of a linear function between two consecutive levels of TreeMerge.

Lemma 4.7. *Let $v \in A \cup B$ and $a \in [0,1]^E$ with $\operatorname{supp}(a) \subseteq \delta(v)$. Let $y'_{j+1} \in [0,1]^E$ be the fractional solution of the $(j+1)$-th level of TreeMerge and $y'_j \in [0,1]^E$ that of the j-th level. Let $t = 132 \ln k$. Then with probability at least $1 - 1/k^{10}$, it holds that*

$$|a^\mathsf{T} y'_j - a^\mathsf{T} y'_{j+1}| \leq 2^{-j/2} \left(t + \sqrt{a^\mathsf{T} y'_{j+1} \cdot t} \right). \tag{4.11}$$

Proof. Let T_i, $i \in [2^j]$, be the random edge set created on the j-th level from the edge sets S_{2i-1},S_{2i}, $i \in [2^j]$ of the $(j+1)$-th level. Recall that the procedure from lemma 4.6 actually creates two alternative edge sets T_i, T'_i of which TreeMerge selects just one. However, the solution derived from T_i satisfies Equation (4.11) if and only if the one from T'_i does. Hence, it suffices to show it for one of the two. Further, for the sake of convenience, we analyze the scaled expression $|2^j \cdot a^\mathsf{T} y'_j - 2^j \cdot a^\mathsf{T} y'_{j+1}|$ instead of $|a^\mathsf{T} y'_j - a^\mathsf{T} y'_{j+1}|$. Due to Lemma 4.6, it holds that $\mathbb{E}(T_i) = (S_{2i-1} + S_{2i})/2$. Thus,

$$\mathbb{E}[2^j \cdot a^\mathsf{T} y'_j] = a^\mathsf{T} \Big(\sum_{i \in [2^j]} \mathbb{E}[T_i] \Big) = \frac{1}{2} a^\mathsf{T} \Big(\sum_{i \in [2^{j+1}]} S_i \Big) = 2^j \cdot a^\mathsf{T} y'_{j+1}.$$

Since TreeMerge independently constructs the T_i on the j-th level, any two random edge sets T_i and T_{i+1} are independent. Moreover, it follows from lemma 4.6 that for all T_i and $e \in \delta(v)$, there exists at most one edge $e' \in \delta(v)$ such that $(T_i)_e$ depends on $(T_i)_{e'}$. Consequently, there exists a partition of $\delta(v)$ into P_1 and P_2 such that all variables $(T_i)_e$, $e \in P_1$, as well as $(T_i)_e$, $e \in P_2$ are independent. Let $u = \sum_{i \in [2^j]} U_i \in [0,1]^E$

and $w = \sum_{i \in [2^j]} W_i \in [0,1]^E$ where $U_i, W_i \in \{0,1\}^E$ are defined for all $e \in E$ as

$$(U_i)_e = \begin{cases} 1, & \text{if } e \in \delta(v) \text{ and } (T_i)_e \in P_1 \\ 0, & \text{otherwise} \end{cases}$$

and

$$(W_i)_e = \begin{cases} 1, & \text{if } e \in \delta(v) \text{ and } (T_i)_e \in P_2 \\ 0, & \text{otherwise.} \end{cases}$$

Hence, we have $2^j \cdot a^\mathsf{T} y'_j = a^\mathsf{T}(u + w)$. Next, we show that

$$\mathbb{P}\left[|a^\mathsf{T}u - \mathbb{E}[a^\mathsf{T}u]| > \frac{1}{2}t + \frac{1}{2}\sqrt{\mathbb{E}[a^\mathsf{T}u] \cdot t}\right] < \frac{1}{2k^{10}}. \tag{4.18}$$

For brevity, define $\mu = \mathbb{E}[a^\mathsf{T}u]$. We distinguish two cases. If $t > 4\mu$, then set $\delta = t/(4\mu) > 1$. Notice that $a^\mathsf{T}u$ is the sum of independent variables, each contained in $[0,1]$. We have

$$\mathbb{P}\left[a^\mathsf{T}u < \mu - t/4\right] = 0$$

It follows from a standard Chernoff bound (see Equation (4.24)) that

$$\begin{aligned} \mathbb{P}\left[a^\mathsf{T}u > \mu + t/4\right] &= \mathbb{P}\left[a^\mathsf{T}u > (1+\delta)\mu\right] \\ &\leq \exp(-\frac{1}{3}\delta\mu) \\ &= \exp(-11 \ln k) \\ &\leq \frac{1}{2k^{10}}. \end{aligned}$$

If $t \leq 4\mu$, then set $\delta = 1/2 \cdot \sqrt{t/\mu} \in (0,1]$. Again by a Chernoff bound

(see Equation (4.23)), we have

$$\begin{aligned}\mathbb{P}\left[|a^{\mathsf{T}}u-\mu|>\frac{1}{2}\sqrt{\mu t}\right] &= \mathbb{P}\left[|a^{\mathsf{T}}u-\mu|>\delta\mu\right]\\ &\leq 2\exp(-\frac{1}{3}\delta^2\mu)\\ &= 2\exp(-11\ln k)\leq\frac{1}{2k^{10}}.\end{aligned}$$

By symmetry, Equation (4.18) holds for w as well. We conclude that with probability at least $1-1/k^{10}$, we have

$$\begin{aligned}|2^j a^{\mathsf{T}}y'_j-2^j a^{\mathsf{T}}y'_{j+1}| &\leq |a^{\mathsf{T}}u-\mathbb{E}[a^{\mathsf{T}}u]|+|a^{\mathsf{T}}w-\mathbb{E}[a^{\mathsf{T}}w]|\\ &\leq \frac{1}{2}t+\frac{1}{2}\sqrt{\mathbb{E}[a^{\mathsf{T}}u]\cdot t}+\frac{1}{2}t+\frac{1}{2}\sqrt{\mathbb{E}[a^{\mathsf{T}}w]\cdot t}\\ &\leq t+\sqrt{\mathbb{E}[2^j\cdot a^{\mathsf{T}}y'_j]\cdot t}\\ &\leq 2^{j/2}(t+\sqrt{a^{\mathsf{T}}y'_{j+1}\cdot t}).\end{aligned}$$

□

4.2.2 Chernoff Bounds

Concentration inequalities quantify how tightly a random quantity is expected to fluctuate around its mean. For sums of independent *bounded* random variables, the most commonly used tools are

Let $X=\sum_{i=1}^{n}X_i$ be the sum of independent bounded random variables with independent $X_i\in[0,1]$ and let $\mu=\mathbb{E}[X]$. The two most frequently used variations are

- **Additive** (Hoeffding) bounds, which control $\mathbb{P}[X-\mathbb{E}[X]>\lambda]$ with a rate proportional to λ^2,
- **Multiplicative** (Chernoff) bounds, which control relative deviations $\mathbb{P}[X>(1+\delta)\mathbb{E}[X]]$ with a rate proportional to δ^2 (for moderate δ) or δ (for large δ).

Hence, Chernoff tail bounds give exponentially small probabilities that a sum of independent / bounded random variables deviates from its mean. Throughout the proof of Lemma 4.7 we need tail estimates for the sum of independent, $[0,1]$–bounded random variables. Below, we state precisely the two Chernoff bounds that are used in our analysis and reproduce short proofs for completeness [80].

Theorem 4.9 (Mitzenmacher and Upfal[80], Theorem 20)**.** *Let $X_1, \ldots, X_n$ be independent Poisson trials such that $\mathbb{P}[X_i] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. Then the following Chernoff bounds hold*

1. *For any $\delta > 0$,*

$$\mathbb{P}[X \geq (1+\delta)\mu] < \Big(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\Big)^{\mu}. \tag{4.19}$$

2. *For $0 < \delta \leq 1$,*

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\,(-\frac{1}{3}\delta^2\mu). \tag{4.20}$$

Proof. Applying Markov's inequality, we have that for any $t > 0$,

$$\mathbb{P}[X \geq (1+\delta)\mu] = \mathbb{P}\left[(e^{tX} \geq e^{t(1+\delta)\mu}\right] \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1+\delta)\mu}} \leq \frac{e^{(e^t-1)\mu}}{e^{t(1+\delta)\mu}}.$$

For any $\delta > 0$, we can set $t = \ln(1+\delta) > 0$ to get Equation (4.19)

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \Big(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\Big)^{\mu}.$$

To obtain Equation (4.20), we need to show that for $0 < \delta \leq 1$,

$$\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \leq e^{-\delta^2/3}.$$

Taking the logarithm of both sides, we obtain the equivalent condition

$$f(\delta) = \delta - (1+\delta)\ln(1+\delta) + \frac{\delta^2}{3} \leq 0.$$

Computing the derivatives of $f(\delta)$ we get

$$f'(\delta) = 1 - \frac{1+\delta}{1+\delta} - \ln(1+\delta) + \frac{2}{3}\,\delta = -\ln(1+\delta) + \frac{2}{3}\,\delta,\; f''(\delta) = -\frac{1}{1+\delta} + \frac{2}{3}.$$

We see that Since $f''(\delta) < 0$ for $0 \leq \delta < 1/2$, and $f''(\delta) > 0$ for $\delta > 1/2$. Hence, $f'(0)$ first decreases and then increases over the interval $[0,1]$. Since $f'(0) = 0$ and $f'(1) < 0$, we can conclude that $f'(\delta) \leq 0$ in the interval $[0,1]$. Since $f(0) = 0$ and $f(\delta) \leq 0$ in that interval, proving, Equation (4.20). □

Theorem 4.10 (Mitzenmacher and Upfal[80], Theorem 21)**.** *Let $X_1, \ldots, X_n$ be independent Poisson trials such that $\mathbb{P}[X_i] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. For $0 < \delta < 1$,*

1. $$\mathbb{P}[X \leq (1-\delta)\mu] \leq \Big(\frac{e^{-\delta}}{(1-\delta)^{1-\delta}}\Big)^{\mu}. \tag{4.21}$$

2. $$\mathbb{P}[X \leq (1-\delta)\mu] \leq e^{-\mu\delta^2/2}. \tag{4.22}$$

Proof. Using Markov's inequality, for any $t < 0$,

$$\mathbb{P}[X \leq (1-\delta)\mu] = \mathbb{P}(e^{tX} \geq e^{t(1-\delta)\mu}) \leq \frac{\mathbb{E}[e^{tX}]}{e^{t(1-\delta)\mu}} \leq \frac{e^{(e^t-1)\mu}}{e^{t(1-\delta)\mu}}.$$

For $0 < \delta < 1$, we set $t = \ln(1-\delta) < 0$ to get Equation (4.21).

To prove Equation (4.22), we need to show that for $0 < \delta < 1$,

$$\frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \leq e^{-\delta^2/2}.$$

Taking the logarithm of both sides, we obtain the equivalent condition

$$f(\delta) = -\delta - (1-\delta)\ln(1-\delta) + \frac{1}{2}\delta^2 \leq 0.$$

For $0 < \delta < 1$. Differentiating $f(\delta)$ we get

$$f'(\delta) = \ln(1-\delta) + \delta, \qquad f''(\delta) = -\frac{1}{1-\delta} + 1.$$

Since $f''(\delta) < 0$ on the range $(0,1)$, and $f'(0) = 0$, we have $f'(\delta) \leq 0$ in the range $[0,1)$. Therefore, $f(\delta)$ is monotonically decreasing in that interval. Since $f(0) = 0$, $f(\delta) \leq 0$ when $0 < \delta < 1$, as required. □

The following form of the Chernoff bounds is used in Lemma 4.7. Note that Corollary 4.11 is derived immediately from Equations (4.20) and (4.22).

Corollary 4.11. *Let $X_1, \ldots, X_n$ be independent Poisson trials such that $\mathbb{P}[X_i] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ $\mu = \mathbb{E}[X]$. For $0 < \delta < 1$,*

$$\mathbb{P}(|X - \mu| \geq \delta\mu) \leq 2e^{-\mu\delta^2/3}. \tag{4.23}$$

Corollary 4.12. *Let $X_1, \ldots, X_n$ be independent random variables with $X_i \in [0,1]$, and let $X = \sum_{i=1}^{n} X_i$ and $\mu = \mathbb{E}[X]$. For any $\delta \geq 1$:*

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(-\frac{1}{3}\delta\mu\right). \tag{4.24}$$

Proof. Consider Equation (4.19), taking the logarithm of both sides, we obtain

$$\ln\left(\mathbb{P}[X \geq (1+\delta)\mu]\right) \leq \mu\left(\delta - (1+\delta)\ln(1+\delta)\right).$$

Thus

$$\mathbb{P}[X \geq (1+\delta)\mu] \leq \exp\left(\mu\left(\delta - (1+\delta)\ln(1+\delta)\right)\right).$$

To prove Equation (4.24), we need to show that for $\delta \geq 1$

$$\delta - (1+\delta)\ln(1+\delta) \leq -\frac{\delta}{3}.$$

Consider the function $f(\delta) = (1+\delta)\ln(1+\delta) - 4\delta/3$. We need to show $f(\delta) \geq 0$ for $\delta \geq 1$. Differentiating $f(\delta)$ we get

$$f'(\delta) = \ln(1+\delta) + \frac{1+\delta}{1+\delta} - \frac{4}{3} = \ln(1+\delta) - \frac{1}{3}, \quad f''(\delta) = \frac{1}{1+\delta} > 0.$$

Since $f''(\delta) > 0$, $f'(\delta)$ is strictly increasing. We have $f'(1) > 0$, so $f'(\delta) \geq f'(1) > 0$ for $\delta \geq 1$. Thus, $f(\delta)$ is strictly increasing. We have $f(1) > 0$ and since $f(\delta)$ increases for $\delta \geq 1$, $f(\delta) \geq f(1) > 0$. Hence, $f(\delta) \geq 0$ for $\delta \geq 1$.

□

4.3 Application to Budgeted Santa Claus Problem

In this section, we present our approximation algorithm for the Budgeted Santa Claus Problem based on the dependent rounding scheme described in Section 4.2.

4.3.1 Linear Programming Formulation

Introducing an LP relaxation for the Budgeted Santa Claus Problem, we first reduce the problem to its decision variant. For a given threshold $T \geq 0$, the goal is to either find a solution of value T/α or determine that $\mathrm{OPT} < T$, where OPT is the optimal value of the original optimization problem. This variant is equivalent to an α-approximation algorithm by a standard binary search framework.

Based on T, intuitively thought of as the optimal value, we partition the resources into two sets by size. Set $\mathcal{B}$ consists of the *big resources* with $\mathcal{B} \coloneqq \{j \in \mathcal{R} : v_j \geq T/\alpha\}$ and set $\mathcal{S}$ consists of the *small resources* $\mathcal{S} \coloneqq \{j \in \mathcal{R} : v_j < T/\alpha\}$. We use assignment variables that indicate whether a particular resource is assigned to a particular player. For

clarity, we use different symbols for big and small resources. Let $x_{ib} \in [0,1]$ denote the portion of big resource $b \in \mathcal{B}$ that player $i \in \mathcal{P}$ receives. Similarly, denote by $z_{is} \in [0,1]$ the portion of the small resource $s \in \mathcal{S}$ that player i receives. Unlike the original problem, we allow these assignments to be fractional in the relaxation. Since naive constraints on these variables lead to an unbounded integrality gap, see e.g. [11], we use non-trivial constraints inspired by an LP formulation of Davies, Rothvoss and Zhang [42].

$$\min \sum_{i \in \mathcal{P}} \left[\sum_{b \in \mathcal{B}} c_{ib} \cdot x_{ib} + \sum_{s \in \mathcal{S}} c_{is} \cdot z_{is} \right] \tag{4.25}$$

$$\sum_{s \in \mathcal{S}} v_s \cdot z_{is} \geq T \cdot \left(1 - \sum_{b \in \mathcal{B}} x_{ib} \right) \qquad \forall i \in \mathcal{P} \tag{4.26}$$

$$z_{is} \leq 1 - \sum_{b \in \mathcal{B}} x_{ib} \qquad \forall s \in \mathcal{S}, i \in \mathcal{P} \tag{4.27}$$

$$\sum_{i \in \mathcal{P}} x_{ib} \leq 1 \qquad \forall b \in \mathcal{B} \tag{4.28}$$

$$\sum_{i \in \mathcal{P}} z_{is} \leq 1 \qquad \forall s \in \mathcal{S} \tag{4.29}$$

$$\sum_{b \in \mathcal{B}} x_{ib} \leq 1 \qquad \forall i \in \mathcal{P} \tag{4.30}$$

$$z_{is}, x_{ib} \geq 0 \qquad \forall s \in \mathcal{S}, b \in \mathcal{B}, i \in \mathcal{P} \tag{4.31}$$

Here, we make the structural assumption that in any solution, a player either receives exactly one big resource (and nothing else) or only small resources. Towards the goal of obtaining a solution of value T/α, any big resource is sufficient for any player, and receiving more would be wasteful. If there is a solution of value T, then there is also a pseudo-solution such that each player either gets exactly one big resource (and nothing else) or a value of at least T from small resources only. Note that it might be that the former type of player only has a

value of T/α. Thus, if $T \leq \mathrm{OPT}$, then the previous relaxation called $\mathrm{economical}(T)$ is feasible and has a value at most C.

The Constraints (4.28) and (4.29) describe that each big or small resource is only assigned once. Justified by earlier arguments, Constraint (4.30) ensures that each player receives at most one big resource. Considering constraint (4.27), we only need to verify that the constraint is valid for integer solutions. By our assumption, if player i receives one big resource, then it should not get any small resources, which is exactly what the constraint expresses. Conversely, if the player does not receive any big resources, the constraint is trivially satisfied. Similarly, there are two cases for Constraints (4.26). If player i receives one big resource, the constraint is trivially satisfied. Otherwise, it must receive a value of at least T in small resources. During our rounding procedure in Section 4.3.4, we essentially lose some value from small resources and can only guarantee a value of T/α for players without a big resource.

4.3.2 Technical Goals

Let (x, z) be a feasible solution to $\mathrm{economical}(T)$, introduced in the previous subsection, where x and z represent the vectors of assignment variables corresponding to big and small resources, respectively. Formally, we have $x_{ib}, z_{is} \in [0, 1]$ for $i \in \mathcal{P}, b \in \mathcal{B}, s \in \mathcal{S}$. We will define a randomized rounding procedure that constructs a distribution over the binary variables $X_{ib} \in \{0, 1\}$ and $Z_{is} \in \{0, 1\}$ describing whether a big resource $b \in \mathcal{B}$ or small resource $s \in \mathcal{S}$ is assigned to player $i \in \mathcal{P}$. For notational convenience, define $Y_i = 1 - \sum_{b \in B} X_{ib}$ as the indicator variable whether a player i *does not* get a big resource (and needs small resources). Similarly, let $y_i = 1 - \sum_{b \in \mathcal{B}} x_{ib}$ be the corresponding value to Y_i from the corresponding value from the LP variables.

Our goal is that the total cost of assignments does not exceed the budget C and the integral solution (X, Z) is an α-approximation solution with respect to the minimum value a player receives. In other words, we want to obtain an integral solution (X, Z) that satisfies the following two properties.

1. $\sum_{i\in\mathcal{P}}\left[\sum_{b\in\mathcal{B}} c_{ib}\cdot X_{ib}+\sum_{s\in\mathcal{S}} c_{is}\cdot Z_{is}\right]\leq C$.
2. Every player receives resources of value at least T/α with high probability.

We first apply our dependent rounding scheme to round the assignment of big resources to an integral one. To cover the players that do not receive big resources, i.e., those with $Y_i = 1$, we need to change the assignment of small resources as well. Initially, some small resources will be assigned fractionally and even more than once. In a second step, we transform the solution into one where each small resource is assigned only once—incurring a loss in the value that the players receive.

4.3.3 Rounding of Big Resources

The following lemma summarizes the properties we derive from the dependent rounding scheme. Note that while the assignment of small resources can change, it remains fractional for now.

Lemma 4.13. *Let (x, z) be a feasible solution to* $\mathrm{economical}(T, C)$*. There is a randomized algorithm that produces an assignment $X_{ib} \in \{0, 1\}$, $i \in \mathcal{P}$, $b \in \mathcal{B}$, and $z'_{is} \in [0, 1]$, $i \in \mathcal{P}$, $s \in \mathcal{S}$ such that*

1. $\sum_{i\in\mathcal{P}}\sum_{b\in\mathcal{B}} c_{ib}\cdot X_{ib}+\sum_{i\in\mathcal{P}}\sum_{s\in\mathcal{S}} c_{is}z'_{is}\leq C$,
2. *Each big resource is assigned at most one time, i.e.,* $\forall b \in \mathcal{B} : \sum_{i\in\mathcal{P}} X_{ib} \leq 1$,
3. *With high probability, each small resource is assigned at most $O(\log n)$ times, i.e.,* $\forall s \in \mathcal{S} : \sum_{i\in\mathcal{P}} z'_{is} \leq O(\log n)$,
4. *Each player receives either one big resource or a value of at least T in small resources.*

Proof. We first describe the instance to which we will apply the dependent rounding procedure from Theorem 4.2. We build a bipartite graph $G_{x,z} = (\mathcal{P} \cup (\mathcal{B} \cup \{d\}), E)$, where $\mathcal{P}$ is the set of players, $\mathcal{B}$ is the

set of big resources, and d is a *dummy node*. The role of d can be summarized as: every player who does not get a big resource selects the edge to d. Let graph $G_{x,z}$ contain an edge $(i, b) \in E$ labeled with cost c_{ib} between each big resource b and player i with $x_{ib} > 0$. Let J denote the set of all these edges. For every player i with $y_i > 0$, add an edge (i, d) to the dummy node of cost $\sum_{s\in\mathcal{S}} c_{is} \cdot (z_{is}/y_i) \in \mathbb{R}^n$. Let K denote the set of these edges and set $E = J \cup K$. Intuitively, x_{ib} is a fractional edge selection of edges in J and y_i a fractional edge selection of K (where y_i corresponds to edge $(i, d) \in K$). This fractional edge set satisfies the following.

(i) For each player i, we have $\sum_{b\in\mathcal{B}} x_{ib} + y_i = 1$ due to the definition of y_i.

(ii) For each big resource b, we have $\sum_{i\in\mathcal{P}} x_{ib} \leq 1$, due to Constraint (4.28).

Applying the dependent rounding procedure of Theorem 4.2 with $k = |\mathcal{S}|$ linear functions (we defer a precise definition to the end of the proof), we obtain an integral edge selection $X_{ib} \in \{0, 1\}$, $i \in \mathcal{P}$, $b \in \mathcal{B}$ and $Y_i \in \{0, 1\}$, $i \in \mathcal{P}$. Here, similar to before, Y_i defines the edge selection in K, i.e., edge (i, d) is selected if $Y_i = 1$. From (ii) and Theorem 4.2, it follows that $\sum_{i\in\mathcal{P}} X_{ib} \leq 1$ for all $b \in \mathcal{B}$, which means that Property 2 of the lemma holds. Further

$$\begin{aligned}\sum_{i\in\mathcal{P}}\sum_{b\in\mathcal{B}} c_{ib}\cdot X_{ib} + \sum_{i\in\mathcal{P}:y_i>0} Y_i \cdot \left(\sum_{s\in\mathcal{S}} c_{is}\cdot\frac{z_{is}}{y_i}\right) &\leq \sum_{i\in\mathcal{P}}\sum_{b\in\mathcal{B}} c_{ib}\cdot x_{ib} + \sum_{i\in\mathcal{P}:y_i>0}\left(\sum_{s\in\mathcal{S}} c_{is}\cdot\frac{z_{is}}{y_i}\right)\cdot y_i \\ &\leq \sum_{i\in\mathcal{P}}\sum_{b\in\mathcal{B}} c_{ib}\cdot x_{ib} + \sum_{i\in\mathcal{P}}\sum_{s\in\mathcal{S}} c_{is}\cdot z_{is} \\ &\leq C\end{aligned}$$

where the first inequality follows from the cost preservation property of Theorem 4.2 and the last inequality from the feasibility of (x, z). Notably, if $y_i = 0$, then $Y_i = 0$ as there does not exist an edge (i, d). In particular, $Y_i = 1$ implies that $y_i > 0$. We define the assignment of

small resources as

$$z'_{is} = \begin{cases} z_{is}/y_i & \text{if } Y_i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

Due to Constraint (4.27), we have $z_{is} \leq y_i$ and hence $z'_{is} \in [0,1]$. Consequently, Property 1 immediately follows from the previous cost calculation and the definition of z'_{is}. From (i) and Theorem 4.2, we obtain $\sum_{b \in \mathcal{B}} X_{ib} + Y_i = 1$ for each $i \in \mathcal{P}$. In words, player i either gets a big resource or $Y_i = 1$. In the latter case holds that

$$\sum_{s \in \mathcal{S}} v_s z'_{is} = \sum_{s \in \mathcal{S}} v_s z_{is}/y_i \geq T.$$

Here, the inequality follows from the definition of y_i and Constraint (4.26). Therefore, Property 4 holds.

It remains to show Property 3, i.e, that every small resource is assigned at most $O(\log n)$ times. To this end, we define the linear functions provided to Theorem 4.2: for each small resource $s \in \mathcal{S}$, there is one linear function $a^{(s)} \in [0,1]^K$, i.e., specified by the edges incident to d. Implicitly, the coefficient for all other edges is zero. Thus, the linear function satisfies the support restriction of Theorem 4.2. For $e = (i,d) \in K$, we define $a_e^{(s)} = z_{is}/y_i$. Using Constraint (4.27), it holds that

$$\sum_{e=(i,d) \in K} y_i \cdot a_e^{(s)} = \sum_{e=(i,d) \in K} y_i \cdot \frac{z_{is}}{y_i} = \sum_{e=(i,d) \in K} z_{is} \leq 1.$$

Finally, from the concentration bound of Theorem 4.2 follows

$$\sum_{i \in \mathcal{P}} z'_{is} = \sum_{e=(i,d) \in K} Y_i \cdot z'_{is} = \sum_{e=(i,d) \in K} Y_i \cdot z_{is}/y_i = \sum_{e=(i,d) \in K} Y_i \cdot a_e^{(s)} \leq O(\log n). \qquad \square$$

4.3.4 Rounding of Small Resources

In the previous subsection, we described how big resources $b \in \mathcal{B}$ were integrally assigned to the players. As some players did not receive any big resources and still need to be covered by small resources, let $\mathcal{Q}$ be the set of those players, i.e., players $i \in \mathcal{P}$ for which $Y_i = 1$. The linear program in the following lemma corresponds to the property of the assignment variables for small resources from the previous section.

Lemma 4.14. *Let $\mathcal{Q} \subseteq \mathcal{P}$ and consider the LP* $\mathrm{small}(T, \beta)$ *defined as*

$$\begin{aligned} \sum_{s \in \mathcal{S}} v_s \cdot z'_{is} &\geq T && \forall i \in \mathcal{P} \\ \sum_{i \in \mathcal{Q}} z'_{is} &\leq \beta && \forall s \in \mathcal{S} \\ z'_{is} &\geq 0 && \forall i \in \mathcal{Q}, s \in \mathcal{S} \end{aligned}$$

If $\mathrm{small}(T, \beta)$ *has a fractional solution z', then* $\mathrm{small}(T/\beta - \max_{s \in \mathcal{S}} v_s, 1)$ *has an integral solution Z that can be found in polynomial time with*

$$\sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{Q}} c_{is} \cdot Z_{is} \leq \sum_{s \in \mathcal{S}} \sum_{i \in \mathcal{Q}} c_{is} \cdot z'_{is},$$

Proof. Since $\mathrm{small}(T, \beta)$ is feasible, there exists a solution z'. Further, we obtain a (fractional) solution z'' to $\mathrm{small}(T/\beta, 1)$ by dividing all variables of z' by β. Using the approach by Shmoys and Tardos [101], we round z'' to an integral solution. For the sake of completeness, we provide the proof below.

Assume without loss of generality that $\mathcal{S} = \{1, 2, \ldots, |\mathcal{S}|\}$ is ordered such that $v_s \leq v_{s-1}$ for all $s = 2, 3, \ldots, |\mathcal{S}|$. We construct an auxiliary bipartite graph G (see fig. 4.1 for an illustration). The elements on the right side of G are elements of $\mathcal{S}$. On the left side, there are $k(i) := \lfloor \sum_{s \in \mathcal{S}} z''_{is} \rfloor$ many vertices $u_{i,1}, \ldots, u_{i,k(i)}$ for each $i \in \mathcal{Q}$. In the fractional solution z'', a player $i \in Q$ can get several small resources. Let $k(i)$

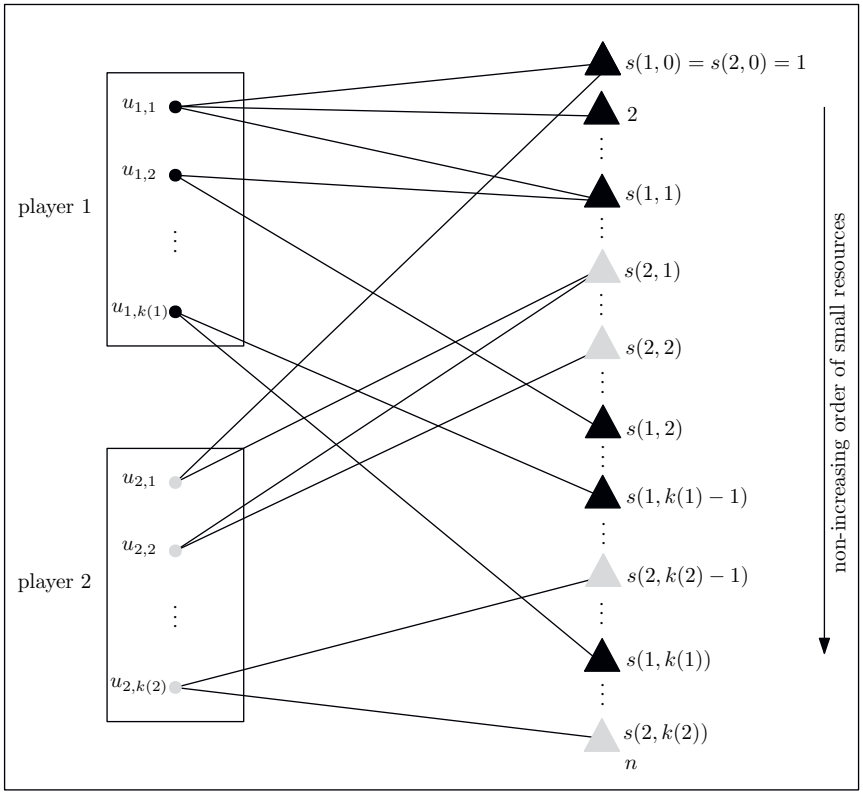


Figure 4.1: Bipartite graph G where on left side there are $k(i)$ copies of each player i for an instance of two players with edges $(u_{i,\ell}, s)$ of weight c_{is} for each $s = s(i,\ell-1), s(i,\ell-1)+1, \ldots, s(i,\ell)$.

denote their (rounded) number. Introducing $u_{i,1}, \ldots, u_{i,k(i)}$ as copies of player $i \in Q$ allows us to argue about matchings (where every vertex is only involved in one assignment). Suppose we add one edge between every two vertices of the different sides of the bipartite graph. It is straightforward that there exists a (fractional) left-perfect matching by distributing the resources assigned in z'' among the copies of each player. Due to the rounding in $k(i)$, this matching gives a slightly lower value to each player, but stays within the bounds we are aiming for.

To round to a good integral matching, however, we require a specific definition of the fractional left-perfect matching. Essentially, we need a monotone assignment where the first copy $u_{i,1}$ of player $i \in Q$ has the highest value resources (the first in the order above) and the last player the lowest value resources. Then G only contains the edges that are actually used in this assignment.

This requires some careful definitions. For each $i \in \mathcal{Q}$, we set $s(i,0) = 1$. This describes the first resource that can be fractionally assigned to player i. For $\ell = 1, 2, \ldots, k(i)$, we choose $s(i,\ell)$ as the element in $\mathcal{S}$ that satisfies

$$z''_{i,1} + \cdots + z''_{i,s(i,\ell)-1} < \ell \text{ and } z''_{i,1} + \cdots + z''_{i,s(i,\ell)} \geq \ell.$$

Note that $s(i,\ell)$ exists, because the sum of all $z''_{i,j}$ is at least $k(i) \geq \ell$. We only assign resources $s(i,\ell-1), \ldots, s(i,\ell)$ to copy $u_{i,\ell}$. Intuitively, the resources $1, 2, \ldots, s(i,\ell-1)-1$ should be exclusively used for the copies $u_{i,1}, \ldots, u_{i,\ell-1}$. Simply because the sum of fractions associated with i (according to z'') is not enough to cover the copies and $u_{i,\ell}$ should not receive any of them in order to maintain the monotonicity goals. Similarly, as the sum of fractions of resources $1, 2, \ldots, s(i,\ell)$ belonging to i exceeds ℓ, they are enough to cover all players in $u_{i,1}, \ldots, u_{i,\ell}$. Thus, it is not necessary to give any less valuable resources to $u_{i,\ell}$.

Consequently, for all $i \in \mathcal{Q}$ and $\ell = 1, 2, \ldots, k(i)$, we introduce an edge $(u_{i,\ell}, s)$ of weight c_{is} for each $s = s(i,\ell-1), s(i,\ell-1)+1, \ldots, s(i,\ell)$. We now formally show that there is a left-perfect fractional matching of weight at most $\sum_{s\in\mathcal{S}} \sum_{i\in\mathcal{Q}} c_{is} \cdot z''_{is}$. Towards this, consider some $i \in \mathcal{Q}$ and $\ell \in \{1, 2, \ldots, k(i)\}$. We select edge $(u_{i,\ell}, s(i,\ell-1))$ to an extend of

$$z''_{i,1} + \cdots + z''_{i,s(i,\ell-1)} - (\ell-1) \in [0, z''_{i,s(i,\ell-1)}].$$

For $s = s(i,\ell-1)+1, \ldots, s(i,\ell)-1$, we pick edge $(u_{i,\ell}, s)$ to an extend of $z''_{i,s}$. Finally, we choose edge $(u_{i,\ell}, s(i,\ell))$ to an extend of

$$\ell - (z''_{i,1} + \cdots + z''_{i,s(i,\ell)-1}) \in [0, z''_{i,s(i,\ell)}].$$

This fractional selection of edges satisfies the following properties.

1. For each $i \in \mathcal{Q}$ and $\ell \in \{1, 2, \ldots, k(i)\}$, the total fractional amount of selected edges that are incident to $u_{i,\ell}$ is exactly 1.

2. For each $i \in \mathcal{Q}$ and $s \in \mathcal{S}$, the total fractional amount of selected edges that are between $u_{i,1}, \ldots, u_{i,k(i)}$ and s is at most $z''_{i,s}$.

Property 2 implies that the total fractional amount that edges incident to s (over all $i \in \mathcal{Q}$) are selected is at most 1. Hence, the total weight is at most the cost of z''. For a bipartite graph, the set of all fractional matchings is precisely the convex hull of integral matchings, see e.g. [100, Chapter 18]. Furthermore, z'' must lie on a facet spanned by only left-perfect matchings. Thus, there must also exist an integral left-perfect matching M of weight at most

$$\sum_{i \in \mathcal{Q}} \sum_{s \in \mathcal{S}} z''_{i,s} c_{i,s} \leq \sum_{i \in \mathcal{Q}} \sum_{s \in \mathcal{S}} z'_{i,s} c_{i,s}.$$

We can find such a matching using standard algorithms for minimum weight bipartite matching. We interpret M as an integral assignment Z where each $i \in \mathcal{Q}$ receives all $s \in \mathcal{S}$, for which there is an edge $(u_{i,\ell}, s)$ in M for some $\ell \in \{1, 2, \ldots, k(i)\}$. Finally, we analyze how much value each $i \in \mathcal{Q}$ receives in this assignment. Notice that all resources $s \in \mathcal{S}$ that are connected to $u_{i,\ell}$ have a value of at least $v_{s(i,\ell)}$. Therefore, i receives a total value of at least

$$v_{s(i,1)} + \cdots + v_{s(i,k(i))}.$$

On the other hand, since $z''_{i,1} + \cdots + z''_{i,s(i,\ell)-1} + z''_{i,s(i,\ell)} < \ell + z''_{i,s(i,\ell)} \leq \ell + 1$ for each ℓ, the total fractional amount of resources that i receives in z'' with value at least $v_{s(i,\ell)}$ is less than $\ell + 1$. As a result, the total value that i received in z'' is at most

$$v_{s(i,0)} + \cdots + v_{s(i,k(i))}.$$

This is at least T/β, because of $z'' \in \text{small}(T/\beta, 1)$. As a consequence, in Z player i receives a value of at least $T/\beta - v_{s(i,0)} \geq T/\beta - \max_{s \in \mathcal{S}} v_s$. □

4.3.5 Approximation Factor

Concluding the previous subsections, this section provides an α-approximation for the Budgeted Santa Claus Problem, where $\alpha = O(\log n)$.

Theorem 4.3. *There is a randomized polynomial time $O(\log n)$-approximation algorithm for the Budgeted Santa Claus problem.*

Proof. Let $\beta = O(\log n)$ be the value from Property 3 of Lemma 4.13. We define $\alpha = 2\beta$ and run our binary search over value T, which is then used in our definition of big and small resources. Then we solve the LP relaxation $\text{economical}(T)$. Let (x, z) be the resulting solution. If the cost of the solution is more than C, we fail (and increase the value of T). Assume now that (x, z) has a cost of at most C. In Lemma 4.13, using the dependent rounding procedure from Theorem 4.2, we show that the fractional assignments x of big resources to players can be rounded to an integral assignment X and the assignment of small resources z can be changed to z' (which is still fractional), such that with high probability each small resource up to β times and the cost is still below C. Lemma 4.14 proves that z' can be rounded to an integral assignment Z such that the cost does not increase, and each player i that does not receive a big resource gets small resources of total value at least

$$\frac{T}{\beta} - \max_{s \in \mathcal{S}} v_s \geq \frac{2T}{\alpha} - \frac{T}{\alpha} = \frac{T}{\alpha}.$$ □

4.4 Concluding Remarks

Based on the findings in this paper, there are several interesting questions arising for future research. For the Budgeted Santa Claus Problem, a naturally arising question is whether the approximation

factor of $O(\log n)$ can be improved. Notably, the special case of restricted assignment (without a budget constraint) admits a constant approximation due to Feige [47]. We are not aware of any hardness results indicating that such a result cannot hold for our problem. As an intermediate question, one could look at the bi-criteria approximation that approximates both the minimum player value and the cost by a constant. This would still generalize the aforementioned algorithm for restricted assignment.

Another intriguing question is whether a dependent rounding scheme exists for rounding matroid bases that simultaneously guarantees cost preservation and Chernoff-type concentration like SwapRounding [26] does. It seems likely that the techniques from this paper transfer at least to a limited class of matroids, namely, strongly base-orderable matroids, because these have very strong decomposition properties for the symmetric difference of two bases. It might, however, require other ideas to generalize to arbitrary matroids.

Bibliography

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[2] Chidambaram Annamalai, Christos Kalaitzis, and Ola Svensson. "Combinatorial Algorithm for Restricted Max-Min Fair Allocation". In: *ACM Trans. Algorithms* 13.3 (2017), 37:1–37:28. DOI: 10.1145/3070694.

[3] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[4] Arash Asadpour, Uriel Feige, and Amin Saberi. "Santa Claus Meets Hypergraph Matchings". In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop (APPROX 2008) and 12th International Workshop (RANDOM 2008)*. Vol. 5171. Lecture Notes in Computer Science. Springer, 2008, pp. 10–20. DOI: 10.1007/978-3-540-85363-3_2.

[5] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. "An $O(\log n/\log \log n)$-Approximation Algorithm for the Asymmetric Traveling Salesman Problem". In: *Oper. Res.* 65.4 (2017), pp. 1043–1061. DOI: 10.1287/OPRE.2017.1603.

[6] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.

[7] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. "Tradeoffs in worst-case equilibria". In: *Approximation and Online Algorithms: First International Workshop, WAOA 2003, Budapest, Hungary, September 16-18, 2003. Revised Papers 1*. Springer. 2004, pp. 41–52.

[8] Yossi Azar, Lisa Fleischer, Kamal Jain, Vahab Mirrokni, and Zoya Svitkina. "Optimal coordination mechanisms for unrelated machine scheduling". In: *Operations Research* 63.3 (2015), pp. 489–500.

[9] Étienne Bamas, Alexander Lindermayr, Nicole Megow, Lars Rohwedder, and Jens Schlöter. "Santa Claus meets Makespan and Matroids: Algorithms and Reductions". In: *35th Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2024)*. SIAM, 2024, pp. 2829–2860. DOI: 10.1137/1.9781611977912.100.

[10] Nikhil Bansal and Viswanath Nagarajan. "Approximation-Friendly Discrepancy Rounding". In: *18th Integer Programming and Combinatorial Optimization Conference (IPCO 2016)*. Vol. 9682. Lecture Notes in Computer Science. Springer, 2016, pp. 375–386. DOI: 10.1007/978-3-319-33461-5_31.

[11] Nikhil Bansal and Maxim Sviridenko. "The Santa Claus problem". In: *38th Annual ACM Symposium on Theory of Computing, (STOC 2006)*. ACM, 2006, pp. 31–40. DOI: 10.1145/1132516.1132522.

[12] Massoud Bazargan. "A linear programming approach for aircraft boarding strategy". In: *European Journal of Operational Research* 183.1 (2007), pp. 394–411.

[13] Andre Berger, Arman Rouhani, and Marc Schröder. "An improved bound for the price of anarchy for related machine scheduling". In: *Discrete Optimization* 58 (2025). DOI: 10.1016/j.disopt.2025.100911.

[14] Andre Berger, Arman Rouhani, and Marc Schröder. “Fixed order scheduling with deadlines”. In: *Operations Research Letters* 61 (2025). DOI: 10.1016/j.orl.2025.107306.

[15] Sayan Bhattacharya, Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. “Coordination mechanisms from (almost) all scheduling policies”. In: *Proceedings of the 5th Conference on Innovations in Theoretical Computer Science (ITCS ’14)*. New York, NY, USA: Association for Computing Machinery, 2014, pp. 121–134. DOI: 10.1145/2554797.2554811.

[16] Kshipra Bhawalkar and Tim Roughgarden. “Welfare guarantees for combinatorial auctions with item bidding”. In: *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*. SIAM. 2011, pp. 700–709.

[17] Vittorio Bilò. “A unifying tool for bounding the quality of non-cooperative solutions in weighted congestion games”. In: *Theory of Computing Systems* 62 (2018), pp. 1288–1317.

[18] Paul Bonsma, Jens Schulz, and Andreas Wiese. “A Constant-Factor Approximation Algorithm for Unsplittable Flow on Paths”. In: *SIAM Journal on Computing* 43.2 (2014), pp. 767–799. DOI: 10.1137/120868360.

[19] Thomas Bosman, Dario Frascaria, Neil Olver, René Sitters, and Leen Stougie. “Fixed-order scheduling on parallel machines”. In: *Integer Programming and Combinatorial Optimization: 20th International Conference, IPCO 2019, Ann Arbor, MI, USA, May 22-24, 2019, Proceedings 20*. Springer. 2019, pp. 88–100.

[20] Ruben Brokkelkamp. “How Close Does It Get? From Near-Optimal Network Algorithms to Suboptimal Equilibrium Outcomes”. PhD thesis. Amsterdam, Netherlands: Universiteit van Amsterdam, 2022.

[21] Peter Brucker. *Scheduling Algorithms*. 5th edition. Berlin, Heidelberg: Springer, 2007. DOI: 10.1007/978-3-540-69516-5.

[22] Jan K Brueckner. "Airport congestion when carriers have market power". In: *American Economic Review* 92.5 (2002), pp. 1357–1375.

[23] Giorgio C Buttazzo and Giorgio Buttanzo. *Hard real-time computing systems*. Vol. 356. Springer, 1997.

[24] Cambridge University Press. *Scheduling*. In: *Cambridge Advanced Learner's Dictionary & Thesaurus*. 4th ed. Cambridge, UK: Cambridge University Press. 2013.

[25] Brecht Cardoen, Erik Demeulemeester, and Jeroen Beliën. "Operating room planning and scheduling: A literature review". In: *European journal of operational research* 201.3 (2010), pp. 921–932.

[26] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. "Dependent Randomized Rounding via Exchange Properties of Combinatorial Structures". In: *51st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2010)*. IEEE Computer Society, 2010, pp. 575–584. DOI: 10.1109/FOCS.2010.60.

[27] Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. "Multi-budgeted Matchings and Matroid Intersection via Dependent Rounding". In: *22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*. SIAM, 2011, pp. 1080–1097. DOI: 10.1137/1.9781611973082.82.

[28] Mei Ching Chen, Say Leng Goh, Nasser R Sabar, Graham Kendall, et al. "A survey of university course timetabling problem: perspectives, trends and opportunities". In: *IEEE Access* 9 (2021), pp. 106515–106529.

[29] George Christodoulou, Elias Koutsoupias, and Akash Nanavati. "Coordination mechanisms". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2004, pp. 345–357.

[30] George Christodoulou, Elias Koutsoupias, and Akash Nanavati. "Coordination mechanisms". In: *Theoretical Computer Science* 410.36 (2009), pp. 3327–3336.

[31] Mark Cieliebak, Thomas Erlebach, Fabian Hennecke, Birgitta Weber, and Peter Widmayer. "Scheduling with release times and deadlines on a minimum number of machines". In: *Exploring New Frontiers of Theoretical Informatics: IFIP 18th World Computer Congress TC1 3rd International Conference on Theoretical Computer Science (TCS2004) 22–27 August 2004 Toulouse, France*. Springer. 2004, pp. 209–222.

[32] Edward G. Coffman, Michael R. Garey, and David S. Johnson. "Approximation Algorithms for Bin Packing: A Survey". In: *Approximation Algorithms for NP-Hard Problems*. Ed. by Dorit S. Hochbaum. PWS Publishing, 1997, pp. 46–93.

[33] Edward G. Jr. Coffman, ed. *Computer and Job-Shop Scheduling Theory*. New York: John Wiley & Sons, 1976.

[34] EG Coffman, MR Garey, and DS Johnson. "Approximation Algorithms for Bin Packing: A Survey. Chapter 2 in Approximation Algorithms for NP-Hard Problems". PhD thesis. ed. D. Hochbaum, 46–93 Boston: PWS Publishing Company, 1997.

[35] Richard Cole, José R Correa, Vasilis Gkatzelis, Vahab Mirrokni, and Neil Olver. "Decentralized utilitarian mechanisms for scheduling games". In: *Games and Economic Behavior* 92 (2015), pp. 306–326.

[36] Richard W. Conway, William L. Maxwell, and Louis W. Miller. *Theory of Scheduling*. Reading, MA: Addison-Wesley, 1967.

[37] Richard Walter Conway, William L Maxwell, and Louis W Miller. *Theory of scheduling*. Courier Corporation, 2003.

[38] José R Correa and Maurice Queyranne. "Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost". In: *Naval Research Logistics (NRL)* 59.5 (2012), pp. 384–395.

[39] Artur Czumaj and Berthold Vöcking. “Tight bounds for worst-case equilibria”. In: *ACM Transactions on Algorithms (TALG)* 3.1 (2007), pp. 1–17.

[40] GB Dantzig and DR Fulkerson. “Minimizing the number of tankers to meet a fixed schedule”. In: *The Basic George B. Dantzig* (2003), pp. 217–222.

[41] George Dantzig. *Linear Programming and Extensions*. RAND Corporation, 1963. DOI: 10.7249/r366.

[42] Sami Davies, Thomas Rothvoss, and Yihao Zhang. “A Tale of Santa Claus, Hypergraphs and Matroids”. In: *31st Annual ACM-SIAM Symposium on Discrete Algorithms, (SODA 2020)*. SIAM, 2020, pp. 2748–2757. DOI: 10.1137/1.9781611975994.167.

[43] Maxence Delorme, Manuel Iori, and Silvano Martello. “Bin packing and cutting stock problems: Mathematical models and exact algorithms”. In: *European Journal of Operational Research* 255.1 (2016), pp. 1–20.

[44] Ulrich Dorndorf, Andreas Drexl, Yury Nikulin, and Erwin Pesch. “Flight gate scheduling: State-of-the-art and recent developments”. In: *Omega* 35.3 (2007), pp. 326–334.

[45] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. DOI: 10.1017/cbo9780511581274.

[46] Stephan Eidenbenz, Aris Pagourtzis, and Peter Widmayer. “Flexible train rostering”. In: *International Symposium on Algorithms and Computation*. Springer. 2003, pp. 615–624.

[47] Uriel Feige. “On allocations that maximize fairness”. In: *19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008)*. SIAM, 2008, pp. 287–293.

[48] Amos Fiat, Haim Kaplan, Meital Levy, and Svetlana Olonetsky. "Strong price of anarchy for machine load balancing". In: *International Colloquium on Automata, Languages, and Programming*. Springer. 2007, pp. 583–594.

[49] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. "Computing Nash equilibria for scheduling on restricted parallel links". In: *Theory of Computing Systems* 47 (2010), pp. 405–432.

[50] Rajiv Gandhi, Samir Khuller, Srinivasan Parthasarathy, and Aravind Srinivasan. "Dependent rounding and its applications to approximation algorithms". In: *J. ACM* 53.3 (2006), pp. 324–360. DOI: 10.1145/1147954.1147956.

[51] Michael R Garey and David S Johnson. *Computers and intractability*. Vol. 29. wh freeman New York, 2002.

[52] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman, 1979.

[53] Fănică Gavril. "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph". In: *SIAM Journal on Computing* 1.2 (1972), pp. 180–187.

[54] Balaji Gopalakrishnan and Ellis L Johnson. "Airline crew scheduling: State-of-the-art". In: *Annals of Operations Research* 140 (2005), pp. 305–337.

[55] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. "Optimization and approximation in deterministic sequencing and scheduling: a survey". In: *Annals of discrete mathematics*. Vol. 5. Elsevier, 1979, pp. 287–326.

[56] Fabrizio Grandoni and Rico Zenklusen. "Approximation Schemes for Multi-Budgeted Independence Systems". In: *18th Annual European Symposium (ESA 2010)*. Vol. 6346. Lecture Notes in Computer Science. Springer, 2010, pp. 536–548. DOI: 10.1007/978-3-642-15775-2_46.

[57] Diwakar Gupta and Brian Denton. "Appointment scheduling in health care: Challenges and opportunities". In: *IIE transactions* 40.9 (2008), pp. 800–819.

[58] René Haijema, Nico van Dijk, Jan van der Wal, and Cees Smit Sibinga. "Blood platelet production with breaks: optimization by SDP and simulation". In: *International Journal of Production Economics* 121.2 (2009), pp. 464–473.

[59] David G Harris. "Dependent rounding with strong negative-correlation, and scheduling on unrelated machines to minimize completion time". In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. SIAM. 2024, pp. 2275–2304.

[60] Julia Heil, Kirsten Hoffmann, and Udo Büscher. "Railway Crew Scheduling: Models, Methods and Applications". In: *European Journal of Operational Research* 283.2 (2020), pp. 405–425.

[61] Birgit Heydenreich, Rudolf Müller, and Marc Uetz. "Games and mechanism design in machine scheduling—an introduction". In: *Production and operations management* 16.4 (2007), pp. 437–454.

[62] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. 8th ed. New York: McGraw-Hill, 2005.

[63] Dorit S Hochba. "Approximation algorithms for NP-hard problems". In: *ACM Sigact News* 28.2 (1997), pp. 40–52.

[64] Ruben Hoeksma and Marc Uetz. "The price of anarchy for utilitarian scheduling games on related machines". In: *Discrete optimization* 31 (2019), pp. 29–39.

[65] Ellis Horowitz and Sartaj Sahni. "Exact and approximate algorithms for scheduling nonidentical processors". In: *Journal of the ACM (JACM)* 23.2 (1976), pp. 317–327.

[66] Oscar H Ibarra and Chul E Kim. "Heuristic algorithms for scheduling independent tasks on nonidentical processors". In: *Journal of the ACM (JACM)* 24.2 (1977), pp. 280–289.

[67] Nicole Immorlica, Li Erran Li, Vahab S Mirrokni, and Andreas S Schulz. "Coordination mechanisms for selfish scheduling". In: *Theoretical computer science* 410.17 (2009), pp. 1589–1598.

[68] Klaus Jansen, Kim-Manuel Klein, and José Verschae. "Closing the Gap for Makespan Scheduling via Sparsification Techniques". In: *Math. Oper. Res.* 45.4 (2020), pp. 1371–1392. DOI: 10.1287/MOOR.2019.1036.

[69] David S. Johnson. *Near-Optimal Bin Packing Algorithms*. Tech. rep. MIT-TR-99. MIT Center for Computer Research in Economics and Management, 1973.

[70] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984, pp. 302–311.

[71] Tsuyoshi Kawaguchi and Seiki Kyan. "Worst case bound of an LRF schedule for the mean weighted flow-time problem". In: *SIAM Journal on Computing* 15.4 (1986), pp. 1119–1129.

[72] Leonid G Khachiyan. "Polynomial algorithms in linear programming". In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72.

[73] Elias Koutsoupias. "Scheduling without payments". In: *Theory of Computing Systems* 54.3 (2014), pp. 375–387.

[74] Elias Koutsoupias and Christos Papadimitriou. "Worst-case equilibria". In: *Annual symposium on theoretical aspects of computer science*. Springer. 1999, pp. 404–413.

[75] Svetlana A Kravchenko and Frank Werner. "Minimizing the number of machines for scheduling jobs with equal processing times". In: *European Journal of Operational Research* 199.2 (2009), pp. 595–600.

[76] Leo G Kroon, Marc Salomon, and Luk N Van Wassenhove. "Exact and approximation algorithms for the tactical fixed interval scheduling problem". In: *Operations Research* 45.4 (1997), pp. 624–638.

[77] Kangbok Lee, Joseph Y-T Leung, and Michael L Pinedo. "Coordination mechanisms for parallel machine scheduling". In: *European Journal of Operational Research* 220.2 (2012), pp. 305–313.

[78] Shachar Lovett and Raghu Meka. "Constructive Discrepancy Minimization by Walking on the Edges". In: *SIAM J. Comput.* 44.5 (2015), pp. 1573–1582. DOI: 10.1137/130929400.

[79] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[80] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge, UK: Cambridge University Press, 2005.

[81] Aziz Moukrim. "On the minimum number of processors for scheduling problems with communicationdelays". In: *Annals of operations Research* 86.0 (1999), pp. 455–472.

[82] Felipe T Muñoz and Rodrigo Linfati. "Bounding the Price of Anarchy of Weighted Shortest Processing Time Policy on Uniform Parallel Machines". In: *Mathematics* 12.14 (2024), p. 2223.

[83] Steven Nahmias. "Perishable inventory theory: A review". In: *Operations research* 30.4 (1982), pp. 680–708.

[84] Noam Nisan et al. "Introduction to mechanism design (for computer scientists)". In: *Algorithmic game theory* 9 (2007), pp. 209–242.

[85] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, eds. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[86] Luis Osorio-Valenzuela, Jordi Pereira, Franco Quezada, and Óscar C Vásquez. "Minimizing the number of machines with limited workload capacity for scheduling jobs with interval constraints". In: *Applied Mathematical Modelling* 74 (2019), pp. 512–527.

[87] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Mineola, NY: Dover Publications, 1998.

[88] Jordi Pereira. "Procedures for the bin packing problem with precedence constraints". In: *European Journal of Operational Research* 250.3 (2016), pp. 794–806.

[89] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. 4th. Springer, 2012.

[90] Prabhakar Raghavan and Clark D Tompson. "Randomized rounding: a technique for provably good algorithms and algorithmic proofs". In: *Combinatorica* 7.4 (1987), pp. 365–374.

[91] Mona Rahn and Guido Schäfer. "Bounding the inefficiency of altruism through social contribution games". In: *Web and Internet Economics: 9th International Conference, WINE 2013, Cambridge, MA, USA, December 11-14, 2013, Proceedings 9*. Springer. 2013, pp. 391–404.

[92] Lars Rohwedder, Arman Rouhani, and Leo Wennmann. "Cost Preserving Dependent Rounding for Allocation Problems". In: *52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik. 2025, pp. 127–1.

[93] Tim Roughgarden. "Algorithmic game theory". In: *Communications of the ACM* 53.7 (2010), pp. 78–86.

[94] Tim Roughgarden. "Intrinsic robustness of the price of anarchy". In: *Journal of the ACM (JACM)* 62.5 (2015), pp. 1–42.

[95] Tim Roughgarden. *Selfish Routing and the Price of Anarchy*. Cambridge, MA: MIT Press, 2005.

[96] Tim Roughgarden, Vasilis Syrgkanis, and Eva Tardos. "The price of anarchy in auctions". In: *Journal of Artificial Intelligence Research* 59 (2017), pp. 59–101.

[97] Tim Roughgarden and Éva Tardos. "How bad is selfish routing?" In: *Journal of the ACM (JACM)* 49.2 (2002), pp. 236–259.

[98] Ruslan Sadykov and François Vanderbeck. "Bin packing with conflicts: a generic branch-and-price algorithm". In: *INFORMS Journal on Computing* 25.2 (2013), pp. 244–255.

[99] Barna Saha and Aravind Srinivasan. "A new approximation technique for resource-allocation problems". In: *Random Struct. Algorithms* 52.4 (2018), pp. 680–715. DOI: 10.1002/RSA.20756.

[100] Alexander Schrijver et al. *Combinatorial optimization: polyhedra and efficiency*. Vol. 24. Springer, 2003.

[101] David B. Shmoys and Éva Tardos. "An approximation algorithm for the generalized assignment problem". In: *Math. Program.* 62 (1993), pp. 461–474. DOI: 10.1007/BF01585178.

[102] Donald Shoup. *High cost of free parking*. Routledge, 2021.

[103] Vasilis Syrgkanis and Eva Tardos. "Composable and efficient mechanisms". In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 2013, pp. 211–220.

[104] Hamdy A Taha. *Operations research: an introduction*. Pearson Education India, 2013.

[105] Éva Tardos. "A strongly polynomial algorithm to solve combinatorial linear programs". In: *Operations Research* 34.2 (1986), pp. 250–256.

[106] Vijay V Vazirani. *Approximation Algorithms*. 2001.

[107] Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir. “Minimizing total completion time with machine-dependent priority lists”. In: *European Journal of Operational Research* 315.3 (2024), pp. 844–854.

[108] Vipin Ravindran Vijayalakshmi, Marc Schröder, and Tami Tamir. “Scheduling games with machine-dependent priority lists”. In: *Theoretical Computer Science* 855 (2021), pp. 90–103.

[109] Berthold Vöcking. “Selfish load balancing”. In: *Algorithmic game theory* 20 (2007), pp. 517–542.

[110] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[111] Guosong Yu and Guochuan Zhang. “Scheduling with a minimum number of machines”. In: *Operations Research Letters* 37.2 (2009), pp. 97–101.

[112] Lingfei Yu, Kun She, Haigang Gong, and Changyuan Yu. “Price of anarchy in parallel processing”. In: *Information Processing Letters* 110.8-9 (2010), pp. 288–293.

[113] Long Zhang, Zhiwen Wang, and Wei Sheng. “Nash equilibria and the price of anarchy under SPT-LPT mixed coordination mechanism in distributed manufacturing”. In: *Computers & Operations Research* (2025), p. 107117.

[114] Long Zhang, Yuzhong Zhang, Donglei Du, and Qingguo Bai. “Improved price of anarchy for machine scheduling games with coordination mechanisms”. In: *Optimization Letters* 13 (2019), pp. 949–959.

Summary

In this thesis, we consider several scheduling problems inspired by practical applications. A scheduling problem concerns assigning tasks to a limited number of machines in order to optimize a given objective. The specific objective depends on the application. We use mathematical techniques to formulate these problems, analyze their structure, and identify the main challenges they present. We then design efficient algorithms that find optimal solutions whenever possible. However, many realistic variants are computationally hard, making optimal solutions difficult to obtain within a reasonable amount of time. This motivates the study of approximation algorithms. These algorithms trade optimality for efficiency, i.e., they do not guarantee an optimal solution but aim to efficiently produce solutions that are "good" enough with a provable guarantee on how far their value can be from the optimum. In this thesis, our focus is on theoretical approximation algorithms for scheduling problems.

Chapter 2 presents an improved upper bound on the price of anarchy for scheduling games on related machines. The machines have different speeds and use the shortest processing time (SPT) rule for scheduling the jobs. Each job is controlled by a selfish player who aims to minimize its own completion time, while the social objective is to minimize the sum of completion times of all jobs. Our main result gives an upper bound of $2 - 1/(4m - 2)$ on the price of anarchy for the general case with m machines. We improve this bound to $3/2$ for the case of two machines and to $2 - 1/(2m)$ for the general case with m machines when the machines have divisible speeds, i.e., when the speed of every machine is divisible by the speed of every slower machine.

Chapter 3 studies a scheduling problem in a machine environment in which each machine must respect a predetermined order for processing the jobs. Given n jobs, each with a processing time and a deadline, we aim to minimize the number of machines used while respecting the

deadlines and preserving the order on each machine. We show that the first-fit algorithm solves the problem optimally in the case of identical processing times and that it is a 2-approximation in the following cases: (1) the order is consistent with non-increasing slacks, (2) the order is consistent with non-decreasing slacks, (3) the order is consistent with non-increasing deadlines, and (4) the optimal solution uses at most three machines. Finally, we provide an algorithm with an $O(\log n)$ approximation factor for the general case.

In Chapter 4, we present a dependent randomized rounding method that rounds fractional solutions to integral solutions that satisfy certain hard constraints on the output while preserving Chernoff-like concentration properties. In contrast to previous dependent rounding schemes, our algorithm guarantees that the cost of the rounded integral solution is not higher than that of the fractional solution. Our algorithm works for a class of assignment problems with restrictions similar to those of prior works. In a non-trivial combination of our general result with a classical approach of Shmoys and Tardos [101] and more recent linear programming techniques developed for the restricted assignment variant by Bansal and Sviridenko [11] and by Davies, Rothvoss, and Zhang [42], we derive an algorithm with an $O(\log n)$ approximation factor for the Budgeted Santa Claus Problem. In this new variant, the objective is to assign resources with different values to players, maximize the minimum value received by a player, and satisfy a budget constraint on the assignment costs between players and resources.

Valorisation

In this dissertation, we address different scheduling problems inspired by real-world applications. A scheduling problem concerns the assignment of tasks to limited resources with the aim of optimizing one or more objectives. Depending on the application, the objective may be to minimize the number of resources used or to optimize a utilitarian or egalitarian social choice function. We apply mathematical techniques to model these problems, uncover their intrinsic structure, and identify the challenges they pose. We then design efficient algorithms that solve them optimally when possible. Because many realistic variants are computationally hard, optimal solutions may be unattainable within a reasonable time. For computationally hard problems, considerable research has focused on the design and analysis of approximation algorithms. These algorithms trade optimality for efficiency, i.e., they do not guarantee an optimal solution but aim to produce solutions that are "good" enough efficiently. Some approximation algorithms are developed from a theoretical perspective, offering worst-case guarantees on both solution quality and running time. Others are driven by practical performance: they tend to work well on real-world instances, but offer little to no formal guarantees regarding their accuracy or computational complexity. In this thesis, our focus is on theoretical approximation algorithms for scheduling problems. Although practical approximation algorithms often demonstrate superior performance in typical instances, the investigation of theoretical approximation algorithms remains essential for several reasons.

- **Worst-case guarantees and robustness.** Practical approaches lack formal performance guarantees, which makes their behavior unpredictable on worst-case instances. They may yield very good solutions for typical data but perform poorly on adversarially constructed inputs. In contrast, theoretical approximation algorithms provide worst-case guarantees, ensuring that the

solution quality is always within a bounded factor of the optimum. Such guarantees are important in applications where reliability is essential [106, 110].

- **Complexity classification.** The existence (or non-existence) of approximation algorithms with specific performance ratios helps classify optimization problems within the hierarchy of complexity classes [3, 6].
- **Problem hardness characterization.** Inapproximability results establish fundamental limits on achievable approximation factors, preventing futile research directions and focusing efforts on tractable algorithmic improvements [106]
- **Algorithmic design principles.** The pursuit of improved approximation factors drives the development of novel algorithmic techniques such as LP rounding, primal-dual method, and semidefinite programming that often inspire practical heuristics [110].
- **Performance benchmarking.** Theoretical approximation ratios serve as objective benchmarks for evaluating practical algorithms, providing a quantitative baseline against which heuristic performance can be measured [63].

Thus, theoretical analysis does not compete with practical algorithm design but rather provides the foundation for understanding, evaluating, and systematically improving heuristic and empirical approaches to hard optimization problems.

In Chapter 2, we study utilitarian scheduling games on related machines, where tasks are controlled by strategic entities who are selfish and only wish to optimize their own objective rather than the system's performance. This setting simulates a realistic application where there is a lack of central decision-making. We aim to measure the "anarchy" caused by selfish behavior, called the price of anarchy. For this specific problem, this is equivalent to approximating a greedy algorithm designed for the optimization version of the problem, where

there is a central decision-maker. Our results suggest the improved upper bound of $2 - 1/(4m - 2)$ over the long-standing bound of 2 for the price of anarchy, which only depends on the number of machines. Obtaining an upper bound means that even if every player acts selfishly, the total performance of the system cannot be worse than this factor compared to an ideal centralized solution. Operators can use this as a safety margin when they allow user choice: they can plan capacity, set performance targets, and evaluate observed behavior knowing that the loss from decentralization is provably limited. In practice, many systems can match our setting: there is no central dispatcher, users choose a resource for their own job, and resources mainly differ by speed. Examples include multi-tenant cloud and edge platforms, university clusters, compute marketplaces (such as render farms), on-demand manufacturing and 3D printing platforms, citizen service centers (for instance, clinics or passport offices), and shared laboratory instruments. In these systems, jobs are similar, and machines work at different speeds. The guarantee is relevant when tasks are broadly similar and processing times scale with machine speed. If machines are unrelated, jobs have extra constraints, or preemption and dependencies are present, a different model is needed. Another interesting aspect of this result is the primal-dual technique we use, and we believe that these techniques can be used to help close the gap of $[e/(e-1), 2]$.

In Chapter 3 we study another interesting scheduling problem. In many scheduling problems inspired by real-world applications, scheduling based on a fixed order on a minimum number of machines is required. Fixed order scheduling addresses critical industrial and logistical constraints arising from safety protocols, physical dependencies, or regulatory requirements. Moreover, minimizing the number of machines required is essential for reducing capital expenditure, operational costs, and resource footprint in capital-intensive industries. Fixed-order scheduling arises in many operations where jobs must follow a global sequence while several identical machines run in parallel. Typical areas include airport gate planning, block-scheduled operating rooms, and blood-platelet processing. In these systems, the

order is fixed in advance and cannot be changed on any machine. The aim is to split the given sequence across as few machines as possible so that every job respects its deadline. Since the order cannot be permuted, standard scheduling methods that rely on reordering jobs do not apply. In this chapter, we investigate the impact of the imposed processing order on job scheduling and study the approximation factor of some of the simplest greedy algorithms for the problem: first-fit and next-fit. We prove that first-fit optimally solves the problem when the processing times are equal and achieves the upper bound of 2 for several variants of the problem. On the other hand, we prove that next-fit has an unbounded approximation factor for the problem, and even for the case of equal processing times, has a lower bound of 2. We then provide a logarithmic approximation factor for the general case using another greedy algorithm that fits the largest number of unscheduled jobs on a machine using a dynamic programming approach. This chapter demonstrates how simple algorithms with reasonable running times can provide near-optimal solutions for these realistic scheduling problems.

In Chapter 4, we study max–min allocation under a hard budget. The objective is to maximize the minimum value any player receives while the total assignment cost stays within a fixed budget cap. Items are indivisible and have identical value across players, but each assignment has a cost. This setting arises when a planner must allocate resources under a strict budget and fairness is measured by the minimum recipient value. Typical examples include purchasing devices for sites under a capital budget (e.g., medical monitors or laptops), distributing fixed-size cloud/GPU credit packs across research groups subject to a spend cap, sending standardized emergency-relief kits to regions with logistics costs, awarding fixed-amount grants or vouchers within a funding envelope, or deploying identical environmental sensors across districts with site-specific installation costs. In all these cases, an integral allocation is required, each unit can go to at most one recipient, and overspending is unacceptable. So, standard independent rounding is unsafe. Our contribution is a dependent randomized

rounding method for many-to-many assignments on a bipartite graph that converts a feasible fractional LP solution into an integral solution without violating the cost constraint. We apply this tool to the budgeted Santa Claus problem with identical valuations and obtain a randomized $O(\log n)$ approximation for maximizing the minimum player value subject to the budget. In practice, operators can model the problem as an ILP, solve the LP relaxation, and apply our dependent rounding scheme to obtain an integral solution without risking a budget overrun, while keeping each player's value within a logarithmic factor of the optimal slotion.

Nederlandse Samenvatting

In dit proefschrift behandelen wij verschillende planningsproblemen die zijn geïnspireerd door toepassingen in de praktijk. Een planningsprobleem modelleert de toewijzing van taken aan een beperkt aantal machines met als doel een of meer doelstellingen te optimaliseren. Afhankelijk van de toepassing kan de doelstelling zijn om het aantal gebruikte machines te minimaliseren of de totale of maximale kosten te optimaliseren. Wij passen wiskundige technieken toe om deze problemen te modelleren, hun intrinsieke structuur bloot te leggen en de bijbehorende uitdagingen te identificeren. Vervolgens ontwerpen wij efficiënte algoritmen die ze waar mogelijk optimaal oplossen. Omdat veel realistische varianten computationeel moeilijk zijn, kunnen optimale oplossingen binnen redelijke tijd onbereikbaar zijn. Voor computationeel moeilijke problemen richt veel onderzoek zich op het ontwerp en de analyse van benaderingsalgoritmen. Deze algoritmen ruilen optimaliteit in voor efficiëntie, i.e., zij garanderen geen optimale oplossing maar zijn erop gericht efficiënt oplossingen te produceren die "goed" genoeg zijn. In dit proefschrift ligt onze focus op theoretische benaderingsalgoritmen voor planningsproblemen.

Hoofdstuk 2 introduceert een verbeterde bovengrens voor de efficiëntie van Nash evenwichten in planningsspellen op gerelateerde machines. De machines hebben verschillende snelheden en hanteren de shortest processing time first policy. Het doel van elke taak is haar voltooiingstijd te minimaliseren, terwijl de sociale doelstelling de som van de voltooiingstijden minimaliseert. Onze hoofdbevinding geeft een bovengrens van $2 - \frac{1}{4m-2}$ op de prijs van anarchie voor het algemene geval met m machines. Wij verbeteren deze grens tot $3/2$ voor het geval van twee machines en tot $2 - \frac{1}{2m}$ voor het algemene geval met m machines wanneer de machines deelbare snelheden hebben, i.e., als de snelheid van elke machine deelbaar is door de snelheid van iedere langzamere machine.

Hoofdstuk 3 bestudeert een planningsprobleem in een machine omgeving waarin elke machine een vooraf vastgelegde volgorde moet aanhouden voor de verwerking van de taken. Gegeven n taken, elk met verwerkingstijden en deadlines, streven wij ernaar het aantal gebruikte machines te minimaliseren met inachtneming van deadlines en behoud van de volgorde op elke machine. Wij laten zien dat het first-fit algoritme het probleem optimaal oplost in het geval van identieke verwerkingstijden en dat het een 2-benadering is in de volgende gevallen: (1) de volgorde is in overeenstemming met niet stijgende speling, (2) de volgorde is in overeenstemming met niet dalende speling, (3) de volgorde is in overeenstemming met niet stijgende deadlines, en (4) de optimale oplossing gebruikt hoogstens 3 machines. Ten slotte geven wij een $O(\log n)$ benaderingsalgoritme voor het algemene geval.

In Hoofdstuk 4 presenteren wij een afhankelijke gerandomiseerde afrondingsmethode die fractionele oplossingen afrondt naar integrale oplossingen die bepaalde harde restricties op de output respecteren en Chernoffachtige concentratie eigenschappen behouden. In tegenstelling tot eerdere afhankelijke afrondingsschema's garandeert ons algoritme dat de kostprijs van de afgeronde integrale oplossing niet hoger is dan die van de fractionele oplossing. Ons algoritme werkt voor een klasse van toewijzingsproblemen met restricties die vergelijkbaar zijn met die in eerder werk. In een niet triviale combinatie van ons algemene resultaat met een klassieke aanpak van Shmoys en Tardos [101] en recentere lineaire programmeringstechnieken ontwikkeld voor de restricted assignment variant door Bansal en Sviridenko [11] en door Davies, Rothvoss en Zhang [42] leiden wij een $O(\log n)$ benaderingsalgoritme af voor het *Budgeted Santa Claus Problem*. In deze nieuwe variant is het doel om artikelen met verschillende waardes toe te wijzen aan spelers, de minimale waarde die een speler ontvangt te maximaliseren en te voldoen aan een budgetrestrictie op de kosten van toewijzing tussen speler en middel.

Curriculum Vitae

Arman Rouhani was born on February 1, 1995, in Zanjan, Iran. He completed his high school education in 2013 and subsequently began his Bachelor of Science studies at the Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan, Iran. In 2018, he obtained his B.Sc. degree in Information Technology Engineering and was recognized as an exceptional talent, which granted him direct admission to the Master of Science program in Computer Science at the Department of Computer Science and Information Technology, IASBS. In 2021, he joined the Department of Quantitative Economics at Maastricht University as a Ph.D. candidate under the supervision of Dr. André Berger and Dr. Marc Schröder. The findings of the research conducted during his time as a Ph.D. candidate have been published in journals and conferences, and he has presented these papers at various international conferences and workshops. In September 2025, Arman started a lecturer position at Maastricht University

