Algorithmic and Game-Theoretic Perspectives on Scheduling

by

Nelson A. Uhan

A.B., Applied Mathematics
Harvard University, 2002

S.M., Applied Mathematics
Harvard University, 2002

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2008

Author . . . .
Sloan School of Management
May 15, 2008

Certified by . . . .
Andreas S. Schulz
Associate Professor of Mathematics of Operations Research
Thesis Supervisor

Accepted by . . . .
Dimitris Bertsimas
Co-Director, Operations Research Center
Boeing Professor of Operations Research

Algorithmic and Game-Theoretic Perspectives on Scheduling

by

Nelson A. Uhan

Submitted to the Sloan School of Management on May 15, 2008, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Operations Research

Abstract

In this thesis, we study three problems related to various algorithmic and game-theoretic aspects of scheduling.

First, we apply ideas from cooperative game theory to study situations in which a set of agents faces supermodular costs. These situations appear in a variety of scheduling contexts, as well as in some settings related to facility location and network design. Although cooperation is unlikely when costs are supermodular, in some situations, the failure to cooperate may give rise to negative externalities. We study the least core value of a cooperative game—the minimum penalty we need to charge a coalition for acting independently that ensures the existence of an efficient and stable cost allocation—as a means of encouraging cooperation. We show that computing the least core value of supermodular cost cooperative games is strongly NP-hard, and design an approximation framework for this problem that in the end, yields a $(3 + \epsilon)$-approximation algorithm. We also apply our approximation framework to obtain better results for two special cases of supermodular cost cooperative games that arise from scheduling and matroid optimization.

Second, we focus on the classic precedence-constrained single-machine scheduling problem with the weighted sum of completion times objective. We focus on so-called 0-1 bipartite instances of this problem, a deceptively simple class of instances that has virtually the same approximability behavior as arbitrary instances. In the hope of improving our understanding of these instances, we use models from random graph theory to look at these instances with a probabilistic lens. First, we show that for almost all 0-1 bipartite instances, the decomposition technique of Sidney (1975) does not yield a non-trivial decomposition. Second, for almost all 0-1 bipartite instances, we give a lower bound on the integrality gap of various linear programming relaxations of this problem. Finally, we show that for almost all 0-1 bipartite instances, all feasible schedules are arbitrarily close to optimal.

Finally, we consider the problem of minimizing the sum of weighted completion times in a concurrent open shop environment. We present some interesting properties of various linear programming relaxations for this problem, and give a combinatorial primal-dual 2-approximation algorithm.

Thesis Supervisor: Andreas S. Schulz
Title: Associate Professor of Mathematics of Operations Research

Acknowledgments

First and foremost, I would like to express my sincerest gratitude to my advisor, Andreas S. Schulz, for his guidance, wisdom, and support during my time at MIT. I have learned tremendously from his mathematical insights and professional advice, and his tireless enthusiasm and relentless drive for excellence have really been an inspiration to me. Also, I thank him for just being a cool person to work with and to know.

I am also greatly indebted to Maurice Queyranne and David Gamarnik for generously making the time in their busy schedules to serve on my thesis committee. I have greatly benefited from their support, advice, and expertise over the past year. In addition, thanks to Monaldo Mastrolilli and Ola Svensson for also sharing their insights on some of the research in this thesis.

I would also like to thank Özlem Ergun for graciously hosting me at Georgia Tech during the spring semester of 2006, and David Shmoys for kindly hosting me at Cornell during July of 2007. Both of these visits were enjoyable and intellectually stimulating. Thanks to Doug Altner for all the interesting discussions we had while I was at Georgia Tech.

The ORC has really been a special home away from home these past five years, and I am grateful to everyone at the ORC for making my time at MIT pleasant, memorable, and productive. Special thanks are in order to Paulette, Laura, Andrew, and Veronica for running a tight ship, and for always being extremely helpful. Thanks to Théo and Tim for all the games of Tetris Attack, Mohamed for the greatest comic strip of all time, Hamed for knowing lots of secrets, Dave C. for all the snacks I pilfered from his desk, Pavithra for always accepting my offering of jellybeans, Mike for sharing clips from American Gladiator, Katy for sharing my addiction to Diet Coke, Lincoln for Pocky and scotch, Ilan, Ruben, and Dan for their Tetris commentaries, Carol and Susan for being regular lunch buddies, and everyone else who stopped by my desk to chat. Thanks also to the current and past friends of the Mittagsseminar, including Nico, José, Pranava, Juliane, Dave G., Diana, Shashi, and Claudio, for giving good talks and being around to share interesting ideas. Last but not least, thanks to Margrét for being my closest friend, and making my last two years at MIT the best.

My friends outside the ORC have also been a great source of support these past five years. Thanks especially to SFG for always being there during the craziest of times, and to the Boston Lindy Hop community for all the dances.

Finally, I would like to thank my family for their unconditional love and support.

Cambridge, May 2008 *Nelson A. Uhan*

Contents

1 Introduction 11

2 Encouraging Cooperation In Sharing Supermodular Costs 15

2.1 Introduction . . . 15

2.1.1 Previous related work . . . 17

2.1.2 Contributions of this work . . . 17

2.2 A class of optimization problems with supermodular optimal costs . . . 18

2.3 Complexity and approximation . . . 21

2.3.1 Computational complexity . . . 21

2.3.2 Approximation by fixing a cost allocation . . . 22

2.3.3 Approximation without fixing a cost allocation . . . 24

2.4 A special case from single-machine scheduling . . . 29

2.4.1 Key properties of the least core of scheduling games . . . 30

2.4.2 Computational complexity . . . 33

2.4.3 Tighter bounds on approximation based on fixing a cost allocation . 33

2.5 Submodular profits and a special case from matroid optimization . . . 37

2.5.1 Matroid profit games . . . 38

2.6 Conclusion . . . 41

2.A Proof of Theorem 2.3.9 . . . 43

2.A.1 Preliminaries . . . 43

2.A.2 Approximate separation and non-emptiness . . . 43

2.B Maximizing non-monotone submodular functions . . . 49

2.C Additional Examples . . . 51

3 Scheduling Meets Random Graphs: A Probabilistic Analysis of Precedence-Constrained Scheduling 55

3.1 Introduction . . . 55

3.1.1 Contributions of this work . . . 56

3.2 Definitions and background . . . 57

3.2.1 Sidney's decomposition theory . . . 57

3.2.2 IP formulations, LP relaxations and the minimum vertex cover problem . . . 58

3.2.3 0-1 bipartite instances . . . 60

3.3 Sidney-decomposability and 0-1 bipartite instances . . . 61

3.3.1 Almost all 0-1 bipartite instances are non-Sidney-decomposable . . 63

3.4 Two-dimensional Gantt charts and 0-1 bipartite instances 70
3.4.1 Almost all 0-1 bipartite instances have all schedules almost optimal 74
3.4.2 A lower bound on integrality gaps for almost all 0-1 bipartite instances 75
3.5 Conclusion 77
3.A Examples 79

4 Scheduling in concurrent open shop environments 81
4.1 Introduction 81
4.1.1 Contributions of this work 82
4.2 Mathematical programming formulations and relaxations 82
4.2.1 Completion time variables 82
4.2.2 Linear ordering variables 85
4.2.3 Relative strength of LP relaxations 86
4.2.4 Integrality gaps for LP relaxations 87
4.3 A combinatorial primal-dual 2-approximation algorithm 88
4.4 Conclusion 93
4.A Hardness of approximation 95
4.B Additional Examples 97

A Review of Essential Background 101
A.1 Approximation algorithms 101
A.2 Notation for machine scheduling 102

Bibliography 103

List of Figures

2-1 Precedence constraints for Example 2.C.3 52

3-1 An example of a two-dimensional Gantt chart. 71
3-2 2D Gantt chart depicting region R_B. 72
3-3 2D Gantt chart that satisfies (R-α) with the smallest area under its work curve. 75
3-4 Precedence constraints for the instance in Example 3.A.1. 79
3-5 Precedence constraints for the instance in Example 3.A.2. 80

Chapter 1

Introduction

In the field of scheduling, we are concerned with the problem of effectively allocating limited resources to tasks over time. Scheduling problems are found everywhere: examples include manufacturing, project management, computer operating system design, compiler optimization, and Internet advertising. Many scheduling problems, however, are extremely difficult to solve. The study of scheduling problems from a mathematical perspective has often triggered the development of innovative ideas in the theory and practice of mathematical programming and combinatorial optimization.

This thesis focuses on both algorithmic and game-theoretic problems in scheduling. We offer a detailed preview of the contributions of this thesis below. Each chapter in this thesis considers a different problem related to scheduling, and is designed to be self-contained. We assume the reader is familiar with the basic concepts of linear and integer programming, combinatorial optimization, approximation algorithms and the scheduling classification of Graham et al. (1979). In Appendix A, we provide a brief review of the most essential definitions common to the subsequent chapters, as well as some pointers to useful references.

Encouraging cooperation in scheduling. In Chapter 2, we apply ideas from cooperative game theory, mathematical programming, and combinatorial optimization to study settings in which agents face supermodular, or increasing marginal costs. These situations appear in a variety of scheduling contexts, as well as in some settings related to facility location and network design.

Intuitively, cooperation amongst rational agents who face supermodular costs is unlikely: the cost of adding a particular agent increases as the number of players increases, diminishing the appeal of cooperation. However, the failure to cooperate may lead to negative externalities. For example, a set of agents may need to process its jobs on a type of machine that generates excessive amounts of pollution; the agents may have the opportunity to share the cost of processing their jobs on an existing single machine, but the cost of processing their jobs is such that it is cheaper for each agent to open their own machine, and as a result, generate more pollution. An authority may be interested in methods of encouraging cooperation in these types of situations. One approach would be to directly incorporate the cost of the externalities into the processing costs; however, these externality costs may sometimes be hard to precisely define. Instead one might ask, "how much do we need to penalize a subset of agents for acting independently in order to encourage all the agents to cooperate?" This notion is captured in the *least core value* of a cooperative game. The least core value of a cooperative game is the minimum penalty we need to charge a coalition for acting independently that ensures the existence of an efficient and stable cost allocation. The set of all such cost allocations is called the *least core*.

We study the computational complexity and algorithmic aspects of computing the least core value of supermodular cost cooperative games. To motivate the study of these games, we show that a particular class of optimization problems has supermodular optimal costs. This class includes a variety of problems in combinatorial optimization, especially in machine scheduling. We show that computing the least core value of supermodular cost cooperative games is strongly NP-hard, and build a framework to approximate the least core value of these games. With recent work on maximizing submodular functions, our framework yields a $(3 + \epsilon)$-approximation algorithm for computing the least core value of supermodular cost cooperative games.

We also apply our approximation framework to two particular classes of supermodular cost cooperative games: *scheduling games* and *matroid profit games*. Scheduling games are cooperative games in which the cost to a coalition is derived from the minimum sum of weighted completion times on a single machine. By specializing some of the results for arbitrary supermodular cost cooperative games, we are able to show that the Shapley value is a cost allocation in the least core of scheduling games, and design a fully polynomial-time approximation scheme for computing the least core value of these games. Matroid profit games are cooperative games with submodular profits: the profit to a coalition arises from the maximum weight of an independent set of a matroid. We show that a cost allocation in the least core and the least core value of matroid profit games can be computed in polynomial time.

This chapter is based on joint work with Andreas S. Schulz. A preliminary version of some of the results in this chapter has appeared in Schulz and Uhan (2007).

Probabilistic analysis of precedence-constrained scheduling. The classic precedence-constrained single-machine scheduling problem with the weighted sum of completion times objective has intrigued researchers for a long time. Researchers have devoted much attention to the approximability of this problem, taking inspiration from its mathematical programming formulations, structural properties, and relationships with other combinatorial optimization problems. Currently, the best approximation algorithms all have a performance guarantee of 2, and this problem has been strongly conjectured to be inapproximable within a factor of $2 - \epsilon$ for any $\epsilon > 0$, unless P = NP.

In Chapter 3, we focus on *0-1 bipartite instances*, in which precedence constraints are represented by a bipartite partial order, with minimal jobs having unit processing time and zero weight, and maximal jobs having zero processing time and unit weight. These instances are appealing because of their simple combinatorial structure. More importantly, it turns out that these simple instances capture the inherent difficulty of the problem: Woeginger (2003) showed that a ρ-approximation algorithm for 0-1 bipartite instances implies a $(\rho + \epsilon)$-approximation algorithm for *arbitrary* instances. In other words, the approximability behavior of 0-1 bipartite instances and arbitrary instances are virtually identical. As a result, in order to obtain better approximation algorithms for this scheduling problem, it suffices to consider the family of 0-1 bipartite instances.

In an effort to better understand the properties of this simple but important class of instances, we study these instances from a probabilistic lens. One appealing feature of 0-1 bipartite instances is that they are completely defined by their precedence constraints. Since precedence relations in bipartite partial orders are independent, we can apply the Erdös-Rényi model often used in random graph theory to obtain random models of 0-1 bipartite instances. Our analysis of these random 0-1 bipartite instances yields various "almost all"-type results. First, we show that for almost all 0-1 bipartite instances, the decomposition technique of Sidney (1975) does not yield a non-trivial decomposition. Second, for almost all 0-1 bipartite instances, we give a lower bound on the integrality gap of various linear programming relaxations for this scheduling problem. For the random models of 0-1 bipartite instances that we study, this lower bound approaches 2 as the precedence constraints become sparser in expectation. Finally, we show that for almost all 0-1 bipartite instances, all feasible schedules are arbitrarily close to optimal. The first two results confirm that 0-1 bipartite instances are the "right" instances to study: it is known that in order to design approximation algorithms with a performance guarantee better than 2, it suffices to consider non-Sidney-decomposable instances, and that it does not suffice to consider existing linear programming relaxations. The last result offers an interesting paradox, especially in light of the difficulty of obtaining an approximation algorithm with performance guarantee better than 2.

This chapter is based on joint work with Andreas S. Schulz.

Scheduling in concurrent open shop environments. In the *concurrent open shop model*, we are given a set of jobs, where each job consists of different components to be processed on a specific machine. Components are independent of each other: in particular, components from the same job can be processed in parallel. A job is completed when all its components are completed. This model has a variety of applications, such as make-to-order manufacturing, airplane maintenance and repair, and distributed computing.

In Chapter 4, we consider the problem of minimizing the sum of weighted completion times in a concurrent open shop environment. We present some interesting properties of various linear programming relaxations for this problem. In addition, we give a simple combinatorial primal-dual 2-approximation algorithm. Although there exist approximation algorithms that achieve the same performance guarantee, these algorithms require the solution of a linear program. Our algorithm, on the other hand, runs in $O(n(m+n))$ time, where n is the number of jobs and m is the number of machines.

This chapter is based on ongoing joint work with Monaldo Mastrolilli, Maurice Queyranne, and Andreas S. Schulz.

Chapter 2

Encouraging Cooperation In Sharing Supermodular Costs

2.1 Introduction

Consider a situation in which a set of agents has the option of sharing the cost of their joint actions. For example, a set of agents, each with a job that needs to be processed by a particular machine, may decide to share the cost of optimally processing their jobs on a single machine. In these situations, the agents may or may not be motivated to cooperate together, depending on the structure of their costs. Cooperative game theory offers a mathematical framework to study the cooperative behavior of the parties involved. A *cooperative game* is a pair (N, v) where $N = \{1, \dots, n\}$ represents a set of agents, and $v(S)$ represents the cost to a subset of agents $S \subseteq N$.

In this chapter, we are concerned with situations in which agents face *supermodular* costs. A set function $v : 2^N \mapsto \mathbb{R}$ is supermodular if

$$v(S \cup \{j\}) - v(S) \leq v(S \cup \{j, k\}) - v(S \cup \{k\}) \qquad \text{for all } S \subseteq N \setminus \{j, k\}. \quad (2.1.1)$$

In words, supermodularity captures the notion of increasing marginal costs. We study cooperative games (N, v) where v is nonnegative, supermodular, and $v(\emptyset) = 0$. We call such games *supermodular cost cooperative games.* Supermodularity often naturally arises in situations in which the costs are intimately tied with congestion effects. It has been shown that several variants of the facility location problem have supermodular costs (Nemhauser et al. 1978), and as we will show later, various problems from scheduling and network design also exhibit supermodular costs.

Intuitively, cooperation amongst rational agents who face supermodular costs is unlikely: as the size of a coalition grows, the marginal cost associated with adding a particular agent increases, diminishing the appeal of cooperation. Various solution concepts from cooperative game theory help us formalize this intuition. Suppose $x \in \mathbb{R}^N$ is a cost allocation vector: for all $i \in N$, x_i represents the cost allocated to agent i. (For notational convenience, for any vector x we define $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$.) The prominent solution concept for cooperative games is the *core* (Gillies 1959). The core of a cooperative game (N, v) is the set of all cost allocations x such that

$$x(N) = v(N), \tag{2.1.2}$$
$$x(S) \leq v(S) \quad \text{for all } S \subseteq N. \tag{2.1.3}$$

The condition (2.1.2) requires that a cost allocation in the core is *efficient*: the total cost allocated to all agents, $x(N)$, is equal to the cost of all agents cooperating, $v(N)$. The conditions (2.1.3) guarantee that a cost allocation in the core is *stable*: no subset of agents, or coalition, would be better off by abandoning the rest of the agents and acting on its own. The existence of an efficient and stable cost allocation—in other words, a non-empty core—can be seen as a rudimentary indication that cooperation is attainable. It is well-known that when costs are submodular[1], the core is non-empty (Shapley 1971). On the other hand, it is straightforward to see that for supermodular cost cooperative games, the core is in fact empty (as long as costs are not modular[2]).

In certain situations, the failure to cooperate may give rise to negative externalities. Consider the following example. A set of agents needs to process its jobs on a machine that generates an excessive amount of pollution. The agents have the opportunity to share the cost of processing their jobs on an existing single machine, but the cost of processing their jobs is such that it is cheaper for each agent to open their own machine, and as a result, generate more pollution. An authority may be interested in reducing such negative externalities. One approach would be to incorporate the cost of the pollution externalities directly into the processing costs; however, these externality costs may be hard to precisely define. Instead, one might ask, "How much do we need to charge for opening an additional machine in order to encourage all agents to share a single machine?" For an arbitrary cooperative game, the analogous question is, "How much do we need to penalize a coalition for acting independently in order to encourage all the agents to cooperate?" This notion is captured in the *least core value* of a cooperative game. The *least core* of a cooperative game (N, v) is the set of cost allocations x that are optimal solutions to the linear optimization problem

$$\begin{aligned} z^* = \text{minimize} \quad & z \\ \text{subject to} \quad & x(N) = v(N), \\ & x(S) \leq v(S) + z \quad \text{for all } S \subseteq N,\ S \neq \emptyset, N \end{aligned} \tag{LC}$$

(Shapley and Shubik 1966; Maschler et al. 1979). The optimal value z^* of (LC) is the least core value[3] of the game (N, v). In words, the least core value z^* is the minimum penalty we need to charge a coalition for acting independently that ensures a basic prerequisite for cooperation is satisfied: the existence of an efficient and stable cost allocation. Note that the

[1]A function v is submodular if $-v$ is supermodular.

[2]A set function is modular if it is both submodular and supermodular.

[3]Adding the inequalities $x_i \leq v(\{i\}) + z$ for all $i \in N$ and using the equality $x(N) = v(N)$, we can bound z^* below by $(v(N) - \sum_{i \in N} v(\{i\}))/|N|$. So as long as costs are finite, the least core value is well defined. Moreover, if v is supermodular and $v(\emptyset) = 0$, then $z^* \geq 0$.

linear optimization problem (LC) is in fact equivalent to the optimization problem

$$z^* = \min_{x : x(N) = v(N)} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S),$$

where $e(x, S) = x(S) - v(S)$ for all $S \subseteq N$. The quantity $e(x, S)$ is the *dissatisfaction* of a subset of agents S under a cost allocation x: it is the extra cost that S pays when costs are allocated according to x.[4] A cost allocation in the least core therefore minimizes the maximum dissatisfaction of any subset of agents.

2.1.1 Previous related work

Cooperative games whose costs are determined by various problems in operations research and computer science have been studied before. A short and necessarily incomplete list of examples includes assignment games (Shapley and Shubik 1971), linear production games (Owen 1975), minimum-cost spanning tree games (Granot and Huberman 1981), traveling salesman games (Potters et al. 1991), scheduling-related games (Curiel et al. 1989; Maniquet 2003; Mishra and Rangarajan 2005), facility location games (Goemans and Skutella 2004), newsvendor games and inventory centralization games (Gerchak and Gupta 1991; Hartman et al. 2000), and economic lot-sizing games (van den Heuvel et al. 2005; Chen and Zhang 2006). The textbook by Peleg and Sudhölter (2007) is a good introduction to cooperative game theory.

A fair amount of attention has been devoted to the least core of various cooperative games in the economics and game theory literature (e.g. Peleg and Rosenmüller 1992; Reijnierse et al. 1996; Einy et al. 1998, 1999; Lehrer 2002). In addition, the computational complexity of computing an element in the least core has been studied previously in several contexts. Faigle et al. (2000) showed that computing an element in the least core of minimum-cost spanning tree games is NP-hard. Kern and Paulusma (2003) presented a polynomial description of the linear optimization problem (LC) for cardinality matching games. Faigle et al. (2001) showed that by using the ellipsoid method, a so-called pre-kernel element in the least core of a cooperative game can be efficiently computed if the maximum dissatisfaction can be efficiently computed for any given efficient cost allocation. Properties of the least core value, on the other hand, seem to have been largely ignored.

2.1.2 Contributions of this work

In this chapter, we study the computational complexity and approximability of the least core value of supermodular cost cooperative games. In Section 2.2, we motivate the interest in supermodular cost cooperative games by providing a class of optimization problems whose optimal costs are supermodular. This class of optimization problems includes a variety of classical scheduling and network design problems. Then, in Section 2.3, we show that finding the least core value of supermodular cost cooperative games is NP-hard, and design approximation algorithms for computing the least core value of these games, using oracles

[4]This quantity is sometimes referred to as the *excess* of a coalition of agents in the cooperative game theory literature.

that approximately determine maximally violated constraints. As a by-product, we also show how to compute accompanying approximate least core cost allocations.

In Sections 2.4 and 2.5, we apply our results to two special cases of supermodular cost cooperative games that arise from scheduling and matroid optimization. In Section 2.4, we study *scheduling games*, or cooperative games in which the costs are derived from the minimum sum of weighted completion times on a single machine. By improving on some of the results for general supermodular cost cooperative games, we are able to give an explicit formula for an element of the least core of scheduling games, and design a fully polynomial time approximation scheme for computing the least core value of these games. Finally, in Section 2.5, we consider a cooperative game with submodular profits: *matroid profit games*. Matroid profit games are cooperative games in which the profit to a coalition arises from the maximum weight of an independent set of a matroid. Some scheduling and network design problems have been shown to be special cases of finding a maximum weight independent set of a matroid. Using the framework established in Section 2.3 with the appropriate natural modifications, we show that the least core value of these games can be computed in polynomial time.

2.2 A class of optimization problems with supermodular optimal costs

We begin by providing some motivation for looking at cooperative games with supermodular costs. The problem of minimizing a linear function over a *supermodular polyhedron*—a polyhedron of the form $\{x \in \mathbb{R}^N : x(S) \geq u(S) \text{ for all } S \subseteq N\}$, where $u : 2^N \mapsto \mathbb{R}$ is supermodular—arises in many areas of combinatorial optimization, especially in scheduling. For example, Wolsey (1985) and Queyranne (1993) showed that the convex hull of feasible completion time vectors on a single machine is a supermodular polyhedron. Queyranne and Schulz (1995) showed that the convex hull of feasible completion time vectors for unit jobs on parallel machines with nonstationary speeds is a supermodular polyhedron. The scheduling problem they considered includes various classical scheduling problems as special cases. Goemans et al. (2002) showed that for a scheduling environment consisting of a single machine and jobs with release dates, the convex hull of mean busy time vectors of preemptive schedules is a supermodular polyhedron.

In this section, we show that the optimal cost of minimizing a linear function over a supermodular polyhedron is a supermodular function. As a result, by studying supermodular cost cooperative games, we are able to gain insight into the sharing of optimal costs for a wide range of situations.

Theorem 2.2.1. *Let N be a finite set, and let $u : 2^N \mapsto \mathbb{R}$ be a supermodular function. If $f_j \geq 0$ for all $j \in N$, then the function $v : 2^N \mapsto \mathbb{R}$ defined by*

$$v(S) = \min \left\{ \sum_{j \in S} f_j x_j : x(A) \geq u(A) \textit{ for all } A \subseteq S \right\} \quad \textit{for all } S \subseteq N \qquad (2.2.1)$$

is supermodular on N.

Proof. Let S be a subset of N with s elements. Without loss of generality, we assume that

$$S = \{1, \dots, j-1, j, j+1, \dots, k-1, k, k+1, \dots, s\},$$

and that the associated costs are nonincreasing: $f_1 \geq \cdots \geq f_s$. Define $S^i = \{1, \dots, i\}$ for $i = 1, \dots, s$ and $S^0 = \emptyset$.

It is well known that minimizing a linear function over a supermodular polyhedron can be achieved by a greedy procedure (Edmonds 1970). In particular, the value of $v(S)$ is

$$\begin{aligned} v(S) &= \sum_{i=1}^{s} f_i \big(u(S^i) - u(S^{i-1})\big) \\ &= \sum_{i=1}^{s} f_i u(S^i) - \sum_{i=0}^{s-1} f_{i+1} u(S^i) \\ &= \sum_{i=1}^{s-1} (f_i - f_{i+1}) u(S^i) + f_s u(S^s) - f_1 u(S^0). \end{aligned}$$

Similarly, we have that

$$\begin{aligned} v(S \setminus \{j\}) = &\sum_{i=1}^{j-2} (f_i - f_{i+1}) u(S^i) + (f_{j-1} - f_{j+1}) u(S^{j-1}) \\ &+ \sum_{i=j+1}^{s-1} (f_i - f_{i+1}) u(S^i \setminus \{j\}) + f_s u(S^s \setminus \{j\}) - f_1 u(S^0), \\ v(S \setminus \{k\}) = &\sum_{i=1}^{k-2} (f_i - f_{i+1}) u(S^i) + (f_{k-1} - f_{k+1}) u(S^{k-1}) \\ &+ \sum_{i=k+1}^{s-1} (f_i - f_{i+1}) u(S^i \setminus \{k\}) + f_s u(S^s \setminus \{k\}) - f_1 u(S^0), \\ v(S \setminus \{j,k\}) = &\sum_{i=1}^{j-2} (f_i - f_{i+1}) u(S^i) + (f_{j-1} - f_{j+1}) u(S^{j-1}) \\ &+ \sum_{i=j+1}^{k-2} (f_i - f_{i+1}) u(S^i \setminus \{j\}) + (f_{k-1} - f_{k+1}) u(S^{k-1} \setminus \{j\}) \\ &+ \sum_{i=k+1}^{s-1} (f_i - f_{i+1}) u(S^i \setminus \{j,k\}) + f_s u(S^s \setminus \{j,k\}) - f_1 u(S^0). \end{aligned}$$

For any $l \in N$ and $A \subseteq N \setminus \{l\}$, we define $\Delta(A, l)$ to be the marginal value of adding l to A; that is, $\Delta(A, l) = u(A \cup \{l\}) - u(A)$. Therefore,

$v(S \setminus \{j\}) - v(S \setminus \{j,k\})$

$$
\begin{aligned}
&= (f_{k-1} - f_k)u(S^{k-1} \setminus \{j\}) + (f_k - f_{k+1})u(S^k \setminus \{j\}) \\
&\quad - (f_{k-1} - f_{k+1})u(S^{k-1} \setminus \{j\}) + \sum_{i=k+1}^{s-1} (f_i - f_{i+1})\big(u(S^i \setminus \{j\} - u(S^i \setminus \{j,k\})\big) \\
&\quad + f_s\big(u(S^s \setminus \{j\}) - u(S^s \setminus \{j,k\})\big) \\
&= (f_k - f_{k+1})\Delta(S^{k-1} \setminus \{j\}, k) + \sum_{i=k+1}^{s} (f_i - f_{i+1})\Delta(S^i \setminus \{j,k\}, k) \\
&\quad + f_s\Delta(S^s \setminus \{j,k\}, k).
\end{aligned}
$$

Similar to above, we consider the effects of adding k to $S \setminus \{k\}$:

$$
\begin{aligned}
&v(S) - v(S \setminus \{k\}) \\
&\quad = (f_{k-1} - f_k)u(S^{k-1}) + (f_k - f_{k+1})u(S^k) - (f_{k-1} - f_{k+1})u(S^{k-1}) \\
&\qquad + \sum_{i=k+1}^{s-1} (f_i - f_{i+1})\big(u(S^i) - u(S^i \setminus \{k\})\big) + f_s\big(u(S^s) - u(S^s \setminus \{k\})\big) \\
&\quad = (f_k - f_{k+1})\Delta(S^{k-1}, k) + \sum_{i=k+1}^{s-1} (f_i - f_{i+1})\Delta(S^i \setminus \{k\}, k) + f_s\Delta(S^s \setminus \{k\}, k).
\end{aligned}
$$

By the supermodularity of u, we have that $\Delta(A,k) \leq \Delta(B,k)$ for any $A \subseteq B \subseteq N \setminus \{k\}$. This, in addition with the fact that $f_i - f_{i+1} \geq 0$ for all $i = 1, \ldots, s-1$ and $f_s \geq 0$, implies that

$$
\begin{aligned}
&v(S) - v(S \setminus \{k\}) \\
&\quad = (f_k - f_{k+1})\Delta(S^{k-1}, k) + \sum_{i=k+1}^{s-1} (f_i - f_{i+1})\Delta(S^i \setminus \{k\}, k) + f_s\Delta(S^s \setminus \{k\}, k) \\
&\quad \geq (f_k - f_{k+1})\Delta(S^{k-1} \setminus \{j\}, k) + \sum_{i=k+1}^{s} (f_i - f_{i+1})\Delta(S^i \setminus \{j,k\}, k) \\
&\qquad + f_s\Delta(S^s \setminus \{j,k\}, k) \\
&\quad = v(S \setminus \{j\}) - v(S \setminus \{j,k\}).
\end{aligned}
$$

Therefore, v is supermodular. □

As mentioned above, the problem of minimizing a linear function over a supermodular polyhedron models a variety of scheduling problems. As a result, we have the following corollary of Theorem 2.2.1.

Corollary 2.2.2. *If for all $S \subseteq N$, $v(S)$ is the objective value of optimally scheduling jobs in S for the problem* (a) $1 \,|\,|\, \sum w_j C_j$, (b) $Q \,|\, p_j = 1 \,|\, \sum w_j C_j$, (c) $P \,|\, p_j = 1,\ r_j$ *integral* $|\, \sum w_j C_j$, (d) $P \,|\,|\, \sum C_j$, *or* (e) $1 \,|\, r_j$, *pmtn* $|\, \sum w_j M_j$, *then v is supermodular.*

Proof. The work of Wolsey (1985) and Queyranne (1993) implies (a). The work of Queyranne and Schulz (1995) implies (b)-(d); in particular, (d) follows since the problem $P \,|\,|\, \sum C_j$ can be equivalently cast as the problem $P \,|\, p_j = 1 \,|\, \sum w_j C_j$. Finally, the work of Goemans et al. (2002) implies (e). □

Unfortunately, Corollary 2.2.2(d) does not extend to the case with arbitrary weights and processing times, as shown in Example 2.C.1. In addition, one can show that the scheduling problems $1 \,|\, r_j \,|\, \sum C_j$ and $1 \,|\, \text{prec} \,|\, \sum C_j$ do not have supermodular optimal costs, as can be seen in Example 2.C.2 and Example 2.C.3, respectively.

Using almost identical techniques to those in the proof of Theorem 2.2.1, we can also show that maximizing a nonnegative linear function over a *submodular polyhedron*—a polyhedron of the form $\{x \in \mathbb{R}^N : x(S) \leq u(S) \text{ for all } S \subseteq N\}$ where $u : 2^N \mapsto \mathbb{R}$ is submodular—has submodular optimal values.

Theorem 2.2.3. *Let N be a finite set, and let $u : 2^N \mapsto \mathbb{R}$ be a submodular function. If $f_j \geq 0$ for all $j \in N$, then the function $v : 2^N \mapsto \mathbb{R}$ defined by*

$$v(S) = \max \left\{ \sum_{j \in S} f_j x_j : x(A) \leq u(A) \text{ for all } A \subseteq S \right\} \quad \text{for all } S \subseteq N$$

is submodular on N.

An important example of maximizing a nonnegative linear function over a submodular polyhedron is finding a maximum weight independent set of a matroid; in fact, a version of Theorem 2.2.3 has been mentioned in the literature for this special case (see Nemhauser and Wolsey 1988, page 715). One example of finding a maximum weight independent set of matroid is finding a maximum weight forest in an undirected graph (Birkhoff 1935; Whitney 1935). Later, in Section 2.5, we study a cooperative game in which the profit to a coalition arises from the maximum weight of an independent set of a matroid.

2.3 Complexity and approximation

We now turn our attention to the computational complexity and approximability of computing the least core value of an arbitrary supermodular cost cooperative game (N, v). Note that an arbitrary supermodular function v may not be compactly encoded. Therefore, for the remainder of this section we assume that we have a value-giving oracle for v. In addition, for the remainder of this chapter, we assume that there are at least two agents ($n \geq 2$).

2.3.1 Computational complexity

Theorem 2.3.1. *Computing the least core value of supermodular cost cooperative games is strongly NP-hard.*

Proof. We show that any instance of the strongly NP-hard maximum cut problem on an undirected graph (Garey et al. 1976) can be reduced to an instance of computing the least core value of a supermodular cost cooperative game. Consider an arbitrary undirected graph $G = (N, E)$. Let $\kappa : 2^N \mapsto \mathbb{R}$ be the *cut function* of G; that is,

$$\kappa(S) = \left|\{\{i, j\} \in E : i \in S, j \in N \setminus S\}\right|.$$

Also, let the function $\eta : 2^N \mapsto \mathbb{R}$ be defined as

$$\eta(S) = \left|\{\{i, j\} \in E : i \in S, j \in S\}\right|.$$

Clearly, η is nonnegative. Using the increasing marginal cost characterization of supermodularity (2.1.1), it is straightforward to see that η is supermodular. Using counting arguments, it is also straightforward to show that

$$\eta(S) + \eta(N \setminus S) + \kappa(S) = \eta(N)$$

for any $S \subseteq N$.

Now consider the supermodular cost cooperative game (N, v), where $v(S) = 2\eta(S)$ for all $S \subseteq N$. For each player $i \in N$, we define the cost allocation $x_i = \deg(i)$, where $\deg(i)$ denotes the degree of node i in G. In addition, let $z = \max_{S \subseteq N, S \neq \emptyset, N} \kappa(S)$. Note that $x(N) = \sum_{i \in N} \deg(i) = v(N)$, and for all $S \subseteq N$, $S \neq \emptyset, N$,

$$z \geq \kappa(S) = (2\eta(S) + \kappa(S)) - 2\eta(S) = x(S) - v(S).$$

Therefore, (x, z) is a feasible solution to (LC). Now suppose (x^*, z^*) is an optimal solution to (LC). Adding the inequalities $x^*(S) \leq v(S) + z^*$ and $x^*(N \setminus S) \leq v(N \setminus S) + z^*$ for any $S \subseteq N$, $S \neq \emptyset, N$, and using the equality $x^*(N) = v(N)$, we have that

$$2z^* \geq v(N) - v(S) - v(N \setminus S) = 2\kappa(S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Therefore, $z^* \geq z$. It follows that $z^* = z = \max_{S \subseteq N, S \neq \emptyset, N} \kappa(S)$. In other words, finding the least core value of (N, v) is equivalent to finding the value of a maximum cut in $G = (N, E)$. $\square$

In our proof of the above theorem, we show that for any instance of the maximum cut problem on an undirected graph, there exists a supermodular cost cooperative game whose least core value is exactly equal to the value of the maximum cut. Since the maximum cut problem is not approximable within a factor of 1.0624 (Håstad 2001), we immediately obtain the following inapproximability result:

Corollary 2.3.2. *There is no ρ-approximation algorithm for computing the least core value of supermodular cost cooperative games, where $\rho < 1.0624$, unless P = NP.*

2.3.2 Approximation by fixing a cost allocation

The above negative results indicate that it is rather unlikely that we will be able to compute the least core value of supermodular cost cooperative games exactly in polynomial time. In fact, the proof of Theorem 2.3.1 indicates that this may be difficult, even if an element of the least core is known. This motivates us to design methods with polynomial running time that approximate the least core value of these games.

As a first attempt at approximation, we fix a cost allocation x such that $x(N) = v(N)$, and then try to determine the minimum value of z such that (x, z) is feasible in the least core optimization problem (LC). Recall that the dissatisfaction of a subset of agents S under a cost allocation x is defined as $e(x, S) = x(S) - v(S)$. For any cooperative game (N, v), we define the following problem:

x-maximum dissatisfaction problem for cooperative game (N, v) (x-MD).
For a given cost allocation x such that $x(N) = v(N)$, find a subset of agents S^* whose dissatisfaction is maximum:

$$e(x, S^*) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{x(S) - v(S)\big\}.$$

We want to find a value z that is as close to $e(x, S^*)$ as possible, but *larger* than $e(x, S^*)$, since (x, z) is feasible if and only if $z \geq e(x, S^*)$. Note that an algorithm for the x-MD problem acts as a separation oracle for the vector (x, z) to the linear optimization problem (LC): if $z \geq e(x, S^*)$, then (x, z) is feasible in (LC); otherwise, we have $z < e(x, S^*)$, which implies that $x(S^*) \leq v(S^*) + z$ is a constraint violated by (x, z).

How should we fix x? We would like to ensure that the cost allocation x we choose is at least in the vicinity of the least core of (N, v), so that we do not prematurely weaken the resulting approximation to the least core value. Suppose z^* is the least core value of (N, v). For any $\rho \geq 1$, we define the *ρ-approximate least core* of (N, v) as the set of all cost allocations x such that

$$\begin{aligned} &x(N) = v(N), \\ &x(S) \leq v(S) + \rho z^* \qquad \text{for all } S \subseteq N, S \neq \emptyset, N, \end{aligned}$$

or equivalently,

$$x(N) = v(N), \qquad \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(x, S) \leq \rho z^*.$$

Note that under a cost allocation in the ρ-approximate least core of (N, v), the maximum dissatisfaction of a coalition is at most a factor ρ away from the least core value z^*–the minimum possible maximum dissatisfaction of a coalition under any efficient cost allocation.

For any set function $v : 2^N \mapsto \mathbb{R}$, we define the polytope

$$B_v = \big\{x \in \mathbb{R}^N : x(N) = v(N),\ x(S) \geq v(S) \text{ for all } S \subseteq N\big\}.$$

For an arbitrary set function v, computing an element of B_v may require an exponential number of oracle calls, or B_v may be empty. Fortunately, when v is supermodular, the vertices of B_v are computable in polynomial time, and even have explicit formulas (Edmonds 1970). It turns out that any cost allocation x in B_v is an element of the 2-approximate least core of (N, v).

Theorem 2.3.3. *Suppose (N, v) is a supermodular cost cooperative game, and x is a cost allocation in B_v. Let $e(x, S^*)$ be the optimal value of the x-maximum dissatisfaction problem for (N, v), and let z^* be the least core value of (N, v). Then, x is an element of the 2-approximate least core of (N, v), or equivalently, $e(x, S^*) \leq 2z^*$.*

Proof. Let (x^*, z^*) be an optimal solution to (LC). As in the proof of Theorem 2.3.1, we have that

$$2z^* \geq v(N) - v(S) - v(N \setminus S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Since $x \in B_v$, we can deduce that for any $S \subseteq N$, $S \neq \emptyset, N$,

$$2z^* \geq v(N) - v(S) - v(N \setminus S) = x(S) - v(S) + x(N \setminus S) - v(N \setminus S) \geq e(x, S).$$

Since the above lower bound on $2z^*$ holds for any $S \subseteq N$, $S \neq \emptyset, N$, it follows that $2z^* \geq e(x, S^*)$. □

We use this observation, in conjunction with a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), to approximate the least core value of (N, v).

Theorem 2.3.4. *Suppose (N, v) is a supermodular cost cooperative game, and x is a cost allocation in B_v. If there exists a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), then there exists a 2ρ-approximation algorithm for computing the least core value of (N, v).*

Proof. Let $\bar{S}$ be the output from a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), and let $z = \rho e(x, \bar{S})$. We show that (x, z) is a feasible solution to the optimization problem (LC), and that z is within a factor of 2ρ of z^*, the least core value of (N, v). Since $x \in B_v$, we have that $x(N) = v(N)$. Since $\bar{S}$ is output from a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), it follows that $z = \rho e(x, \bar{S}) \geq e(x, S^*) \geq x(S) - v(S)$ for all $S \subseteq N$, $S \neq \emptyset, N$. So (x, z) is a feasible solution to (LC). By Theorem 2.3.3, it follows that $z = \rho e(x, \bar{S}) \leq \rho e(x, S^*) \leq 2\rho z^*$. □

Note that the x-maximum dissatisfaction problem for a supermodular cost cooperative game is an instance of submodular function maximization. In addition, for any $x \in B_v$, the objective function $e(x, \cdot)$ of the x-maximum dissatisfaction problem is nonnegative. Feige et al. (2007) gave a $5/2$-approximation algorithm for maximizing nonnegative submodular functions. With Theorem 2.3.4, this immediately implies the following corollary.

Corollary 2.3.5. *Suppose (N, v) is a supermodular cost cooperative game. Then, there exists a 5-approximation algorithm for computing the least core value (N, v).*

2.3.3 Approximation without fixing a cost allocation

Until now, we have considered approximating the least core value of a supermodular cost cooperative game (N, v) by fixing a cost allocation x and then finding z such that (x, z) is feasible in the least core optimization problem (LC). Suppose that, instead of fixing a cost allocation in advance, we compute a cost allocation along with an approximation to the least core value. Let us assume that we have a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), for *every* x such that $x(N) = v(N)$.[5] By using the ellipsoid method with binary search, we can establish one of the main results of this work:

Theorem 2.3.6. *Suppose (N, v) is a supermodular cost cooperative game, and there exists a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), for every cost allocation x such that $x(N) = v(N)$. Let z^* be the least core value of (N, v). Then,*

(a) *there exists a ρ-approximation algorithm for computing the least core value of (N, v), and*
(b) *there exists a polynomial-time algorithm for computing a cost allocation in the ρ-approximate least core of (N, v).*

Using an approximate separation oracle in conjunction with the ellipsoid method to achieve approximate optimization has been studied previously for a variety of problems (e.g. Jansen 2003; Jain et al. 2003; Fleischer et al. 2006). The proofs for these results all depend on the structure of the constraints describing the polytope of the problem. This is the case here as well.

With Theorem 2.3.6 in hand, it remains to show how to solve the x-maximum dissatisfaction problem for a supermodular cost cooperative game (N, v), for every cost allocation x such that $x(N) = v(N)$. As we noted in Section 2.3.2, the x-maximum dissatisfaction problem for a supermodular cost cooperative game (N, v) is an instance of submodular function maximization. Unlike in Section 2.3.2, however, the objective functions for the instances of the x-maximum dissatisfaction problem that need to be solved for Theorem 2.3.6 are not necessarily nonnegative. Feige et al. (2007) designed a local-search based approximation algorithm for maximizing a submodular function $f : 2^N \mapsto \mathbb{R}$ with $f(\emptyset) \geq 0$ and $f(N) \geq 0$, that has a performance guarantee of $(3 + \epsilon)$ for any $\epsilon > 0$. Since $e(x, \emptyset) = e(x, N) = 0$ for any cost allocation x such that $x(N) = v(N)$, we obtain the following corollary.

Corollary 2.3.7. *Suppose (N, v) is a supermodular cost cooperative game. Then for any $\epsilon > 0$,*

(a) *there exists a $(3 + \epsilon)$-approximation algorithm for computing the least core value of (N, v), and*
(b) *there exists a polynomial-time algorithm for computing a cost allocation in the $(3+\epsilon)$-approximate least core of (N, v).*

By computing a cost allocation on the fly using the ellipsoid method, we are able to design an approximation algorithm for computing the least core value of a supermodular cost cooperative game with a worst-case performance guarantee of $(3 + \epsilon)$, which compares favorably to the worst-case performance guarantee of 5 for the fixed-cost-allocation-based approximation algorithm designed in Section 2.3.2. Interestingly, however, the comparison for the accompanying cost allocations of these approximation algorithms is reversed: the

[5]Note that since v is supermodular and $v(\emptyset) = 0$, for any x such that $x(N) = v(N)$, we have that $\sum_{i \in N}(x_i - v(\{i\})) \geq \sum_{i \in N} x_i - v(N) = 0$. Therefore, there must exist $i \in N$ such that $x_i - v(\{i\}) \geq 0$, and so $\max_{S \subseteq N, S \neq \emptyset, N} e(x, S) \geq 0$. This ensures that the notion of a ρ-approximation algorithm for the x-maximum dissatisfaction problem is sensible, for any given cost allocation x such that $x(N) = v(N)$.

cost allocation that is computed by the ellipsoid-method-based approximation algorithm is guaranteed to be in the $(3+\epsilon)$-approximate least core, while the cost allocation used in the fixed-cost-allocation-based approximation algorithm is guaranteed to be in the 2-approximate least core.

Before proving Theorem 2.3.6, we first need to establish some definitions and intermediate results. To simplify the exposition, for the remainder of this subsection, we assume that v is integer-valued. Suppose $K \subseteq \mathbb{R}^n$ is a polyhedron, and φ and ν are positive integers. We say that K has *facet complexity at most φ* if there exists a system of inequalities with rational coefficients that has solution set K and such that the encoding length of each inequality of the system is at most φ. We say that K has *vertex complexity at most ν* if there exist finite sets V, E of rational vectors such that $K = \text{conv}(V) + \text{cone}(E)$ and such that each of the vectors in V and E has encoding length at most ν. We will use the following well-known lemma that relates the facet complexity and the vertex complexity of a polyhedron.

Lemma 2.3.8 (Grötschel et al. 1988, 6.2.4)**.** *Let $K \subseteq \mathbb{R}^n$ be a polyhedron.*

(a) *If K has facet complexity at most φ, then K has vertex complexity at most $4n^2\varphi$.*
(b) *If K has vertex complexity at most ν, then K has facet complexity at most $3n^2\nu$.*

We define Q to be the feasible region of the optimization problem (LC):

$$Q = \{x \in \mathbb{R}^N, z \in \mathbb{R} : x(N) = v(N),\ x(S) \le v(S) + z \text{ for all } S \subseteq N, S \ne \emptyset, N\}.$$

In addition, for any fixed $\gamma \ge 0$, let

$$Q_\gamma = \{x \in \mathbb{R}^N : x(N) = v(N),\ x(S) \le v(S) + \gamma \text{ for all } S \subseteq N, S \ne \emptyset, N\}.$$

We define the strong approximate separation problem and approximate non-emptiness problem for Q_γ using $Q_{\rho\gamma}$ as its “approximation:”

Strong approximate separation problem for Q_γ (S-APP-SEP-Q_γ).
Given $x \in \mathbb{Q}^N$ such that $x(N) = v(N)$, either
(i) *assert $x \in Q_{\rho\gamma}$ or*
(ii) *find a hyperplane separating x and Q_γ.*

Approximate non-emptiness problem for Q_γ (APP-NEMPT-Q_γ).
Either
(i) *find $x \in Q_{\rho\gamma}$ or*
(ii) *assert Q_γ is empty.*

Using techniques from Grötschel et al. (1988) and Jansen (2003), we can show the following theorem. We provide the proof in Appendix 2.A.

Theorem 2.3.9. *Fix γ so that its encoding length is polynomially bounded by n and $\log v(N)$. Suppose S-APP-SEP-Q_γ can be solved in time polynomial in n and $\log v(N)$. Then APP-NEMPT-Q_γ can be solved in time polynomial in n and $\log v(N)$.*

The following lemma is a consequence of Theorem 2.3.9 and the fact that an approximation algorithm for the x-maximum dissatisfaction problem can be used to solve the approximate separation problem for x and Q_γ.

Lemma 2.3.10. *Fix γ so that its encoding length is polynomially bounded by n and* $\log v(N)$. *Suppose (N, v) is a cooperative game, and there exists a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), for all cost allocations x such that $x(N) = v(N)$. Then APP-NEMPT-Q_γ can be solved in time polynomial in n and* $\log v(N)$.

Proof. Fix some cost allocation x such that $x(N) = v(N)$. Suppose we run a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v), and it outputs $\bar{S}$. If $e(x, \bar{S}) \leq \gamma$, then for all $S \subseteq N$, $S \neq \emptyset, N$, we have that

$$x(S) - v(S) \leq \max_{\substack{S\subseteq N\\ S\neq\emptyset,N}} e(x, S) \leq \rho e(x, \bar{S}) \leq \rho\gamma,$$

and therefore $x \in Q_{\rho\gamma}$. Otherwise, $e(x, \bar{S}) > \gamma$, and for all $y \in Q_\gamma$ we have that

$$x(\bar{S}) - v(\bar{S}) > \gamma \geq y(\bar{S}) - v(\bar{S}).$$

So using a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v) allows us to solve S-APP-SEP-Q_γ in time polynomial in n and $\log v(N)$, which by Theorem 2.3.9, allows us to solve APP-NEMPT-Q_γ in time polynomial in n and $\log v(N)$. □

Finally, we are ready to prove Theorem 2.3.6. We do this by showing that using a polynomial-time algorithm for APP-NEMPT-Q_γ in conjunction with binary search yields an appropriate cost allocation and approximation to the least core value of (N, v).

Proof of Theorem 2.3.6. Suppose that (N, v) is a supermodular cost cooperative game, and $\mathcal{A}$ is an algorithm that solves APP-NEMPT-Q_γ in time polynomial in n and $\log v(N)$ for every $\gamma \geq 0$ whose encoding length is polynomially bounded by n and $\log v(N)$. Since we assume that a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v) exists for every cost allocation x such that $x(N) = v(N)$, by Lemma 2.3.10, such an algorithm $\mathcal{A}$ exists.

Consider the following algorithm:

Input: supermodular cost cooperative game (N, v) with v integer-valued; algorithm $\mathcal{A}$ that solves APP-NEMPT-Q_γ for every $\gamma \geq 0$ whose encoding length is polynomially bounded by n and $\log v(N)$.

Output: a feasible solution (x, z) to the least core optimization problem (LC) for (N, v).

1. Set the following values:

$$m = 4(n+1)^2(2(n+1) + \lceil \log(v(N)+1) \rceil + 1), \tag{2.3.1a}$$

$$M = 2^m, \tag{2.3.1b}$$

$$\epsilon = (2M)^{-2}. \tag{2.3.1c}$$

2. Using $\mathcal{A}$, find $\bar{\gamma} \in \mathbb{Q}$ by binary search on $[0, v(N)]$ such that $Q_{\bar{\gamma}-\epsilon}$ is empty, but $Q_{\rho\bar{\gamma}}$ is non-empty. Denote the vector that $\mathcal{A}$ finds in $Q_{\rho\bar{\gamma}}$ by $\bar{x}$.
3. Find $p, q \in \mathbb{Z}$ such that

$$1 \leq q \leq 2M \qquad \text{and} \qquad \left|\bar{\gamma} - \frac{p}{q}\right| < \frac{1}{2Mq}. \tag{2.3.2}$$

Use $\mathcal{A}$ to solve APP-NEMPT-$Q_{p/q}$. If $\mathcal{A}$ finds a vector in $Q_{\rho p/q}$, denote that vector by $\hat{x}$.
4. Output:
   - If $\mathcal{A}$ finds a vector $\hat{x} \in Q_{\rho p/q}$ in Step 3, and $p/q < \bar{\gamma}$, then output $(x, z) = (\hat{x}, \rho p/q)$.
   - Otherwise, output $(x, z) = (\bar{x}, \rho\bar{\gamma})$.

First, we establish that the above algorithm is well-defined, by proving the following claims:

1. *The binary search interval prescribed in Step 2 is valid.* Consider the uniform cost allocation x where $x_i = v(N)/n$ for all $i \in N$. Since v is nonnegative, $(x, v(N))$ is feasible for (LC): for any $S \subseteq N$, $S \neq \emptyset, N$, we have that $v(S) + v(N) \geq |S|v(N)/n = x(S)$. Therefore, $z^* \leq v(N)$. Since v is supermodular and $v(\emptyset) = 0$, it follows that $z^* \geq 0$. So, the least core value of (N, v) lies in the interval $[0, v(N)]$.
2. *Every trial value of $\bar{\gamma}$ in the binary search of Step 2 has encoding length polynomially bounded by n and* $\log v(N)$. Since the binary search of Step 2 is on the interval $[0, v(N)]$, the numerator and denominator of any trial value of $\bar{\gamma}$ is nonnegative. In addition, the binary search of Step 2 undergoes $\lceil \log \frac{v(N)}{\epsilon} \rceil + 1$ iterations. This implies that the denominator of any trial value of $\bar{\gamma}$ is at most

$$2^{\lceil \log \frac{v(N)}{\epsilon} \rceil + 1} \leq 2^{2 + \log \frac{v(N)}{\epsilon}} = \frac{4v(N)}{\epsilon}.$$

Since the binary search is performed on the interval $[0, v(N)]$, the numerator of any trial value of $\bar{\gamma}$ is at most $4v(N)^2/\epsilon$. By (2.3.1a)-(2.3.1c), the claim follows.
3. *The integers p and q computed in Step 3 have encoding lengths polynomially bounded by n and* $\log v(N)$. By (2.3.2), and since $M \geq 1$ and $\bar{\gamma} \in [0, v(N)]$, we have that

$$p < \bar{\gamma}q + \frac{1}{2M} \leq 2Mv(N) + 1,$$
$$p > \bar{\gamma}q - \frac{1}{2M} \geq -\frac{1}{2}.$$

Therefore, $|p| < 2Mv(N) + 1$. Since $|q| \leq 2M$, the claim follows by (2.3.1a)-(2.3.1c).

Next, we analyze the running time of the above algorithm. The algorithm makes a total of $O(\log \frac{v(N)}{\epsilon})$ calls to $\mathcal{A}$, which runs in time polynomial in n and $\log v(N)$ each time it is called. It follows by (2.3.1a)-(2.3.1c) that the total running time of $\mathcal{A}$ throughout the algorithm is polynomial in n and $\log v(N)$. By using the method of continued fractions (Grötschel et al. 1988, pp. 134-137), finding integers p and q to satisfy (2.3.2) in Step 3 of the algorithm can be done in time polynomial in n and $\log v(N)$. Therefore, the above algorithm runs in time polynomial in n and $\log v(N)$.

Finally, we analyze the quality of the solution returned by the above algorithm. We start by showing that $\min\{p/q, \bar{\gamma}\} \leq z^*$ by considering two cases:

1. $\bar{\gamma} - \epsilon < z^* < \bar{\gamma}$. Consider p, q computed in Step 3 of the algorithm. Since v is integer-valued, nonnegative, and supermodular with $v(\emptyset) = 0$, $z^* = r/s$ for some $r \in \mathbb{Z}_{\geq 0}$ and $s \in \mathbb{Z}_{>0}$. Note that since v is nonnegative, supermodular, and $v(\emptyset) = 0$, the facet complexity of Q is at most $\varphi = 2(n+1) + \lceil \log(v(N)+1) \rceil + 1$. Therefore, the vertex complexity of Q is at most $m = 4(n+1)^2\varphi$, and so $s \in (0, 2^m) = (0, M)$. Since
$$\bar{\gamma} - \frac{r}{s} = \bar{\gamma} - z^* < \epsilon = \frac{1}{(2M)^2} \leq \frac{1}{2Mq},$$
it follows that
$$\left|\frac{p}{q} - z^*\right| = \left|\frac{p}{q} - \frac{r}{s}\right| \leq \left|\frac{p}{q} - \bar{\gamma}\right| + \left|\bar{\gamma} - \frac{r}{s}\right| < \frac{1}{Mq} < \frac{1}{sq}.$$
Therefore, $z^* = \frac{p}{q}$. It follows that $\min\{p/q, \bar{\gamma}\} \leq z^*$.
2. $z^* \geq \bar{\gamma}$. Clearly, $\min\{p/q, \bar{\gamma}\} \leq z^*$.

With this fact in hand, we now show that the solution (x, z) computed in Step 4 of the above algorithm is feasible in the optimization problem (LC), and that $z \leq \rho z^*$. We consider the following cases:

1. $p/q < \bar{\gamma}$. In this case, we have that $p/q \leq z^*$. Consider the output of $\mathcal{A}$ in Step 3 of the algorithm:
   (a) $\mathcal{A}$ finds $\hat{x} \in Q_{\rho p/q}$. Therefore, $(x, z) = (\hat{x}, \rho p/q)$ is feasible in (LC), and $z = \rho p/q \leq \rho z^*$.
   (b) $\mathcal{A}$ asserts that $Q_{p/q}$ is empty. Therefore, $z^* > p/q$. By the arguments above, we have that $z^* \geq \bar{\gamma}$. So, $(x, z) = (\bar{x}, \rho\bar{\gamma})$ is feasible in (LC), and $z = \rho\bar{\gamma} \leq \rho z^*$.
2. $p/q \geq \bar{\gamma}$. In this case, we have that $z^* \geq \bar{\gamma}$. So, $(x, z) = (\bar{x}, \rho\bar{\gamma})$ is feasible in (LC), and $z \leq \rho z^*$. $\square$

2.4 A special case from single-machine scheduling

In this section, we study a particular supermodular cost cooperative game that arises from scheduling situations. Consider a setting where agents each have a job that needs to be processed on a machine, and any coalition of agents can potentially open their own machine. Suppose each agent $i \in N$ has a job whose processing time is $p_i \in \mathbb{R}_{>0}$ and weight is $w_i \in \mathbb{R}_{\geq 0}$. Jobs are independent, and are scheduled non-preemptively on a single machine, which can process at most one job at a time. A *scheduling game* is a cooperative game (N, v) where the cost $v(S)$ to a coalition S is the minimum sum of weighted completion times of jobs in S. If weight w_i is interpreted as agent i's per-unit-time waiting cost, then $v(S)$ can be seen as the minimum total waiting cost for agents in S. The least core value of scheduling games has a natural interpretation: it is the minimum amount we need to charge any coalition for opening a new machine in order to encourage cooperation.

Various cooperative games that arise from scheduling situations have been studied previously. In sequencing games (e.g. Curiel et al. 1989), agents—each with a job that needs to be processed—start with a feasible schedule on a fixed number of machines, and the profit assigned to a coalition of agents is the maximal cost savings the coalition can achieve by rearranging themselves. Scheduling games have received somewhat limited attention in the past; several authors have developed axiomatic characterizations of various cost sharing rules for these games (Maniquet 2003; Mishra and Rangarajan 2005).

From Corollary 2.2.2, it follows that scheduling games are indeed supermodular cost cooperative games, and the results from Section 2.3 apply. We will apply the results of Section 2.3.2, in which approximation is based on fixing a cost allocation, to finding the least core value of scheduling games. Before doing so, however, we establish some useful and interesting properties of the least core of scheduling games. These properties will in turn help us determine the computational complexity of this special case, and choose a specific cost allocation $\bar{x}$ in B_v with especially nice features. In particular, we will be able to design approximation algorithms for the $\bar{x}$-maximum dissatisfaction problem for scheduling games, as well as show a stronger translation between the approximability of the $\bar{x}$-maximum dissatisfaction problem and the approximability of the least core value of these games.

2.4.1 Key properties of the least core of scheduling games

The structure of the cost function for scheduling games allows us to explicitly express an element of the least core of scheduling games and recast the least core optimization problem as the maximization of a set function defined solely in terms of the cost function v.

Smith (1956) showed that scheduling jobs in nonincreasing order of w_j/p_j minimizes the sum of weighted completion times on a single machine. To simplify the analysis, for the remainder of this section we assume without loss of generality that

$$\frac{w_1}{p_1} \geq \cdots \geq \frac{w_n}{p_n}.$$

We consider the cost allocation $\bar{x}$ defined as follows:

$$\begin{aligned} \bar{x}_i &= \frac{1}{2}\big(v(S^i) - v(S^{i-1})\big) + \frac{1}{2}\big(v(N \setminus S^{i-1}) - v(N \setminus S^i)\big) && (2.4.1) \\ &= \frac{1}{2} w_i \sum_{j=1}^{i} p_j + \frac{1}{2} p_i \sum_{j=i}^{n} w_j && (2.4.2) \end{aligned}$$

for $i = 1, \ldots, n$, where $S^i = \{1, \ldots, i\}$ and $S^0 = \emptyset$. It is straightforward to show that $\bar{x} \in B_v$; in fact, it is a convex combination of two vertices of B_v. Since $\bar{x} \in B_v$, we have that $\bar{x}(S) \geq v(S)$ for all $S \subseteq N$. As it turns out, for scheduling games, we are able to show a more precise relationship between the cost allocation $\bar{x}(S)$ of a coalition S and its cost $v(S)$.

Lemma 2.4.1. *Suppose (N, v) is a scheduling game. Then, the cost allocation $\bar{x}$ as defined* *in* (2.4.2) *satisfies*

$$\bar{x}(S) - v(S) = \frac{1}{2}\big(v(N) - v(S) - v(N \setminus S)\big)$$

for all $S \subseteq N$.

Proof. Since jobs are assumed to be indexed according to nonincreasing weight-to-processing time ratio, by Smith's rule we know that for any $S \subseteq N$,

$$v(S) = \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j.$$

Therefore,

$$\begin{aligned}
2\big(\bar{x}(S) - v(S)\big) &= \sum_{i \in S} \sum_{j=1}^{i} w_i p_j + \sum_{i \in S} \sum_{j=i}^{n} p_i w_j - 2 \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j \\
&= \sum_{i \in S} \sum_{j=1}^{i} w_i p_j + \sum_{i \in S} \sum_{j=i}^{n} p_i w_j - \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j - \sum_{i \in S} \sum_{\substack{j=i \\ j \in S}}^{n} p_i w_j \\
&= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N\setminus S}}^{i} w_i p_j + \sum_{i \in S} \sum_{\substack{j=i \\ j \in N\setminus S}}^{n} p_i w_j \qquad (2.4.3) \\
&= \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N\setminus S}}^{i} w_i p_j + \sum_{i \in N\setminus S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j \\
&= \sum_{i \in N} \sum_{j=1}^{i} w_i p_j - \sum_{i \in S} \sum_{j=1}^{i} w_i p_j - \sum_{i \in N\setminus S} \sum_{j=1}^{i} w_i p_j \\
&\qquad + \sum_{i \in S} \sum_{\substack{j=1 \\ j \in N\setminus S}}^{i} w_i p_j + \sum_{i \in N\setminus S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j \\
&= \sum_{i \in N} \sum_{j=1}^{i} w_i p_j - \sum_{i \in S} \sum_{\substack{j=1 \\ j \in S}}^{i} w_i p_j - \sum_{i \in N\setminus S} \sum_{\substack{j=1 \\ j \in N\setminus S}}^{i} w_i p_j \\
&= v(N) - v(S) - v(N \setminus S).
\end{aligned}$$

□

With this lemma in hand, we can show the following key properties of the least core of scheduling games.

Theorem 2.4.2. *Suppose* (N, v) *is a scheduling game.*

(a) *The cost allocation* $\bar{x}$ *as defined in* (2.4.2) *is an element of the least core of* (N, v).

(b) *The least core value of* (N, v) *is*

$$z^* = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{ v(N) - v(S) - v(N \setminus S) \big\}. \tag{2.4.4}$$

Proof. Let $\bar{z}$ be the value of the right-hand side of (2.4.4). First, we show that $(\bar{x}, \bar{z})$ is a feasible solution to (LC). By Lemma 2.4.1, we have that $\bar{x}(N) = v(N)$, and for any $S \subseteq N, S \neq \emptyset, N$,

$$\bar{z} \geq \frac{1}{2}\big(v(N) - v(S) - v(N \setminus S)\big) = \bar{x}(S) - v(S).$$

Now suppose (x^*, z^*) is an optimal solution to (LC). As in the proof of Theorem 2.3.1, we obtain the following lower bound on $2z^*$:

$$2z^* \geq v(N) - v(S) - v(N \setminus S) \quad \text{for all } S \subseteq N, S \neq \emptyset, N.$$

Therefore, $z^* \geq \bar{z}$. It follows that $\bar{x}$ is an element of the least core of (N, v), and the least core value of (N, v) is $\bar{z}$. □

In addition to being an element of the least core, it happens that the cost allocation $\bar{x}$ as defined in (2.4.2) is the Shapley value[6] of scheduling games (Mishra and Rangarajan 2005). This is quite special: for a supermodular cost cooperative game (N, v), the Shapley value is not necessarily an element of the least core of (N, v). Example 2.4.3 illustrates this point.

One might wonder if the cost allocation $\bar{x}$ as defined in (2.4.1) is an element of the least core for general supermodular cost cooperative games. Note that the definition of $\bar{x}$ in (2.4.1) depends on the ordering of N. For a given permutation $\sigma : N \mapsto N$ where $\sigma(i)$ denotes the position of player $i \in N$, we define the cost allocation $\bar{x}^\sigma$ as follows:

$$\bar{x}^\sigma_{\sigma^{-1}(i)} = \frac{1}{2}\big(v(S^i) - v(S^{i-1})\big) + \frac{1}{2}\big(v(N \setminus S^{i-1}) - v(N \setminus S^i)\big)$$

for $i = 1, \dots, n$, where $S^i = \{\sigma^{-1}(1), \dots, \sigma^{-1}(i)\}$, and $S^0 = \emptyset$. The cooperative game (N, v) defined in Example 2.4.3 is an instance of a supermodular cost cooperative game (in particular, v is of the form (2.2.1)) for which the cost allocation $\bar{x}^\sigma$ is not a least core element of (N, v), for all permutations σ of N. We can also show that when (N, v) is a scheduling game, not every cost allocation in B_v is necessarily an element of the least core of (N, v).

[6]The Shapley value (Shapley 1953) of a cooperative game (N, v) is the cost allocation ϕ, where

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \big(v(S \cup \{i\}) - v(S)\big) \quad \text{for all agents } i \in N.$$

In words, the Shapley value of each agent i reflects agent i's average marginal contribution to the coalition N. The Shapley value is one of the most important solution concepts in cooperative game theory; for example, see Roth (1988).

Example 2.4.3. Consider the cooperative game (N, v) defined as follows. There are four players: $N = \{1, 2, 3, 4\}$. Each agent $i \in N$ has a processing time $p_i = i$. The cost $v(S)$ to a coalition S is the optimal value of the scheduling problem P2 | | $\sum C_j$, restricted to jobs in S. By Corollary 2.2.2, v is supermodular. The Shapley value of this game is $\phi_1 = 3/2$, $\phi_2 = 17/6$, $\phi_3 = 23/6$, and $\phi_4 = 29/6$, and the optimal value of the ϕ-maximum dissatisfaction problem for this game is $\max_{S \subseteq N, S \neq \emptyset, N} e(\phi, S) = 5/3$. However, the least core value of this game is 3/2. It is also straightforward to check that $\max_{S \subseteq N, S \neq \emptyset, N} e(\bar{x}^\sigma, S) = 2$ for all permutations σ of N. □

2.4.2 Computational complexity

Although computing the least core value of supermodular cost cooperative games is strongly NP-hard, it is still unclear if this remains the case for scheduling games. In the previous subsection, we showed that we can efficiently compute an element of the least core of scheduling games. In fact, we have an explicit formula for a least core element. Computing the least core value of scheduling games, however, remains NP-hard.

Theorem 2.4.4. *Computing the least core value of scheduling games is NP-hard, even when $w_j = p_j$ for all $j \in N$.*

Proof. By Theorem 2.4.2, the least core value of scheduling games is

$$z^* = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{v(N) - v(S) - v(N \setminus S)\big\} = \frac{1}{2} v(N) - \frac{1}{2} \min_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{v(S) + v(N \setminus S)\big\}.$$

Note that the minimization problem above is equivalent to the problem of minimizing the sum of weighted completion times of jobs in N, with weight w_j and processing time p_j for each job $j \in N$, on two identical parallel machines. Bruno et al. (1974) showed that this two-machine problem is NP-hard, even when $w_j = p_j$ for all jobs $j \in N$. □

The above result is in stark contrast to the underlying problem defining the costs in scheduling games—minimizing the sum of weighted completion times on a single machine—for which *any* order is optimal when each job has its weight equal to its processing time.

2.4.3 Tighter bounds on approximation based on fixing a cost allocation

In Section 2.3.2, we showed that for any supermodular cost cooperative game (N, v) and a cost allocation x in B_v, a ρ-approximation algorithm for the x-maximum dissatisfaction problem implies a 2ρ-approximation algorithm for computing the least core value of (N, v). It is reasonable to believe, though, that for scheduling games, we may be in a position to do better, since the cost allocation $\bar{x}$ as defined in (2.4.2) is in B_v, and is in fact an element of the least core. This is indeed the case: from Theorem 2.4.2, it follows that

$$z^* = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{\bar{x}(S) - v(S)\big\} = \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(\bar{x}, S).$$

This is exactly the $\bar{x}$-maximum dissatisfaction problem for scheduling games! Therefore, we obtain the following strengthening of Theorem 2.3.4.

Theorem 2.4.5. *Suppose there exists a ρ-approximation algorithm for the $\bar{x}$-maximum dissatisfaction problem for scheduling games, where the cost allocation $\bar{x}$ is as defined in* (2.4.2). *Then there exists a ρ-approximation algorithm for computing the least core value of scheduling games.*

Some of the proofs from the previous subsections give us insight into how to design oracles that approximately solve $\bar{x}$-MD for scheduling games. By carefully looking at the proof of Lemma 2.4.1, we can show that $\bar{x}$-MD for scheduling games is actually a special case of finding a maximum weighted cut in a complete undirected graph. In the proof of Theorem 2.4.4, we see that $\bar{x}$-MD for scheduling games is actually equivalent (with respect to optimization) to the scheduling problem $P2 \,|\,|\, \sum w_j C_j$, implying that we might be able to use or modify existing algorithms to approximately solve $\bar{x}$-MD.

A maximum-cut based approximate oracle for $\bar{x}$-MD

By (2.4.3), we have that

$$e(\bar{x}, S^*) = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \sum_{j \in S} \sum_{i \in N \setminus S} \mu_{ij} \quad \text{where} \quad \mu_{ij} = \begin{cases} w_j p_i & \text{if } i < j \\ w_i p_j & \text{if } i > j \end{cases}$$

for all $i \neq j$. Observe that $\mu_{ij} = \mu_{ji}$. So $e(\bar{x}, S^*)$ is proportional to the value of the maximum cut on a complete undirected graph with node set N and capacity μ_{ij} for arc $\{i, j\}$. Therefore, if we have a ρ-approximation algorithm for the maximum cut problem, then by Theorem 2.4.5, we have a ρ-approximation algorithm for finding the least core value of scheduling games. For example, using the approximation algorithm of Goemans and Williamson (1995) based on a semidefinite relaxation of the maximum cut problem yields a 1.1382-approximation algorithm for finding the least core value of scheduling games. However, using the algorithm of Goemans and Williamson does not exploit the special structure of this particular maximum cut problem, since their method applies for maximum cut problems on general undirected graphs.

Relationships between certain scheduling problems and maximum cut problems have been observed previously. For example, Skutella (2001) showed that for any $\rho \geq 1$, a ρ-approximation algorithm for MAXmCUT translates into an approximation algorithm for $Pm \,|\,|\, \sum w_j C_j$ with performance guarantee $(1 + m(\rho - 1))/\rho$.

A fully polynomial-time approximation scheme for $\bar{x}$-MD based on two-machine scheduling

In this subsection, we provide a fully polynomial time approximation scheme (FPTAS) for the $\bar{x}$-maximum dissatisfaction problem for scheduling games. By Theorem 2.4.2, we know that $\bar{x}$-MD is in fact

$$\max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} e(\bar{x}, S) = \frac{1}{2} \max_{\substack{S \subseteq N \\ S \neq \emptyset, N}} \big\{ v(N) - v(S) - v(N \setminus S) \big\}.$$

For simplicity of exposition, we consider maximizing $g(S) = 2e(\bar{x}, S)$ for the remainder of this subsection.

As mentioned earlier, maximizing $g(S)$ is equivalent to minimizing the sum of weighted completion times of jobs in N on two identical parallel machines. This problem is denoted by $P2 \,||\, \sum w_j C_j$ in the notation of Graham et al. (1979). $P2 \,||\, \sum w_j C_j$ is NP-complete (Bruno et al. 1974), and has an FPTAS (Sahni 1976). Although the two problems are equivalent from the optimization perspective, as is often the case with equivalent minimization and maximization problems, it is not immediately obvious if they are equivalent in terms of approximability; Example 2.C.4 shows why an FPTAS for $P2 \,||\, \sum w_j C_j$ does not imply an FPTAS for the $\bar{x}$-MD problem. We present a dynamic program that solves $\bar{x}$-MD for scheduling games exactly in pseudopolynomial time, and then convert this dynamic program into an FPTAS. This development is inspired by the FPTAS for $P2 \,||\, \sum w_j C_j$ by Sahni (1976). The analysis is similar to the analysis of the FPTAS for $P2 \,||\, C_{\max}$ by Schuurman and Woeginger.

We think of determining the maximizer S^* by scheduling the jobs in N on two machines: the jobs scheduled on machine 1 will form S^*, and the jobs scheduled on machine 2 will form $N \setminus S^*$. As usual, we consider the jobs in order of nonincreasing weight-to-processing-time ratios (i.e. $1, \dots, n$). We can partition the jobs into S^* and $N \setminus S^*$ sequentially using the following dynamic program. The state space E is partitioned into n disjoint sets, $E_1, \dots, E_n$. A schedule σ for jobs $\{1, \dots, k\}$ on two machines corresponds to a state $(a, b, c) \in E_k$. The first coordinate a is the sum of processing times of all jobs scheduled by σ on machine 1. The second coordinate b is the sum of processing times of all jobs scheduled by σ on machine 2. The third coordinate c is the running objective value: $v(\{1, \dots, k\})$ *minus* the sum of weighted completion times on two machines for σ.

Suppose jobs $1, \dots, k-1$ have already been scheduled, and job k is under consideration. If job k is scheduled on machine 1, then the running objective value increases by $w_k(a + b + p_k) - w_k(a + p_k) = w_k b$. If job k is scheduled on machine 2, then the running objective value increases by $w_k(a + b + p_k) - w_k(b + p_k) = w_k a$. This suggests the following dynamic programming algorithm.

Algorithm 2.4.6. (Exact dynamic program)

Input: scheduling game (N, v) with weights w_i, processing times p_i for all $i \in N$.
Output: the optimal value of $\bar{x}$-MD for scheduling games, $e(\bar{x}, S^*)$.

```
$E_1 = \{(p_1, 0, 0), (0, p_1, 0)\}$
For $k = 2, \dots, n$
  For every vector $(a, b, c) \in E_{k-1}$
    Put $(a + p_k, b, c + w_k b)$ and $(a, b + p_k, c + w_k a)$ in $E_k$
Find $(a, b, c) \in E_n$ with maximum $c$ value, $c^*$
```

```
    Return $e(\bar{x}, S^*) = \frac{1}{2}g(S^*) = \frac{1}{2}c^*$
```

Let $P = \sum_{i=1}^n p_i$ and $W = \sum_{i=1}^n w_i$. Each state corresponds to a point in $\{(a,b,c) \in \mathbb{Z}^3 : 0 \le a \le P, 0 \le b \le P, 0 \le c \le WP\}$. Note that for any state $(a,b,c) \in E_k$ for a given $k = 1,\dots,n$, if a is known, b is already determined, and vice versa. Therefore, the running time of this dynamic program is $O(nWP^2)$.

Let $\delta = (1 + \epsilon/(2n))^{-1}$ for some $0 < \epsilon < 1$. Note that $\delta \in (0,1)$. In addition, define $L = \lceil \log_{1/\delta} P \rceil$ and $M = \lceil \log_{1/\delta} WP \rceil$. Consider the grid formed by the points $(\delta^{-r}, \delta^{-s}, \delta^{-t})$, $r = 1,\dots,L$, $s = 1,\dots,L$, $t = 1,\dots,M$. We divide each of the state sets E_k, $k = 1,\dots,n$, into the boxes formed by the grid:

$$B(r,s,t) = \{(a,b,c) \in \mathbb{R}^3 : \delta^{-r+1} \le a \le \delta^{-r},\ \delta^{-s+1} \le b \le \delta^{-s},\ \delta^{-t+1} \le c \le \delta^{-t}\}$$
$$r = 1,\dots,L,\ s = 1,\dots,L,\ t = 1,\dots,M.$$

Observe that if (a_1,b_1,c_1) and (a_2,b_2,c_2) are in the same box,

$$\delta a_1 \le a_2 \le \frac{a_1}{\delta}, \qquad \delta b_1 \le b_2 \le \frac{b_1}{\delta}, \qquad \delta c_1 \le c_2 \le \frac{c_1}{\delta}. \tag{2.4.5}$$

We simplify the state sets E_k by using *a single point in each box* as a representative for all vectors in the same box. We denote these simplified state sets by E_k^δ. The "trimmed" dynamic program is as follows.

Algorithm 2.4.7. (Dynamic program with "trimmed" state space)

```
Input: scheduling game $(N, v)$ with weights $w_i$, processing times $p_i$ for all
    $i \in N$.
Output: an approximation to the optimal value of $\bar{x}$-MD for scheduling games,
    $e(\bar{x}, \bar{S})$.
  Pick $\epsilon \in (0,1)$, calculate $\delta$
  $E_1^\delta = \{(p_1,0,0),(0,p_1,0)\}$
  For $k = 2,\dots,n$
    For every vector $(a,b,c) \in E_{k-1}^\delta$
      Put corresponding representatives of $(a + p_k, b, c + w_k b)$ and $(a, b +
        p_k, c + w_k a)$ in $E_k^\delta$
  Find $(a,b,c) \in E_n^\delta$ with maximum $c$ value, $\bar{c}$
  Return $e(\bar{x}, \bar{S}) = \frac{1}{2}g(\bar{S}) = \frac{1}{2}\bar{c}$
```

The key property of the "trimmed" state space used in Algorithm 2.4.7 is that every element in the original state space has an element in the "trimmed" state space that is relatively close. In particular,

Lemma 2.4.8. *For every $(a,b,c) \in E_k$, there exists a vector $(a',b',c') \in E_k^\delta$ such that*

$$a' \ge \delta^k a, \qquad b' \ge \delta^k b, \qquad c' \ge \delta^k c.$$

Proof. By induction. The base case $k = 1$ holds by (2.4.5). Assume the induction hypothesis holds for $1,\dots,k-1$. Consider an arbitrary $(a,b,c) \in E_k$. The exact dynamic program puts $(a,b,c) \in E_k$ when it schedules job k. Therefore, $(a,b,c) = (\alpha + p_k, \beta, \gamma + w_k\beta)$ or $(a,b,c) = (\alpha, \beta + p_k, \gamma + w_k\alpha)$ for some $(\alpha,\beta,\gamma) \in E_{k-1}$.

Suppose $(a,b,c) = (\alpha + p_k, \beta, \gamma + w_k\beta)$ for some $(\alpha,\beta,\gamma) \in E_{k-1}$. By the induction hypothesis, there exists a vector $(\alpha',\beta',\gamma') \in E^\delta_{k-1}$ such that $\alpha' \geq \delta^{k-1}\alpha$, $\beta' \geq \delta^{k-1}\beta$, and $\gamma' \geq \delta^{k-1}\gamma$. In the kth phase, the trimmed dynamic program puts a state (a',b',c') in E^δ_k that is in the same box as $(\alpha' + p_k, \beta', \gamma' + w_k\beta')$. Therefore, since $\delta \in (0,1)$, there exists a vector $(a',b',c') \in E^\delta_k$ such that

$$\begin{aligned}
&a' \geq \delta(\alpha' + p_k) \geq \delta^k\alpha + \delta p_k \geq \delta^k(\alpha + p_k) = \delta^k a\\
&b' \geq \delta\beta' \geq \delta^k\beta = \delta^k b\\
&c' \geq \delta(\gamma' + w_k\beta') \geq \delta^k\gamma + \delta^k w_k\beta = \delta^k(\gamma + w_k\beta) = \delta^k c.
\end{aligned}$$

The case where $(a,b,c) = (\alpha, \beta + p_k, \gamma + w_k\alpha)$ for some $(\alpha,\beta,\gamma) \in E_{k-1}$ follows similarly. Therefore, the induction step is complete. □

Finally, we analyze the performance and running time of the trimmed dynamic programming algorithm.

Theorem 2.4.9. *Algorithm 2.4.7 is a fully polynomial time approximation scheme for the $\bar{x}$-maximum dissatisfaction problem for scheduling games.*

Proof. Note that each E^δ_k has at most one point from each box, or $O(L^2M)$ points. Therefore, the running time of this algorithm is $O(nL^2M)$. Since $\log z \geq (z-1)/z$ for any $z \in (0,1]$, we can bound L and M as follows:

$$\begin{aligned}
L &= \left\lceil \frac{\log P}{\log 1/\delta} \right\rceil \leq \left\lceil \left(1 + \frac{2n}{\epsilon}\right) \log P \right\rceil,\\
M &= \left\lceil \frac{\log WP}{\log 1/\delta} \right\rceil \leq \left\lceil \left(1 + \frac{2n}{\epsilon}\right) (\log W + \log P) \right\rceil.
\end{aligned}$$

Therefore, the running time of this algorithm is polynomial in n, $\log W$, $\log P$, and $1/\epsilon$.

Now we analyze the performance of this algorithm. Let $c^* = g(S^*)$, the optimal value of g. Note that there exists a vector $(a^*, b^*, c^*) \in E_n$. By Lemma 2.4.8, there exists a vector $(a',b',c') \in E^\delta_n$ such that $c' \geq \delta^n c^*$. Recall that $\delta = (1 + \epsilon/(2n))^{-1}$ for some $0 < \epsilon < 1$. Since $(1 + \epsilon/(2n))^n \leq 1 + \epsilon$, we have that $c' \geq (1 + \epsilon/(2n))^{-n} c^* \geq (1+\epsilon)^{-1} c^*$. □

Combining Theorem 2.4.5 and Theorem 2.4.9, gives us the following result.

Theorem 2.4.10. *There exists a fully polynomial time approximation scheme for computing the least core value of scheduling games.*

2.5 Submodular profits and a special case from matroid optimization

Up to this point, we have only considered cooperative games in which agents are assigned a cost for their joint actions. But what about cooperative games in which agents act together to collect a reward, or profit? Consider a cooperative game (N, v) where $v(S)$ represents the *profit* allocated to the agents in S. For these games, solution concepts should reflect the rationality of a profit allocation; for example, the core for a profit cooperative game (N, v) is defined as the set of all profit allocations x such that

$$\begin{aligned} x(N) &= v(N), \\ x(S) &\geq v(S) \quad \text{for all } S \subseteq N. \end{aligned}$$

The least core for a profit cooperative game (N, v) is defined in a similar manner: it is the set of all profit allocations x that are optimal for the problem

$$\begin{aligned} z^* = \text{minimize} \quad & z \\ \text{subject to} \quad & x(N) = v(N) \qquad\qquad \text{(LC-profit)} \\ & x(S) \geq v(S) - z \quad \text{for all } S \subseteq N, S \neq \emptyset, N. \end{aligned}$$

The least core value of (N, v) is the optimal value z^* to this optimization problem. Note that it still reflects the minimum penalty to a coalition needed to ensure the existence of an efficient and stable profit allocation.

If v is nonnegative, submodular and $v(\emptyset) = 0$, we call (N, v) a *submodular profit cooperative game*. It is straightforward to see that all the results established for supermodular cost cooperative games in Section 2.3 also hold true for submodular profit cooperative games, with the following natural modifications. For a cooperative game (N, v) with v representing profits, the dissatisfaction for any subset of agents S under a profit allocation x is defined as $e(x, S) = v(S) - x(S)$. We define the polytope B_v as $\{x \in \mathbb{R}^n : x(N) = v(N),\ x(S) \leq v(S) \text{ for all } S \subseteq N\}$. The x-maximum dissatisfaction problem for a cooperative game (N, v) with v representing profits is still to find a subset S^* such that $e(x, S^*) = \max_{S \subseteq N, S \neq \emptyset, N} e(x, S)$.

2.5.1 Matroid profit games

Consider the cooperative game (N, v), defined as follows. Each agent $i \in N$ has a job with unit processing time and a deadline $d_i \in \mathbb{Z}_{>0}$. In addition, each agent $i \in N$ has an associated profit $w_i \in \mathbb{R}_{\geq 0}$, which is earned if agent i's job is completed by its deadline. The profit $v(S)$ to any subset of agents S is the maximum profit attainable by scheduling jobs in S on a single machine. It turns out that if we define

$$\mathcal{I} = \{S \subseteq N : \text{every job in } S \text{ can be completed by its deadline}\},$$

then $(N, \mathcal{I})$ is a matroid (Gabow and Tarjan 1984). For any family of sets $\mathcal{I}$, define $\mathcal{I}|S = \{T \in \mathcal{I} : T \subseteq S\}$. In this cooperative game, $v(S)$ is the maximum w-weight of an independent set in $(S, \mathcal{I}|S)$ for any subset of agents S.

In this section, we study the following generalization of the cooperative game described above. Let $(N, \mathcal{I})$ be a matroid with weights $w_i \in \mathbb{R}_{\geq 0}$ for each $i \in N$. We define $v(S)$ as the maximum w-weight of an independent set in $(S, \mathcal{I}|S)$, for every subset of agents $S \subseteq N$. Then (N, v) defines a cooperative game where the profits to a coalition S is represented by $v(S)$. We call such games *matroid profit games*. Cooperative games that arise from matroid optimization have been considered previously. Nagamochi et al. (1997) studied the computational complexity of various solution concepts for *minimum base games*, in which for a given matroid $(N, \mathcal{I})$, the cost $v(S)$ to a coalition S is the minimum weight of a basis in $(S, \mathcal{I}|S)$. In these games, the costs to a coalition are *not* necessarily supermodular, and so the results of Section 2.3 do not apply.

By Theorem 2.2.3, matroid profit games are submodular profit cooperative games. It turns out that the x-maximum dissatisfaction problem for matroid profit games is quite tractable: we show that it can be solved exactly in polynomial time for any profit allocation x such that $x(N) = v(N)$.

Theorem 2.5.1. *Suppose (N, v) is a matroid profit game. Then for any profit allocation x such that $x(N) = v(N)$, the x-maximum dissatisfaction problem for (N, v) can be solved in polynomial time.*

Proof. Fix some profit allocation x such that $x(N) = v(N)$, and let $A = \{i \in N : x_i < 0\}$. Consider the following algorithm for the x-maximum dissatisfaction problem for (N, v):

```
Input: matroid profit game $(N, v)$ with matroid $(N, \mathcal{I})$ and weights $w_i \in \mathbb{R}_{\geq 0}$
    for all $i \in N$.
Output: an optimal solution $\bar{S}$ to $x$-MD for $(N, v)$.
  1. Compute a maximum $\bar{w}$-weight independent set $T^*$ of $(N, \mathcal{I})$, where
         $\bar{w}_i = w_i$ if $i \in A$;  $\bar{w}_i = w_i - x_i$ if $i \in N \setminus A$.
  2. Let $\bar{T} = T^* \cup (A \setminus T^*)$.
       • If $\bar{T} \neq \emptyset, N$, output $\bar{S} = \bar{T}$.
       • Otherwise, output $\bar{S} = \arg\max\{e(x, S) : S \in \{T, N \setminus T\}\}$ for an
         arbitrary $T \subseteq N, T \neq \emptyset, N$.
```

$$\bar{w}_i = \begin{cases} w_i & \text{if } i \in A \\ w_i - x_i & \text{if } i \in N \setminus A. \end{cases}$$

First, note that any optimal solution of the following relaxation of the x-maximum dissatisfaction problem

$$\max_{S \subseteq N} e(x, S) = \max_{S \subseteq N} \big\{ v(S) - x(S) \big\} \tag{2.5.1}$$

must contain all elements of A, since v is nondecreasing. Since A is fixed, the problem (2.5.1) is equivalent to

$$\max_{S \subseteq N} \big\{ v(S) - x(S \setminus A) \big\}. \tag{2.5.2}$$

We show that the independent set T^* computed in Step 1 of the above algorithm is an optimal solution to (2.5.2). Let S^* be an optimal solution to (2.5.2), and suppose that $v(S^*) - x(S^* \setminus A) > v(T^*) - x(T^* \setminus A)$. Note that without loss of generality, S^* is an independent set of $(N, \mathcal{I})$. Otherwise, there exists some $i \in S^*$ that is not in a maximum weight independent set of $(S^*, \mathcal{I}|S^*)$. If $x_i \geq 0$, then $i \in S^* \setminus A$, and can be removed without decreasing the objective value of (2.5.2); if $x_i < 0$, then $i \in A$, and removing it does not affect the objective value of (2.5.2). Therefore,

$$\begin{aligned}\bar{w}(S^*) &= w(S^*) - x(S^* \setminus A)\\ &= v(S^*) - x(S^* \setminus A)\\ &> v(T^*) - x(T^* \setminus A)\\ &= w(T^*) - x(T^* \setminus A)\\ &= \bar{w}(T^*),\end{aligned}$$

which contradicts the assumption that T^* is a maximum $\bar{w}$-weight independent set of $(N, \mathcal{I})$. So T^* is an optimal solution to (2.5.2).

Using this fact, we show that the output of the above algorithm is correct. Note that

$$\begin{aligned}\max_{\substack{S \subseteq N\\ S \neq \emptyset, N}} e(x, S) &\leq \max_{S \subseteq N} e(x, S) && \text{(2.5.3a)}\\ &= \max_{S \subseteq N} \big\{v(S) - x(S \setminus A)\big\} - x(A) && \text{(2.5.3b)}\\ &= v(T^*) - x(T^* \setminus A) - x(A) && \text{(2.5.3c)}\\ &\leq v(\bar{T}) - x(\bar{T}) && \text{(2.5.3d)}\\ &= e(x, \bar{T}). && \text{(2.5.3e)}\end{aligned}$$

We consider two cases.

1. $\bar{T} \neq \emptyset, N$. By (2.5.3a)-(2.5.3e), it follows that $\bar{T}$ is an optimal solution to x-MD for (N, v).
2. $\bar{T} = \emptyset$ or $\bar{T} = N$. In this case, $e(x, \bar{T}) = 0$. Fix some $T \subseteq N, T \neq \emptyset, N$. Since v is submodular and $v(\emptyset) = 0$, we have that

$$e(x, T) + e(x, N \setminus T) = v(T) + v(N \setminus T) - v(N) \geq 0.$$

Therefore, we must have $e(x, T) \geq 0$, or $e(x, N \setminus T) \geq 0$, or both. Without loss of generality, suppose $e(x, T) \geq 0$. It follows from (2.5.3a)-(2.5.3e) that

$$\max_{\substack{S \subseteq N\\ S \neq \emptyset, N}} e(x, S) \leq 0 \leq e(x, T).$$

Therefore, $\arg\max\{e(x, S) : S \in \{T, N \setminus T\}\}$ for any given $T \subseteq N, T \neq \emptyset, N$ is an optimal solution to x-MD for (N, v).

Since a maximum weight independent set of a matroid can be found in polynomial time (Rado 1957; Edmonds 1971), it follows that the above algorithm solves the x-maximum dissatisfaction problem for a matroid profit game (N, v) in polynomial time. □

By the discussion earlier in this section, if (N, v) is a submodular profit cooperative game and we have a ρ-approximation algorithm for the x-maximum dissatisfaction problem for (N, v) for any given profit allocation x such that $x(N) = v(N)$, then we have a ρ-approximation algorithm for computing the least core value of (N, v). This translation in approximability is accomplished by using the ellipsoid method to find feasible solutions to (LC-profit), with the x-maximum dissatisfaction problem as a separation oracle. Therefore, by Theorem 2.5.1, we immediately obtain the following theorem:

Theorem 2.5.2. *Suppose (N, v) is a matroid profit game. Then there exists a polynomial-time algorithm for*

(a) *computing the least core value of (N, v), and*

(b) *computing a cost allocation in the least core of (N, v).*

2.6 Conclusion

In a cooperative game with supermodular costs, cooperation amongst agents is unlikely: as a coalition grows, the cost of adding a particular agent increases, making the prospect of cooperation less appealing. As we showed, these situations arise when sharing the optimal costs of a variety of combinatorial optimization problems, especially in machine scheduling. In circumstances where the failure to cooperate can cause negative externalities, one may be interested in methods of encouraging cooperation. In this chapter, we considered one way of encouraging cooperation in these situations: the least core value of a cooperative game, or the minimum penalty we need to charge a coalition for acting independently that ensures the existence of an efficient and stable cost allocation. We showed that computing the least core value of supermodular cost cooperative games is strongly NP-hard, and provided a general framework for approximating the least core value of these games. This framework, with the appropriate natural modifications, can also be used to approximate the least core value of submodular profit cooperative games. Using this framework with the approximation algorithms for submodular function maximization of Feige et al. (2007), we obtained a $(3 + \epsilon)$-approximation algorithm for computing the least core value of both supermodular cost cooperative games and submodular profit cooperative games. In addition, we used this framework to design a fully polynomial-time approximation scheme for computing the least core value of scheduling games, as well as an exact polynomial-time algorithm for computing the least core value of matroid profit games.

There are several interesting directions for future research that extend from this work. For example, Faigle et al. (2000) proposed the following generalization of the least core of a cooperative game (N, v). Consider the linear optimization problem

$$\begin{aligned} z^* = \text{minimize} \quad & z \\ \text{subject to} \quad & x(N) = v(N), \\ & x(S) \le v(S) + z f(S) \quad \text{for all } S \subseteq N,\ S \neq \emptyset, N \end{aligned} \tag{f-LC}$$

for some *priority function* $f : 2^N \mapsto \mathbb{R}$. The set of all optimal cost allocations x to (f-LC) is called the *f-least core* of (N, v). Several priority functions have been considered in the literature for various cooperative games. Of course, when $f(S) = 1$ for all $S \subseteq N, S \neq \emptyset, N$, the f-least core of (N, v) is simply the least core of (N, v). Shapley and Shubik (1966) proposed the f-least core with $f(S) = |S|$. Faigle and Kern (1993) proposed the f-least core with $f(S) = v(S)$ for all $S \subseteq N, S \neq \emptyset, N$. This case has been studied for a variety of games including traveling salesman games (Faigle et al. 1998), bin-packing games (Faigle and Kern 1998), and matching games (Faigle and Kern 1993), as well as in the context of finding group strategyproof cost sharing mechanisms for a variety of cooperative games arising from combinatorial optimization problems (Immorlica et al. 2005; Pál and Tardos 2003). It would be interesting to study the f-least core for supermodular cost cooperative games, for various forms of f, as it provides a natural way to model different penalties for acting independently for different coalitions.

Another interesting direction of research related to this work is to study the *nucleolus* of supermodular cost cooperative games. Given a cost allocation x, the *excess vector* $\theta(x)$ is the $2^N - 2$ dimensional vector whose components are $e(x, S)$ for all $S \subseteq N, S \neq \emptyset, N$ in nonincreasing order. The nucleolus (Schmeidler 1969) is the cost allocation that lexicographically minimizes the excess vector $\theta(x)$. The nucleolus of a cooperative game always exists and is unique. In addition, the nucleolus is contained in the least core, and can be computed by a polynomial number of least core computations. One line of research would be to study the size of the least core of various supermodular cost cooperative games: for example, does it coincide with nucleolus, and is therefore unique? The computational complexity of computing the nucleolus of supermodular cost cooperative games is also open. It would also be interesting to investigate whether our framework for least core approximation can be used in a fruitful manner to compute cost allocations that approximate the nucleolus of supermodular cost cooperative games.

Last, but not least, the computational complexity of computing an element of the least core of supermodular cost cooperative games remains open.

2.A Proof of Theorem 2.3.9

In this appendix, we establish a result that generalizes Theorem 2.3.9: an approximate separation oracle for a given polytope, in conjunction with the ellipsoid method, can be used to either find an element in an "approximation" of that polytope, or determine that the polytope is empty. The ideas here closely follow the analyses found in Grötschel et al. (1988) and Jansen (2003).

2.A.1 Preliminaries

A *well-described polyhedron* is a triple $(K; n, \varphi)$ where $K \subseteq \mathbb{R}^n$ is a polyhedron with facet complexity at most φ. The encoding length of a well-described polyhedron $(K; n, \varphi)$ is $\varphi + n$.

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, we denote the *spectral norm of A* as

$$\|A\| = \max\left\{|\lambda| : \lambda \text{ is an eigenvalue of } A\right\} = \max\left\{|x^\top A x| : \|x\| = 1\right\}.$$

Finally, we define for any vector $a \in \mathbb{R}^n$ and positive definite matrix A, the *ellipsoid*

$$E(A, a) = \left\{x \in \mathbb{R}^n : (x - a)^\top A^{-1}(x - a) \leq 1\right\}.$$

2.A.2 Approximate separation and non-emptiness

For this subsection, we assume that $(K; n, \varphi)$ is a bounded, rational, well-described polyhedron in $\mathbb{R}^n$. In other words, $K \subseteq \mathbb{R}^n$ is a rational polytope with facet complexity at most φ. Let $\bar{K}$ be an "approximation" to K. Consider the following problem:

Strong approximate separation problem (S-APP-SEP).
Given $y \in \mathbb{Q}^n$, either
(a) *assert $y \in \bar{K}$, or*
(b) *find a hyperplane that separates y from K: find $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$ and $\|c\|_\infty = 1$.*

Suppose we have an oracle for S-APP-SEP. We use this approximate separation oracle in the ellipsoid method as follows.

Algorithm 2.A.1 (Central-cut ellipsoid method with approximate separation oracle (APP-ELL)).

Input: $\epsilon \in \mathbb{Q}$ such that $\epsilon \in (0, 1)$, bounded rational polyhedron $K \subseteq \mathbb{R}^n$ given by an oracle for S-APP-SEP, $R \in \mathbb{Q}$ such that $K \subseteq E(R^2 I, 0)$ (where I denotes the identity matrix).
Output: either
1. $y \in \bar{K}$, or

2. positive definite $A \in \mathbb{Q}^{n\times n}$, $a \in \mathbb{Q}^n$ such that $K \subseteq E(A,a)$ and $\mathrm{vol}(E(A,a)) \le \epsilon$.

1. Set the following values:

$$N = \lceil 5n|\log\epsilon| + 5n^2|\log 2R|\rceil \tag{2.A.1}$$

$$p = 8N \tag{2.A.2}$$

2. Generate the sequence of ellipsoids $E(A_0,a_0), E(A_1,a_1),\dots,E(A_N,a_N)$ as follows:
   - Initialize the sequence:

$$a_0 = 0 \tag{2.A.3}$$

$$A_0 = R^2 I \tag{2.A.4}$$

   - For $k = 0,\dots,N-1$, call S-APP-SEP oracle for K with input $y = a_k$.
   - If the S-APP-SEP oracle asserts $a_k \in \bar{K}$, return a_k. Stop.
   - If the S-APP-SEP oracle returns $c_k \in \mathbb{Q}^n$ such that

$$\|c_k\|_\infty = 1 \tag{2.A.5}$$

$$c_k^\top a_k > c_k^\top x \quad \text{for all } x \in K \tag{2.A.6}$$

     then compute

$$a_{k+1} \approx a_k - \frac{1}{n+1}\frac{A_k c_k}{\sqrt{c_k^\top A_k c_k}} \tag{2.A.7}$$

$$A_{k+1} \approx \frac{2n^2+3}{2n^2}\left(A_k - \frac{2}{n+1}\frac{A_k c_k c_k^\top A_k}{c_k^\top A_k c_k}\right) \tag{2.A.8}$$

     where "$\approx$" means the computations are done with p digits after the binary point.
   - If $k = N$, return a_N, A_N. Stop.

To prove the correctness of the algorithm, we need the following lemma.

Lemma 2.A.2 (Grötschel et al. 1988, 3.2.8-3.2.10)**.** *Let $K \subseteq \mathbb{R}^n$ be a convex set such that $K \subseteq E(R^2 I, 0)$. Let N, p be defined as in* (2.A.1)-(2.A.2)*. Suppose A_k and a_k ($k = 0,1,\dots,N$) are defined as in* (2.A.3)-(2.A.4) *and* (2.A.7)-(2.A.8)*, and c_k ($k = 0,1,\dots,N$) satisfy* (2.A.5)-(2.A.6)*. Then, the following statements hold for $k = 0,1,\dots,N$:*

(a) *A_k is positive definite.*
(b) *$\|a_k\| \le R2^k$, $\|A_k\| \le R^2 2^k$, and $\|A_k^{-1}\| \le R^{-2}4^k$.*
(c) *$K \subseteq E(A_k,a_k)$.*
(d) *$\mathrm{vol}(E(A_{k+1},a_{k+1})) \le e^{-\frac{1}{5n}}\mathrm{vol}(E(A_k,a_k))$.*

Using the above lemma, we can show:

Theorem 2.A.3. *Algorithm 2.A.1 (APP-ELL) is correct.*

Proof. Lemma 2.A.2 immediately implies that A_k and a_k ($k = 0, 1, \dots, N$) as defined in (2.A.3)-(2.A.4) and (2.A.7)-(2.A.8) are well-defined and have polynomial encoding lengths.

If the algorithm stops with $k < N$, the algorithm terminates correctly by construction. If the algorithm returns a_N and A_N, then Lemma 2.A.2 implies that $K \subseteq E(A_N, a_N)$ and

$$\begin{aligned}\operatorname{vol}(E(A_N, a_N)) &\le e^{-\frac{N}{5n}} \operatorname{vol}(E(A_0, a_0))\\ &\le e^{-\frac{N}{5n}} (2R)^n\\ &< 2^{-\frac{N}{5n}} (2R)^n\\ &\le \epsilon.\end{aligned}$$

So if $k = N$, the algorithm terminates correctly. □

Now consider the following problem:

Approximate non-emptiness problem (APP-NEMPT).
Either
(i) *find a vector $y \in \bar{K}$ or*
(ii) *assert K is empty.*

We can use APP-ELL (Algorithm 2.A.1) in conjunction with an oracle for S-APP-SEP to solve APP-NEMPT. To show this, we need the following lemma.

Lemma 2.A.4 (Grötschel et al. 1988, pp. 175-176)**.** *Let $(K; n, \varphi)$ be a well-described polyhedron. In addition, let $\epsilon = 2^{-48n^5\varphi}$. Suppose $K \subseteq E(A, a)$ where* $\operatorname{vol}(E(A, a)) \le \epsilon$. *Then there exists $f \in \mathbb{Z}^n$ and $g \in \mathbb{Z}_{>0}$ such that $f \neq 0$ and $K \subseteq \{x \in \mathbb{R}^n : f^\mathsf{T} x = g\}$. Moreover, f and g can be found in time polynomial in n, φ, and the encoding length of A^{-1}.*

Finally, we are ready to show the main result of this appendix.

Theorem 2.A.5. *Suppose there exists an algorithm that can solve S-APP-SEP in time polynomial in n and φ. Then, there exists an algorithm that can solve APP-NEMPT in time polynomial in n and φ.*

Proof. By assumption, K has facet complexity at most φ. Therefore, by Lemma 2.3.8, K has vertex complexity at most $4n^2\varphi$. Apply APP-ELL (Algorithm 2.A.1) to K with $R = 2^{4n^2\varphi}$ and $\epsilon = 2^{-48n^5\varphi}$. If APP-ELL returns a vector $y \in \bar{K}$, then we have solved APP-NEMPT, and we can stop. Otherwise, APP-ELL returns an ellipsoid $E \subseteq \mathbb{R}^n$ such that $K \subseteq E$ and $\operatorname{vol}(E) \le \epsilon$. Then, by Lemma 2.A.4, we can find $f^1 \in \mathbb{Z}^n$ and $g^1 \in \mathbb{Z}_{>0}$ such that $f^1 \neq 0$ and $K \subseteq \{x \in \mathbb{R}^n : (f^1)^\mathsf{T} x = g^1\}$. Without loss of generality, assume that $f_1^1 \neq 0$.

Suppose that we have found k linearly independent vectors $f^1, \ldots, f^k \in \mathbb{Z}^n$ and $g^1, \ldots, g^k \in \mathbb{Z}_{>0}$ such that $f^i \neq 0$ for $i = 1, \ldots, k$ and

$$K \subseteq \{x \in \mathbb{R}^n : (F_1 \quad F_2)\, x = g\}$$

where $F_1 \in \mathbb{Z}^{k\times k}$ is upper triangular with non-zero diagonal entries and $F_2 \in \mathbb{Z}^{k\times(n-k)}$ such that

$$(F_1 \quad F_2) = \begin{pmatrix} (f^1)^\top \\ \vdots \\ (f^k)^\top \end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix} g^1 \\ \vdots \\ g^k \end{pmatrix}.$$

We show how to find $f^{k+1} \in \mathbb{Z}^n$, $g^{k+1} \in \mathbb{Z}_{>0}$ such that $f^1, \ldots, f^k, f^{k+1}$ are linearly independent, $f^{k+1} \neq 0$, and

$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \ldots, (f^k)^\top x = g^k, (f^{k+1})^\top x = g^{k+1}\}.$$

Let

$$K_k = \left\{ u \in \mathbb{R}^{n-k} : \exists\, z \in \mathbb{R}^k \text{ such that } \begin{pmatrix} z \\ u \end{pmatrix} \in K \right\}.$$

Therefore, $w \in K_k$ if and only if

$$\begin{pmatrix} z \\ w \end{pmatrix} \in K \subseteq \{x \in \mathbb{R}^n : (F_1 \quad F_2)\, x = g\}$$

for some $z \in \mathbb{R}^k$, which happens if and only if

$$\begin{pmatrix} F_1^{-1} g - F_1^{-1} F_2 w \\ w \end{pmatrix} \in K.$$

Note that for any vertex u^* of K_k, there exists $z^* \in \mathbb{R}^k$ such that $\binom{z^*}{u^*}$ is a vertex of K. Therefore, since K has vertex complexity at most $4n^2\varphi$, K_k has vertex complexity at most $4n^2\varphi$. This implies that K_k has facet complexity at most $\varphi' = 3n^2(4n^2\varphi)$. Apply APP-ELL to K_k with $R = 2^{4n^2\varphi'}$ and $\epsilon = 2^{-48n^5\varphi'}$, using the following modified approximate separation oracle for K_k:

Input: $w \in \mathbb{R}^{n-k}$.
Output: either

1. assert $y \in \bar{K}$, where

$$y = \begin{pmatrix} F_1^{-1} g - F_1^{-1} F_2 w \\ w \end{pmatrix}$$

2. find $\bar{c} \in \mathbb{Q}^{n-k}$ such that $\|\bar{c}\|_\infty = 1$ and $\bar{c}^\top w > \bar{c}^\top u$ for all $u \in K_k$.

1. Apply S-APP-SEP oracle for K on
$$y = \begin{pmatrix} F_1^{-1}g - F_1^{-1}F_2 w \\ w \end{pmatrix}$$
2. If the S-APP-SEP oracle asserts $y \in \bar{K}$, then assert $y \in \bar{K}$. Stop.
3. Otherwise, the S-APP-SEP oracle returns $c \in \mathbb{Q}^n$ such that $c^\top y > c^\top x$ for all $x \in K$. Let $c^1 \in \mathbb{Q}^k$ and $c^2 \in \mathbb{Q}^{n-k}$ such that
$$c = \begin{pmatrix} c^1 \\ c^2 \end{pmatrix}$$
Therefore,
$$(c^1)^\top (F_1^{-1}g - F_1^{-1}F_2 w) + (c^2)^\top w > (c^1)^\top (F_1^{-1}g - F_1^{-1}F_2 u) + (c^2)^\top u$$
for all $u \in K^k$. Or equivalently,
$$((c^2)^\top - (c^1)^\top F_1^{-1}F_2)w > ((c^2)^\top - (c^1)^\top F_1^{-1}F_2)u$$
for all $u \in K^k$. Return
$$\bar{c} = \frac{c^2 - (F_1^{-1}F_2)^\top c^1}{\|c^2 - (F_1^{-1}F_2)^\top c^1\|_\infty}$$
as the vector representing a hyperplane that separates w and K^k. Stop.

If APP-ELL returns a vector $y \in \bar{K}$, then we have solved APP-NEMPT and we are done. Otherwise, APP-ELL returns an ellipsoid $E^k \subseteq \mathbb{R}^{n-k}$ such that $K^k \subseteq E^k$ and $\text{vol}(E^k) \leq \epsilon$. Therefore, by Lemma 2.A.4 we can find $\bar{f}^{k+1} \in \mathbb{Z}^{n-k}$ and $g^{k+1} \in \mathbb{Z}_{>0}$ such that $K^k \subseteq \{y \in \mathbb{R}^{n-k} : \bar{f}^{k+1}y = g^{k+1}\}$. Without loss of generality, let $\bar{f}_1^{k+1} \neq 0$. Let $f^{k+1} \in \mathbb{Z}^n$ such that
$$f^{k+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \bar{f}^{k+1} \end{pmatrix}.$$
It follows that $K \subseteq \{x \in \mathbb{R}^n : (f^{k+1})^\top x = g^{k+1}\}$. By the induction hypothesis,
$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^k)^\top x = g^k, (f^{k+1})^\top x = g^{k+1}\}$$
and $f^1, \dots, f^k, f^{k+1}$ are linearly independent.

When $k = n$, we have that
$$K \subseteq \{x \in \mathbb{R}^n : (f^1)^\top x = g^1, \dots, (f^n)^\top x = g^n\}.$$
Since $f^1, \dots, f^n$ are linearly independent, K must be equal to the unique vector y in

$\{x \in \mathbb{R}^n : (f^1)^\mathsf{T} x = g^1, \dots, (f^n)^\mathsf{T} x = g^n\}$, or empty. Running the S-APP-SEP oracle for K on y, we either determine that $y \in \bar{K}$ or K is empty.

By Lemma 2.A.2(b) and Lemma 2.A.4, we can find the vectors $f^1, \dots, f^n$ and the scalars $g^1, \dots, g^n$ in time polynomial in n and φ. In addition, the inputs ϵ and R defined above imply that the calls to APP-ELL above run in time polynomially bounded by n, φ, and the running time of the S-APP-SEP oracle (which is assumed to be polynomial in n and φ). Since at most n calls to APP-ELL are made, the prescribed method above solves APP-NEMPT in time polynomial in n and φ. $\square$

Note that Theorem 2.A.5 implies Theorem 2.3.9, since the facet complexity of Q_γ is polynomially bounded by n and the encoding length of $v(N) + \gamma$.

2.B Maximizing non-monotone submodular functions

Feige et al. (2007) gave a 3-approximation algorithm for maximizing nonnegative submodular functions. We provide the proof here, for the reader's convenience. In addition, we show that some simple observations prove that their algorithm works for any submodular function $f : 2^N \mapsto \mathbb{R}$ with $f(\emptyset) \geq 0$ and $f(N) \geq 0$. We use their algorithm under these weaker conditions in Corollary 2.3.7. Note that these weaker conditions still guarantee that the maximum value of the function f is nonnegative.

As usual, we assume $N = \{1, \ldots, n\}$. For a given set function $f : 2^N \mapsto \mathbb{R}$ and some $\alpha \in \mathbb{R}_{>0}$, a set $S \subseteq N$ is called a $(1+\alpha)$-*approximate local optimum* if $(1+\alpha) f(S) \geq f(S \setminus \{i\})$ for all $i \in S$, and $(1+\alpha) f(S) \geq f(S \cup \{i\})$ for all $i \notin S$.

We consider the following local search algorithm.

Algorithm 2.B.1. Local search algorithm for maximizing submodular functions.

```
Input: submodular function $f : 2^N \mapsto \mathbb{R}$ with $f(\emptyset) = 0$ and $f(N) = 0$, given by a
    value oracle; precision parameter $\epsilon > 0$.
Output: approximate solution $\bar{S}$.

1. (Initialization) $S \leftarrow \{a\}$, where $f(\{a\}) = \max_{i \in N} f(\{i\})$.
2. (Local search)
   (a) If there exists an element $i \in N \setminus S$ such that $f(S \cup \{i\}) > (1 + \frac{\epsilon}{n^2}) f(S)$, then
       $S \leftarrow S \cup \{i\}$, and go to Step 2.
   (b) If there exists an element $i \in S$ such that $f(S \setminus \{i\}) > (1 + \frac{\epsilon}{n^2}) f(S)$, then
       $S \leftarrow S \setminus \{i\}$, and go to Step 2.
3. Return $\bar{S} = \arg\max\{f(S), f(N \setminus S)\}$.
```

If Algorithm 2.B.1 terminates, the set S that it ends with is clearly a $(1+\frac{\epsilon}{n^2})$-approximate local optimum of f. In addition, under our assumption that $f(\emptyset) \geq 0$ and $f(N) \geq 0$, we observe that $f(S)$ is nonnegative. The algorithm starts with $\{a\}$, and $f(\{a\}) \geq 0$, since by submodularity,

$$n \cdot f(\{a\}) \geq \sum_{i \in N} f(\{i\}) \geq f(N) + (n-1) f(\emptyset) \geq 0.$$

At each iteration of the algorithm, the function value increases by a factor of at least $(1+\frac{\epsilon}{n^2})$, so it follows that $f(S)$ is also nonnegative.

Next, we show a nice property of nonnegative $(1+\alpha)$-approximate local optima of submodular functions.

Lemma 2.B.2. *Let $S \subseteq N$ be a $(1+\alpha)$-approximate local optimum of a submodular function f with $f(S) \geq 0$. Then, for any subset $I \subseteq N$ such that $I \subseteq S$ or $I \supseteq S$, we have $f(I) \leq (1 + n\alpha) f(S)$.*

Proof. Fix some $I \subseteq S$, and let $I = T_1 \subseteq T_2 \subseteq \cdots \subseteq T_k = S$ be a chain of sets, where $T_i \setminus T_{i-1} = \{a_i\}$ for $i = 2, \ldots, k$. By submodularity and approximate local optimality, we have that

$$f(T_i) - f(T_{i-1}) \geq f(S) - f(S \setminus \{a_i\}) \geq -\alpha f(S) \quad \text{for } i = 2, \ldots, k.$$

Summing these inequalities, we obtain $f(S) - f(I) \geq -(k-1)\alpha f(S)$. Since $f(S) \geq 0$, we have that $f(I) \leq (1 + (k-1)\alpha) f(S) \leq (1 + n\alpha) f(S)$.

The proof for $I \supseteq S$ works the same way. □

Now we are ready to analyze the performance guarantee and running time of the local search algorithm.

Theorem 2.B.3. *Algorithm 2.B.1 is a* $(3 + \frac{2\epsilon}{n})$*-approximation algorithm for maximizing submodular functions* $f : 2^N \mapsto \mathbb{R}$ *with* $f(\emptyset) \geq 0$ *and* $f(N) \geq 0$.

Proof. Suppose S^* is an optimal solution, and let a be an element of N such that $f(\{a\}) = \max_{i \in N} f(\{i\})$ (as computed as in Step 1 of the algorithm). Above, we observed that $f(\{a\}) \geq 0$. We also have that $f(S^*) \leq n f(\{a\})$, since

$$\begin{aligned} n \cdot f(\{a\}) \geq |S^*| f(\{a\}) \geq |S^*| \max_{i \in S^*} f(\{i\}) & \\ \geq \sum_{i \in S^*} f(\{i\}) \geq f(S^*) + (|S^*| - 1) f(\emptyset) \geq f(S^*). \end{aligned}$$

After each iteration, the value of the function increases by a factor of at least $(1 + \frac{\epsilon}{n^2})$, so after k iterations, $f(S) \geq (1 + \frac{\epsilon}{n^2})^k f(\{a\})$. It follows that the number of iterations is $O(\frac{1}{\epsilon} n^2 \log n)$, and the number of queries to the value oracle of f is $O(\frac{1}{\epsilon} n^3 \log n)$.

Let S be the $(1 + \frac{\epsilon}{n^2})$-approximate local optimum that the algorithm computes. By the observation above, $f(S) \geq 0$. Therefore, by Lemma 2.B.2, we have that

$$f(S \cap S^*) \leq \left(1 + \frac{\epsilon}{n}\right) f(S) \quad \text{and} \quad f(S \cup S^*) \leq \left(1 + \frac{\epsilon}{n}\right) f(S).$$

By submodularity,

$$\begin{gathered} f(S \cup S^*) + f(N \setminus S) \geq f(S^* \setminus S) + f(N), \\ f(S \cap S^*) + f(S^* \setminus S) \geq f(S^*) + f(\emptyset). \end{gathered}$$

Combining the above with the assumption that $f(\emptyset) \geq 0$ and $f(N) \geq 0$, we have that

$$\begin{aligned} \left(3 + \frac{2\epsilon}{n}\right) f(\bar{S}) &= \left(3 + \frac{2\epsilon}{n}\right) \max\{f(S), f(N \setminus S)\} \\ &\geq 2\left(1 + \frac{\epsilon}{n}\right) f(S) + f(N \setminus S) \\ &\geq f(S \cap S^*) + f(S \cup S^*) + f(N \setminus S) \\ &\geq f(S \cap S^*) + f(S^* \setminus S) + f(N) \\ &\geq f(S^*) + f(\emptyset) + f(N) \\ &\geq f(S^*). \end{aligned}$$

□

2.C Additional Examples

Example 2.C.1. Suppose we have a set of jobs $N = \{1, \dots, n\}$, each of which has a weight w_i and processing time p_i. Let $v(S)$ be the optimal value of $P2 \,|\,|\, \sum w_j C_j$ for jobs in S. This example shows that $v(S)$ is not supermodular on all subsets of N. Consider the following instance of $P2 \,|\,|\, \sum w_j C_j$ with 5 jobs:

i	1	2	3	4	5
w_i	1	5	2	4	2
p_i	1	1	5	5	4

It is straightforward to show that $v(\{3,4,5\}) = 46$, $v(\{1,3,4,5\}) = 51$, $v(\{2,3,4,5\}) = 55$, and $v(\{1,2,3,4,5\}) = 59$. Therefore, we have that

$$v(\{1,3,4,5\}) + v(\{2,3,4,5\}) = 106 > 105 = v(\{1,2,3,4,5\}) + v(\{3,4,5\}).$$

So v is not supermodular. □

Example 2.C.2. Suppose we have a set of jobs $N = \{1, \dots, n\}$, each of which has a weight w_i, processing time p_i, and release date r_i. Let $v(S)$ be the optimal value of $1 \,|\, r_j \,|\, \sum w_j C_j$ for jobs in S. This example shows that $v(S)$ is not supermodular on all subsets of N. Consider the following instance of $1 \,|\, r_j \,|\, \sum w_j C_j$ with 4 jobs. Let $w_1 = w_2 = w_3 = w_4 = 1$, and let the processing times and release dates are as follows:

i	1	2	3	4
p_i	2	3	5	2
r_i	0	2	3	5

It is straightforward to show that $v(\{1,3\}) = 10$, $v(\{1,2,3\}) = 17$, $v(\{1,3,4\}) = 20$, and $v(\{1,2,3,4\}) = 26$. Therefore,

$$v(\{1,2,3\}) - v(\{1,3\}) = 7 > 6 = v(\{1,2,3,4\}) - v(\{1,3,4\}).$$

So v is not supermodular. □

Example 2.C.3. Suppose we have a set of jobs $N = \{1, \dots, n\}$, each of which has a weight w_i, and a processing time p_i. There are precedence constraints between the jobs defined by a transitively closed, acyclic directed graph $G = (N, P)$. Job i must precede job j if $(i, j) \in P$. Let $v(S)$ be the minimum sum of weighted completion times of jobs in $S \subseteq N$.

An initial set $I \subseteq N$ is a set of jobs with the following property: if $j \in I$ and $(i, j) \in P$, then $i \in I$. One can show that if I and J are initial sets of N, then $I \cup J$ and $I \cap J$ are initial sets of N.

This example shows that $v(S)$ is not supermodular on initial sets of N. Consider the following instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ with 5 jobs. Let $w_1 = w_2 = \cdots = w_5 = 1$, and let the processing times be as follows:

i	1	2	3	4	5
p_i	6	1	5	1	4

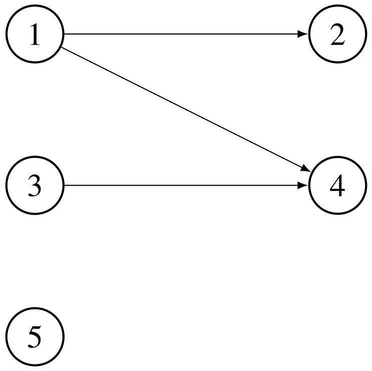

Figure 2-1: Precedence constraints for Example 2.C.3

The precedence constraints are shown in Figure 2-1. It is straightforward to show that $v(\{1,2,3,4,5\}) = 55$, $v(\{1,3,4,5\}) = 44$, $v(\{1,2,3,5\}) = 40$, and $v(\{1,3,5\}) = 28$. Therefore,

$$v(\{1,2,3,5\}) + v(\{1,3,4,5\}) = 84 > 83 = v(\{1,2,3,4,5\}) + v(\{1,3,5\})$$

So v is not supermodular. □

Example 2.C.4. In this example, we show why a "straightforward" approach to an FPTAS for the $\bar{x}$-MD problem for scheduling games does not work.

Fix some $\epsilon > 0$, and let $\epsilon' = \frac{\delta\epsilon}{1-\delta}$. Suppose $\bar{S}$ is output from an FPTAS with precision ϵ' for the two-machine scheduling problem, and S^* is an optimal solution to the two-machine scheduling problem (and the $\bar{x}$-MD problem). Then, by definition,

$$v(\bar{S}) + v(N \setminus \bar{S}) \leq (1+\epsilon')(v(S^*) + v(N \setminus S^*))$$

Suppose that we could show that

$$v(N) - v(S^*) - v(N \setminus S^*) \geq \delta v(N) \tag{2.C.1}$$

for some $\delta > 0$. Then, we have that

$$\begin{aligned}
2e(\bar{x},\bar{S}) &= v(N) - v(\bar{S}) - v(N \setminus \bar{S}) \\
&\geq v(N) - (1+\epsilon')(v(S^*) + v(N \setminus S^*)) \\
&= (1+\epsilon')(v(N) - v(S^*) - v(N \setminus S^*)) - \epsilon' v(N) \\
&\geq (1+\epsilon')(v(N) - v(S^*) - v(N \setminus S^*)) - \frac{\epsilon'}{\delta}(v(N) - v(S^*) - v(N \setminus S^*)) \\
&= \left(1 - \epsilon'\left(\frac{1}{\delta} - 1\right)\right) \cdot 2e(\bar{x}, S^*) \\
&= (1-\epsilon) \cdot 2e(\bar{x}, S^*).
\end{aligned}$$

Therefore, $\bar{S}$ is a $(1-\epsilon)$-approximation for the $\bar{x}$-MD problem.

However, there is a caveat: in order to show that this approach yields an FPTAS for the $\bar{x}$-MD problem, we need to show that $\frac{1-\delta}{\delta}$ is a polynomial in the input size and $1/\epsilon$. Unfortunately, this is not always the case. Consider the following instance:

$$\begin{aligned} w_1 = 2^{n+1}, &\quad w_2 = \cdots = w_n = 1, \\ p_1 = 2^n, &\quad p_2 = \cdots = p_n = 1. \end{aligned}$$

Note that the input size for this instance is polynomial in n. The optimal sequence for the single-machine problem is $(1, \ldots, n)$, and so

$$v(N) = (2^{n+1})(2^n) + (n-1)2^n + \frac{n(n-1)}{2}.$$

An optimal schedule for the two-machine problem is to put job 1 on one machine, and jobs $2, \ldots, n$ on the other, so

$$v(S^*) + v(N \setminus S^*) = (2^{n+1})(2^n) + \frac{n(n-1)}{2}.$$

Therefore,

$$\begin{aligned} \frac{v(N)}{v(N) - v(S^*) - v(N \setminus S^*)} &= \frac{(2^{n+1})(2^n) + (n-1)2^n + \frac{n(n-1)}{2}}{(n-1)2^n} \\ &= \frac{2^{n+1}}{n-1} + 1 + \frac{n}{2^{n+1}}. \end{aligned}$$

So any δ that satisfies (2.C.1) must satisfy

$$\frac{1-\delta}{\delta} \geq \frac{2^{n+1}}{n-1} + \frac{n}{2^{n+1}},$$

which is not polynomial in the input size. $\square$

Chapter 3

Scheduling Meets Random Graphs: A Probabilistic Analysis of Precedence-Constrained Scheduling

3.1 Introduction

We consider the following classic scheduling problem. We have a set of jobs $N = \{1, \dots, n\}$ that needs to be scheduled nonpreemptively on a single machine, which can process at most one job at a time. Each job $i \in N$ has a processing time $p_i \in \mathbb{R}_{\geq 0}$ and weight $w_i \in \mathbb{R}_{\geq 0}$. Precedence constraints are represented by an acyclic, transitively closed directed graph $G = (N, A)$: if $(i, j) \in A$, then job i must be processed before job j. The objective is to schedule these jobs in a way that respects the precedence constraints and minimizes the sum of weighted completion times. In the notation of Graham et al. (1979), this problem is denoted as $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. Lawler (1978) and Lenstra and Rinnooy Kan (1978) showed that this problem is strongly NP-hard.

Researchers have devoted much attention to the approximability of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$, taking inspiration from its mathematical programming formulations, structural properties, and relationships with other combinatorial optimization problems. For example, linear programming relaxations were pivotal in the design of the first constant-factor approximation algorithms for this problem. One class of linear programming relaxations that has been studied extensively is based on variables describing the underlying linear ordering polytope of the problem. Schulz (1996) showed that a linear programming relaxation in these variables, originally proposed by Potts (1980), can be used to obtain a 2-approximation algorithm. Chudak and Hochbaum (1999) proved that a further relaxation of Potts's linear relaxation also yields a 2-approximation algorithm. Furthermore, they showed that their linear programming relaxation can be solved using one minimum-cut computation, making their algorithm combinatorial.

Another stream of literature has focused on structural properties of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ as a means to approximation. Sidney (1975) introduced a way to decompose an instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ into smaller, so-called *stiff* instances, so that the concatenation of optimal schedules for the smaller parts yields an optimal schedule for the entire instance.

Sidney's decomposition theory has been crucial in the design and analysis of approximation algorithms for this scheduling problem. Independently, Chekuri and Motwani (1999) and Margot et al. (2003) showed that any feasible schedule consistent with Sidney's decomposition is a 2-approximate schedule. Goemans and Williamson (2000) provided a nice geometric proof of this result, using the two-dimensional Gantt charts of Eastman et al. (1964). Queyranne and Schulz (2006) showed that for the same problem, but on *identical parallel machines*, stiff instances can be approximated within a factor of 3 (the currently best known approximation algorithm for arbitrary instances of this problem is also due to Queyranne and Schulz (2006), and has a performance guarantee of 4).

Recently, it was shown that the scheduling problem $1 \mid \text{prec} \mid \sum w_j C_j$ is in fact a special case of the minimum vertex cover problem (Correa and Schulz 2005; Ambühl and Mastrolilli 2006). Although these two problems seem quite different at a first glance, they share some striking similarities. For instance, both problems have several 2-approximation algorithms, and both problems have been strongly conjectured to be inapproximable within a factor of $(2 - \epsilon)$ for any $\epsilon > 0$ (Hochbaum 1983; Schuurman and Woeginger 1999). In addition, like Sidney's decomposition theory for $1 \mid \text{prec} \mid \sum w_j C_j$, the vertex cover problem admits a so-called *persistency property*: a structural characterization of optimal vertex covers that implies a straightforward 2-approximation algorithm (Nemhauser and Trotter 1975; Hochbaum 1983). Ambühl et al. (2006) and Ambühl et al. (2007a) exploit this relationship between the two problems to obtain approximation algorithms for instances of $1 \mid \text{prec} \mid \sum w_j C_j$ with special types of precedence constraints. This relationship has also yielded some inapproximability results for $1 \mid \text{prec} \mid \sum w_j C_j$: Ambühl et al. (2007b) showed that the approximability of $1 \mid \text{prec} \mid \sum w_j C_j$ (without fixed costs implied by the precedence constraints) is equivalent to the approximability of the minimum vertex cover problem, which is inapproximable within a factor of $2 - \epsilon$ for any $\epsilon > 0$, assuming the Unique Games Conjecture holds (Khot and Regev 2003).

3.1.1 Contributions of this work

In this work, we focus on *0-1 bipartite instances*, in which precedence constraints are represented by a bipartite partial order, with minimal jobs having unit processing time and zero weight, and maximal jobs having zero processing time and unit weight. These instances are quite appealing because of their simple combinatorial structure. Moreover, it turns out that these simple instances capture the inherent difficulty of the problem. Chekuri and Motwani (1999) used a class of 0-1 bipartite instances to show that Potts's linear programming relaxation has an integrality gap of 2. Woeginger (2003) showed that a ρ-approximation algorithm for 0-1 bipartite instances of $1 \mid \text{prec} \mid \sum w_j C_j$ implies a $(\rho + \epsilon)$-approximation algorithm for *arbitrary* instances of $1 \mid \text{prec} \mid \sum w_j C_j$: the approximability behavior of 0-1 bipartite instances and arbitrary instances are virtually identical. As a result, in order to obtain better approximation algorithms for $1 \mid \text{prec} \mid \sum w_j C_j$, it suffices to consider the family of 0-1 bipartite instances.

In an effort to better understand the properties of this important class of instances, we study these instances of the precedence-constrained scheduling problem from a probabilistic lens. One appealing feature of 0-1 bipartite instances is that they are completely defined by their precedence constraints. Since precedence relations in bipartite partial orders are independent, we can apply the Erdös-Rényi model often used in random graph theory to obtain random models of 0-1 bipartite instances (for an extensive treatment of random graph theory, see the textbook by Bollobás (2001)). Our analysis of these random 0-1 bipartite instances yields several “almost all”-type results that lead to some intriguing insights.

- After introducing the necessary background in Section 3.2, we show that almost all 0-1 bipartite instances are *non-Sidney-decomposable* in Section 3.3. Despite the influence of Sidney’s decomposition technique on the design of approximation algorithms for this problem, our result implies that for 0-1 bipartite instances, the decomposition technique of Sidney will almost never be of any help. This result also strengthens the connection between $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ and the vertex cover problem: Pulleyblank (1979) analogously showed that for almost all graphs under the random graph model in which an edge appears independently with constant probability, the associated vertex cover problem cannot be decomposed using the persistency property. Finally, together with the work of Chekuri and Motwani (1999), Margot et al. (2003), and Goemans and Williamson (2000), this result also implies that for almost all 0-1 bipartite instances, any feasible schedule is a 2-approximation. However, as we will see below, it turns out something much stronger can be shown.

- In Section 3.4, by studying the geometric properties of two-dimensional Gantt charts for random 0-1 bipartite instances, we show that for almost all 0-1 bipartite instances, *all feasible schedules are arbitrarily close to optimal*. This result offers an interesting paradox: despite the apparent difficulty of obtaining an approximation algorithm with a performance guarantee better than 2, for any given $\epsilon > 0$, *any* feasible schedule is a $(1+\epsilon)$-approximation for almost all 0-1 bipartite instances, when the number of jobs is sufficiently large.

- Finally, for almost all 0-1 bipartite instances, we give a *lower bound on the integrality gap of various linear programming relaxations of* $1 \,|\, prec \,|\, \sum w_j C_j$. For the random models of 0-1 bipartite instances that we study, this lower bound approaches 2 as the precedence constraints become sparser in expectation. This result confirms that the class of 0-1 bipartite instances is especially impervious to approaches for better approximation algorithms based on existing linear programming relaxations.

3.2 Definitions and background

3.2.1 Sidney’s decomposition theory

Sidney (1975) introduced a very useful characterization of optimal schedules to $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. We define

$$\rho(S) = \begin{cases} \sum_{j \in S} w_j / \sum_{j \in S} p_j & \text{for any subset of jobs } S \subseteq N \text{ such that } \sum_{j \in S} p_j > 0, \\ +\infty & \text{otherwise.} \end{cases}$$

A set of jobs $I \subseteq N$ is called *initial* if $j \in I$ and $(i, j) \in A$ imply $i \in I$. An initial set I^* is said to be *ρ-maximal* if $I^* = \arg\max\{\rho(I) : I \text{ is a nonempty initial set}\}$.

Sidney showed that there exists an optimal schedule in which all jobs in a ρ-maximal initial set S^* are scheduled before those in $N \setminus S^*$. By recursively applying this result, we naturally obtain a partition of jobs $(S_1, \dots, S_k)$ with $\rho(S_1) \geq \cdots \geq \rho(S_k)$. This partition is called a *Sidney decomposition*. Sidney's decomposition theory can be seen as a generalization of Smith's (1956) rule for the problem without precedence constraints. Lawler (1978) showed how to compute Sidney's decomposition efficiently using minimum cut computations. An instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ is *non-Sidney-decomposable* if the only ρ-maximal initial set is N; otherwise the instance is called *Sidney-decomposable*. An instance is called *stiff* if $\rho(N) \geq \rho(I)$ for all nonempty initial sets I; note that stiffness is a necessary condition for an instance to be non-Sidney-decomposable. Independently, Chekuri and Motwani (1999) and Margot et al. (2003) showed that for stiff instances, any feasible schedule is a 2-approximation.

3.2.2 IP formulations, LP relaxations and the minimum vertex cover problem

Several linear programming relaxations have been studied for $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. Potts (1980) proposed an integer programming formulation based on linear ordering variables. Define the decision variables δ as follows: for all $i, j \in N$, δ_{ij} is equal to 1 if job i is processed before job j, and 0 otherwise. Then $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ can be formulated as

$$
\begin{array}{llll}
\text{[P]} \quad \text{minimize} & \displaystyle\sum_{j \in N} p_j w_j + \sum_{i,j \in N} p_i w_j \delta_{ij} & & (3.2.1a) \\
\text{subject to} & \delta_{ij} + \delta_{ji} = 1 & \text{for all } i, j \in N : i \neq j, & (3.2.1b) \\
& \delta_{ij} + \delta_{jk} + \delta_{ki} \leq 2 & \text{for all } i, j, k \in N : i \neq j \neq k \neq i, & (3.2.1c) \\
& \delta_{ij} = 1 & \text{for all } (i, j) \in A, & (3.2.1d) \\
& \delta_{ij} \in \{0, 1\} & \text{for all } i, j \in N : i \neq j. & (3.2.1e)
\end{array}
$$

The inequalities (3.2.1b) model the fact that either job i must be scheduled before job j, or vice versa. The inequalities (3.2.1c) capture the transitivity property: if job i is scheduled before job j, and job j is scheduled before job k, then job i must be scheduled before job k. The inequalities (3.2.1d) enforce the precedence constraints. It is straightforward to check that [P] is a correct formulation of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$.

Chudak and Hochbaum (1999) suggested relaxing the formulation [P] by including only the transitivity constraints (3.2.1c) in which two jobs are already precedence constrained:

$$
\begin{array}{ll}
\text{[CH]} \quad \text{minimize} & (3.2.1a) \\
\text{subject to} & (3.2.1b), (3.2.1d), (3.2.1e), \\
& \delta_{jk} + \delta_{ki} \leq 1 \quad \text{for all } (i, j) \in A, k \in N : i \neq j \neq k \neq i.
\end{array}
$$

Correa and Schulz (2005) proposed the following relaxation of [P]:

$$
\text{[CS]} \quad \text{minimize} \quad \sum_{j \in N} p_j w_j + \sum_{(i,j) \in A} p_i w_j + \sum_{i,j \in N : i \parallel j} p_i w_j \delta_{ij}
$$

$$\begin{array}{lll}\text{subject to} & \delta_{ij} + \delta_{ji} \geq 1 & \text{for all } i, j \in N : i \parallel j, \\ & \delta_{ik} + \delta_{kj} \geq 1 & \text{for all } (i, j) \in A, k \in N : i \parallel k, k \parallel j, \\ & \delta_{il} + \delta_{kj} \geq 1 & \text{for all } (i, j), (k, l) \in A : i \parallel l, j \parallel k, \\ & \delta_{ij} \in \{0, 1\} & \text{for all } i, j \in N : i \parallel j,\end{array}$$

where $i||j$ means that $i \neq j$ and neither $(i, j) \in A$ nor $(j, i) \in A$. We denote the LP relaxations of [P], [CH] and [CS], obtained by replacing the binary constraints with the nonnegativity constraints $\delta_{ij} \geq 0$ for all $i, j \in N$, by [P-LP], [CH-LP] and [CS-LP], respectively.

The formulation [CS] represents a minimum vertex cover problem, on what is called the *graph of incomparable pairs of* (N, A). This graph has been studied extensively in the dimension theory of partially ordered sets (for example, see Trotter 1992). Correa and Schulz (2005) showed that the optimal solutions to [CS] and [CH] coincide, and Ambühl and Mastrolilli (2006) showed that any solution to [CH] can be transformed in polynomial time into a feasible solution to [P] with no increase in objective value (analogous results were shown for [P-LP], [CH-LP], and [CS-LP] as well). As a result, the scheduling problem $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ is in fact a special case of the minimum vertex cover problem.

In terms of approximability, one important difference between the scheduling problem $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ and its corresponding vertex cover problem is the fixed cost $\sum_{j \in N} p_j w_j + \sum_{(i,j) \in A} p_i w_j$. In particular, because of this fixed cost, it is not immediately clear whether the approximability behavior for the vertex cover problem translates to the scheduling problem. For example, for stiff instances, all feasible schedules are no longer 2-approximations for the problem without fixed costs (see Example 3.A.1). It has been shown that approximating $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ without the fixed costs is as hard as approximating the minimum vertex cover problem (Ambühl et al. 2007b).

One of the striking similarities between the scheduling problem and the vertex cover problem is the existence of a structural decomposition for optimal solutions. Nemhauser and Trotter (1975) considered the following linear programming relaxation of the minimum vertex cover problem on an undirected graph $G = (V, E)$:

$$\begin{array}{llll}\text{minimize} & \sum_{i \in V} w_i x_i & & (3.2.2a) \\ \text{subject to} & x_i + x_j \geq 1 & \text{for all } \{i, j\} \in E, & (3.2.2b) \\ & x_i \geq 0 & \text{for all } i \in V. & (3.2.2c)\end{array}$$

They showed that the formulation (3.2.2a)-(3.2.2c) is half-integral, and that its optimal solutions satisfy a so-called *persistency property*: if x is an optimal solution to (3.2.2a)-(3.2.2c), then there exists an optimal vertex cover that contains $\{i \in V : x_i = 1\}$ and does not contain $\{i \in V : x_i = 0\}$. Hochbaum (1983) showed that if the "all 1/2" solution—$x_i = 1/2$ for all $i \in V$—is an optimal solution to (3.2.2a)-(3.2.2c), then any feasible vertex cover is a 2-approximation. These features of the persistency property are very similar to those of Sidney's decomposition theory. A result of Correa and Schulz (2005) implies that if the "all $1/2$" solution is an optimal solution to [P-LP], [CH-LP], or [CS-LP], then the instance must be non-Sidney-decomposable. However, it turns out that the converse is not true, as shown in Example 3.A.2.

Another linear programming relaxation that has been often studied in the literature uses completion time variables (Queyranne and Wang 1991; Hall et al. 1997).

$$\text{[QW-LP]} \quad \text{minimize} \quad \sum_{j \in N} w_j C_j \tag{3.2.3a}$$

$$\text{subject to} \quad \sum_{j \in S} p_j C_j \geq \frac{1}{2} \sum_{j \in S} p_j^2 + \frac{1}{2}\Big(\sum_{j \in S} p_j\Big)^2 \quad \text{for all } S \subseteq N, \tag{3.2.3b}$$

$$C_j - C_i \geq p_j \quad \text{for all } (i,j) \in A. \tag{3.2.3c}$$

The inequalities (3.2.3b) are known as the *parallel inequalities*; they suffice to describe the convex hull of feasible completion time vectors on a single machine without precedence constraints (Wolsey 1985; Queyranne 1993). The inequalities (3.2.3c) model the precedence constraints.

3.2.3 0-1 bipartite instances

A *0-1 bipartite instance* $B = (N_1, N_2, A)$ of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ consists of a set of jobs $N_1 \cup N_2 = \{1, \ldots, n\}$ with $N_1 \cap N_2 = \emptyset$, and precedence constraints that take the form of a directed bipartite graph $(N_1 \cup N_2, A)$, where $(i, j) \in A$ implies $i \in N_1$ and $j \in N_2$. The jobs in N_1 have unit processing time and zero weight, and the jobs in N_2 have zero processing time and unit weight. Note that a 0-1 bipartite instance is entirely defined by the directed bipartite graph that defines its precedence constraints.

Although 0-1 bipartite instances seem rather simple, as it turns out, they are as hard to approximate as arbitrary instances of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. Woeginger (2003) showed that an arbitrary instance $I = (N, A, (p_i)_{i \in N}, (w_i)_{i \in N})$ with $p_i \in \mathbb{Z}_{>0}$ and $w_i \in \mathbb{Z}_{>0}$ for all $i \in N$, can be converted to a 0-1 bipartite instance $B_I = (N_1, N_2, \bar{A})$ with equal optimal objective value. Woeginger also showed that any feasible schedule for I can be mapped to a feasible schedule for B_I, and vice versa. The transformation works as follows. First, we construct the job sets N_1 and N_2. For each job $i \in N$ from the instance I, we create two sets of jobs, i' and i''. The set i' consists of p_i jobs with unit processing time and zero weight, and the set i'' consists of w_i jobs with zero processing time and unit weight. The set N_1 consists of all jobs in i' for all $i \in N$, and the set N_2 consists of all jobs in i'' for all $i \in N$. Next, we construct the precedence constraints $(N_1 \cup N_2, \bar{A})$. For every job $i \in N$, every job in i' precedes every job in i''. In addition, for all $(i, j) \in A$, every job in i' precedes every job in j''. Note that for every $(i, j) \in \bar{A}$, we have that $i \in N_1$ and $j \in N_2$, so this transformation does create a 0-1 bipartite instance.

Note that this transformation may not be polynomial if weights and processing times are not polynomially bounded. However, by using appropriate rounding techniques, Woeginger (2003) showed that if there is a ρ-approximation algorithm for 0-1 bipartite instances of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$, then there exists a $(\rho + \epsilon)$-approximation algorithm for arbitrary instances of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$, for any $\epsilon > 0$.

Models for random 0-1 bipartite instances

One of the nice properties of 0-1 bipartite instances is that they are completely defined by their precedence constraints. Since precedence relations in bipartite precedence constraints are independent of each other, we can apply the model of Erdös and Rényi (1959) from random graph theory to define the following model of random 0-1 bipartite instances. For any $n_1, n_2 \in \mathbb{Z}_{>0}$ and $q \in (0,1)$, we define $\mathcal{B}(n_1, n_2; q)$ as the probability space of 0-1 bipartite instances with

- jobs $N_1 \cup N_2$ where $N_1 = \{1, \dots, n_1\}$ and $N_2 = \{n_1 + 1, \dots, n_1 + n_2\}$, such that
  - for every $j \in N_1$, $p_j = 1$ and $w_j = 0$,
  - for every $j \in N_2$, $p_j = 0$ and $w_j = 1$,
- precedence constraints that take the form of a directed bipartite graph $G = (N_1 \cup N_2, A)$ where $(i, j) \in A$ implies $i \in N_1$ and $j \in N_2$.

Each edge $(i, j) \in N_1 \times N_2$ appears in A independently and with probability q. Note that in any instance of $\mathcal{B}(n_1, n_2; q)$, $|N_1| = n_1$ and $|N_2| = n_2$. In addition, labels in N_1 and labels in N_2 are fixed for all instances in $\mathcal{B}(n_1, n_2; q)$: jobs in N_1 are always labeled $1, \dots, n_1$, and jobs in N_2 are always labeled $n_1 + 1, \dots, n_1 + n_2$.

We can generalize $\mathcal{B}(n_1, n_2; q)$ so that the jobs in N_1 and N_2 are not fixed. For any $n \in \mathbb{Z}_{>0}$ and $q \in (0,1)$, we define $\mathcal{B}(n; q)$ as the probability space of 0-1 bipartite instances with

- jobs $N = \{1, \dots, n\}$ partitioned into disjoint sets N_1 and N_2 such that
  - for every $j \in N_1$, $p_j = 1$ and $w_j = 0$,
  - for every $j \in N_2$, $p_j = 0$ and $w_j = 1$,
- precedence constraints that take the form of a directed bipartite graph $G = (N_1 \cup N_2, A)$ where $(i, j) \in A$ implies $i \in N_1$ and $j \in N_2$.

Each job $i \in N = \{1, \dots, n\}$ is assigned to N_1 with probability $1/2$, and assigned to N_2 with probability $1/2$. Then, in the resulting bipartition, an arc $(i, j) \in N_1 \times N_2$ appears in A with probability q. Note that $\mathcal{B}(n; 1/2)$ is the uniform distribution over all labeled 0-1 bipartite instances.

3.3 Sidney-decomposability and 0-1 bipartite instances

In this section, we show that almost all 0-1 bipartite instances are non-Sidney-decomposable. We begin by giving the following characterization of Sidney-decomposability for 0-1 bipartite instances. For any directed graph (N, A) and any subset of vertices $X \subseteq N$, we define $\Gamma(X) = \{i \in N \setminus X : (i, j) \in A \text{ or } (j, i) \in A \text{ for some } j \in X\}$; in words, $\Gamma(X)$ is the set of neighbors of X.

Theorem 3.3.1. *A 0-1 bipartite instance of* $1\,|prec|\,\sum w_j C_j$ *with* $|N_1| = n_1$ *and* $|N_2| = n_2$ *is Sidney-decomposable if and only if one of the following conditions hold:*
(SD1) $n_1 = 0$.
(SD2) $n_2 = 0$.

(SD3) *There exists a nonempty subset* $Y \subseteq N_2$ *such that* (i) $Y \neq N_2$ and $n_2|\Gamma(Y)| \leq n_1|Y|$, or (ii) $Y = N_2$ and $|\Gamma(N_2)| \leq n_1 - 1$.

Proof. First, note that a 0-1 bipartite instance with $n_1 = 0$ or $n_2 = 0$ is Sidney-decomposable, since any nonempty subset of jobs I is initial and satisfies $\rho(I) = \rho(N)$.

Now suppose a 0-1 bipartite instance with $n_1 > 0$ and $n_2 > 0$ is Sidney-decomposable. By definition, this occurs if and only if

$$\text{there exists a } \rho\text{-maximal initial set } I \neq N \text{ such that } \rho(I) \geq n_2/n_1. \tag{3.3.1}$$

Recall that by definition, a ρ-maximal initial set is nonempty. Suppose (3.3.1) is satisfied with an initial set I such that $I \not\subseteq N_1$. Since I is ρ-maximal, $I = \Gamma(Y) \cup Y$ for some $Y \subseteq N_2$ such that $Y \neq \emptyset$. We consider the following cases.

- If $Y \neq N_2$, then (3.3.1) holds if and only if $|Y|/|\Gamma(Y)| \geq n_2/n_1$.
- Otherwise, we have $Y = N_2$. In this case, (3.3.1) holds if and only if $|\Gamma(N_2)| \leq n_1 - 1$.

Note that (3.3.1) cannot be satisfied if $I \subseteq N_1$, since in this case, $\rho(I) = 0 < n_2/n_1 = \rho(N)$. □

Note that when $n_1 = n_2$, (SD3) is very similar to Hall's (1935) characterization of perfect matchings in a bipartite graph. We give an analogous characterization of Sidney-decomposable 0-1 bipartite instances that considers subsets of N_1 instead.

Theorem 3.3.2. *The condition* (SD3) *in Theorem 3.3.1 holds if and only if*

(SD3′) *There exists a nonempty subset* $X \subseteq N_1$ *such that* (i) $X \neq N_1$ and $n_1|\Gamma(X)| \leq n_2|X|$, or (ii) $X = N_1$ and $|\Gamma(N_1)| \leq n_2 - 1$.

Proof. We prove that (SD3′) implies (SD3), by contradiction. Suppose that for all $Y \subseteq N_2$ such that $Y \neq \emptyset, N_2$, we have $n_1|Y| < n_2|\Gamma(Y)|$, and that $|\Gamma(N_2)| = n_1$. Take some arbitrary $X \subseteq N_1$ such that $X \neq \emptyset, N_1$. Let $\bar{Y} = N_2 \setminus \Gamma(X)$. Note that this implies that $\Gamma(\bar{Y}) \subseteq N_1 \setminus X$. Therefore, $|\bar{Y}| = n_2 - |\Gamma(X)|$, and $|\Gamma(\bar{Y})| \leq n_1 - |X|$. We consider the following cases:

- $\bar{Y} \neq \emptyset, N_2$. In this case, we have that $n_1(n_2 - |\Gamma(X)|) < n_2(n_1 - |X|)$, or equivalently, $n_1|\Gamma(X)| > n_2|X|$.
- $\bar{Y} = \emptyset$. In this case, we have that $\Gamma(X) = N_2$. Since $X \neq N_1$, it follows that $n_1|\Gamma(X)| = n_1 n_2 > n_2|X|$.
- $\bar{Y} = N_2$. This is impossible, since in this case, $\Gamma(X) = \emptyset$, which implies $\Gamma(N_2) \neq N_1$, or $|\Gamma(N_2)| \neq n_1$.

Putting this all together implies that for all $X \subseteq N_1$ such that $X \neq \emptyset, N_1$, we have $n_1|\Gamma(X)| > n_2|X|$. In addition, since $|\Gamma(N_2)| = n_1$, we have that $|\Gamma(N_1)| = n_2$. Therefore, we have a contradiction.

Showing the reverse direction works in a similar manner. □

3.3.1 Almost all 0-1 bipartite instances are non-Sidney-decomposable

Using the characterization of Sidney-decomposability in Theorem 3.3.2, we can show that almost all 0-1 bipartite instances are non-Sidney-decomposable, under the probability models $\mathcal{B}(n_1, n_2, ; q)$ and $\mathcal{B}(n; q)$. Before proceeding, we establish the following identity.

Lemma 3.3.3. *For any $a \in (0, 1]$ and $s \in \mathbb{Z}_{>0}$ such that $as \in \mathbb{Z}_{>0}$ and $k = 1, \dots, s$,*

$$\binom{as}{\lfloor ak \rfloor} \leq \binom{s}{k}.$$

Proof. We have for any $x = 1, \dots, n$,

$$\binom{n}{x} \geq \binom{n-1}{x-1} \tag{3.3.2}$$

and

$$\binom{n}{x} \geq \binom{n-1}{x}. \tag{3.3.3}$$

By repeatedly applying the identity (3.3.2), we obtain

$$\binom{s}{k} \geq \binom{s-k+\lfloor ak \rfloor}{\lfloor ak \rfloor}.$$

Note that

$$\begin{aligned} s-k+\lfloor ak \rfloor &= s-k+\lfloor as - a(s-k) \rfloor \\ &\geq s-k+\lfloor as-(s-k) \rfloor \\ &= s-k+as-(s-k) \\ &= as. \end{aligned}$$

This fact, in combination with repeated application of (3.3.3), proves the claim. □

First, we consider the model of random 0-1 bipartite instances $\mathcal{B}(n_1, n_2; q)$.

Theorem 3.3.4. *For any fixed $c, d \in \mathbb{Z}_{>0}$ and $q \in (0, 1)$,*

$$\lim_{t \to \infty} \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \textit{ is non-Sidney-decomposable}\big) = 1.$$

Proof. Let $B = (N_1, N_2, A)$ be a random 0-1 bipartite instance from $\mathcal{B}(ct, dt; q)$. By Theorem 3.3.1, an instance in $\mathcal{B}(ct, dt; q)$ is Sidney-decomposable if and only if (SD3) holds. We show that the probability that (SD3) holds goes to zero as t goes to infinity.

Any bipartite graph $(N_1 \cup N_2, A)$ with a subset Y of N_2 of size k such that $|\Gamma(Y)| \leq ck/d$ can be constructed as follows. Choose a subset Y of N_2 of size k, and a subset X of

N_1 of size $\lfloor ck/d \rfloor$, and then forbid all edges between Y and $N_1 \setminus X$. Therefore,

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3)}\big) \\
&\quad \le \sum_{k=1}^{dt-1} \mathbb{P}\big(\exists Y \subseteq N_2 : |Y| = k, |\Gamma(Y)| \le \lfloor ck/d \rfloor\big) + \mathbb{P}\big(|\Gamma(N_2)| \le ct - 1\big) \\
&\quad \le \sum_{k=1}^{dt-1} \binom{dt}{k}\binom{ct}{\lfloor ck/d \rfloor}(1-q)^{k(ct - \lfloor ck/d \rfloor)} + ct(1-q)^{dt}. \qquad (3.3.4)
\end{aligned}$$

Note that we can bound the probability that $B \in \mathcal{B}(ct, dt; q)$ satisfies (SD3′) in a similar manner:

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3}')\big) \\
&\qquad\qquad \le \sum_{k=1}^{ct-1} \binom{ct}{k}\binom{dt}{\lfloor dk/c \rfloor}(1-q)^{k(dt - \lfloor dk/c \rfloor)} + dt(1-q)^{ct}. \qquad (3.3.5)
\end{aligned}$$

When $c < d$, then let $a = c/d$ and $s = dt$, and consider the bound given in (3.3.4). When $c \ge d$, let $a = d/c$ and $s = ct$, and consider the bound given in (3.3.5). Since (SD3) and (SD3′) are equivalent,

$$\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3)}\big) \le \sum_{k=1}^{s-1} \binom{s}{k}\binom{as}{\lfloor ak \rfloor}(1-q)^{k(as - \lfloor ak \rfloor)} + as(1-q)^{s}.$$

So, without loss of generality, for the remainder of this proof, we let $s = ct$, and assume $a = d/c \le 1$.

Define

$$F_k = \binom{s}{k}^2 (1-q)^{ak(s-k)} \qquad k = 1, \dots, s-1.$$

Note that $F_k = F_{s-k}$ for $k = 1, \dots, \lfloor (s-1)/2 \rfloor$. By Lemma 3.3.3, we have

$$\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3}')\big) \le \sum_{k=1}^{s-1} F_k + as(1-q)^{s}.$$

Let $r = (1-q)^{-1}$, and fix some scalar $M_{a,r}$ such that $a(s-3) - 2\log_r((s-1)/2)$ for all $s \ge M_{a,r}$ (clearly such an $M_{a,r}$ exists since the logarithm grows slower than the linear function). Note that $r > 1$.

We would like to show that $F_k \ge F_{k+1}$ for $k = 1, \dots, \lfloor (s-1)/2 \rfloor$ and $s \ge M_{a,r}$, or equivalently,

$$2\log_r \frac{s-k}{k+1} \le a(s - 2k - 1) \quad \text{for } k = 1, \dots, \lfloor (s-1)/2 \rfloor \text{ and } s \ge M_{a,r}. \qquad (3.3.6)$$

Define

$$\Delta(x) = a(s - 2x - 1) - 2\log_r(s - x) + 2\log_r(x + 1).$$

Taking derivatives, we obtain

$$\begin{aligned} \frac{\partial \Delta}{\partial x} &= -2a + \frac{2}{\log r}\left(\frac{1}{s-x} + \frac{1}{x+1}\right), \\ \frac{\partial^2 \Delta}{\partial x^2} &= \frac{2}{\log r}\left(\frac{1}{(s-x)^2} - \frac{1}{(x+1)^2}\right). \end{aligned}$$

Note that for $x \in [1, (s-1)/2]$, we have that $\partial^2\Delta/\partial x^2 \leq 0$, so $\Delta(x)$ is concave on $[1, s/2]$. Evaluating $\Delta(x)$ at $x = 1$ and $x = s/2$ yields

$$\Delta(1) = a(s-3) - 2\log_r((s-1)/2) \qquad \Delta((s-1)/2) = 0.$$

We have that $\Delta(1) \geq 0$ for all $s \geq M_{a,r}$ by construction. Since $\Delta(x)$ is concave on $[1, (s-1)/2]$, it follows that when $s \geq M_{a,r}$, $\Delta(x) \geq 0$ for all $x \in [1, (s-1)/2]$, which establishes the identity (3.3.6). Therefore, when $s \geq M_{a,r}$, $F_k \geq F_{k+1}$ for $k = 1, \ldots, \lfloor (s-1)/2 \rfloor$.

Since $F_k = F_{s-k}$ for $k = 1, \ldots, \lfloor (s-1)/2 \rfloor$, it follows that $F_1 \geq F_k$ for $k = 1, \ldots, s-1$. As a result, when $t = s/c \geq M_{a,r}/c = M_{d/c,(1-q)^{-1}}/c$,

$$\begin{aligned} \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3)}\big) &\leq \sum_{k=1}^{s-1} F_k + as(1-q)^s \\ &\leq sF_1 + as(1-q)^s \\ &= s \cdot s(1-q)^{as-a} + as(1-q)^s \\ &= c^2t^2(1-q)^{dt-d/c} + dt(1-q)^{ct}. \end{aligned}$$

Putting this all together, we have

$$\begin{aligned} &\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ is Sidney-decomposable}\big) \\ &\qquad = \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies (SD3)}\big) \\ &\qquad \leq (1-q)^{cdt^2} + c^2t^2(1-q)^{dt-d/c} + dt(1-q)^{ct}. \end{aligned}$$

Therefore, we can conclude that $\lim_{t\to\infty} \mathbb{P}(B \in \mathcal{B}(ct, dt; q)$ is Sidney-decomposable$) = 0$. □

Note that in Theorem 3.3.4, we required that the number of jobs in N_1 and the number of jobs in N_2 "grow together" as the instance size grows. This is crucial. To illustrate this, consider the class of instances $\mathcal{B}(1, n-1; q)$: the class of instances in which N_1 consists of one job, and N_2 consists of $n-1$ jobs. In this case, an instance $B \in \mathcal{B}(1, n-1; q)$ is non-Sidney-decomposable if and only if the job in N_1 must precede all jobs in N_2. This occurs with probability q^{n-1}, which goes to zero as the instance size n grows. We can get around this if these problematic instances do not occur "too often." With some techniques from the proof of Theorem 3.3.4 and some additional work, we can show that almost all instances in $\mathcal{B}(n;q)$ are non-Sidney-decomposable for this probability model.

Theorem 3.3.5. *For any fixed $q \in (0,1)$,*

$$\lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ is non-Sidney-decomposable}\big) = 1.$$

Proof. Let $B = (N_1, N_2, A)$ be an random 0-1 bipartite instance from $\mathcal{B}(n;q)$. We show that the probability that any of the conditions (SD1)-(SD3) hold goes to zero as n approaches infinity.

First, we consider (SD1). We have that

$$\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD1)}\big) = \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ has } n_1 = 0\big) = \left(\frac{1}{2}\right)^n,$$

and so $\lim_{n\to\infty} \mathbb{P}(B \in \mathcal{B}(n;q) \text{ satisfies (SD1)}) = 0$. Similarly, for (SD2), we have that

$$\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD2)}\big) = \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ has } n_2 = 0\big) = \left(\frac{1}{2}\right)^n,$$

and therefore $\lim_{n\to\infty} \mathbb{P}(B \in \mathcal{B}(n;q) \text{ satisfies (SD2)}) = 0$.

Now we consider (SD3). Note that the conditional probabilities $B \in \mathcal{B}(n;q)$ satisfying (SD3) are symmetric, in the sense that

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3)} \,\big|\, |N_1| = s, |N_2| = n-s\big)\\
&\quad = \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3}')\,\big|\, |N_1| = n-s, |N_2| = s\big)\\
&\quad = \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3)} \,\big|\, |N_1| = n-s, |N_2| = s\big).
\end{aligned}$$

Therefore, we only need to consider instances when $s \geq n-s$. Observe that any bipartite graph $(N_1 \cup N_2, A)$ with $|N_1| = s$ and $|N_2| = n-s$ with a subset X of N_1 of size k such that $|\Gamma(X)| \leq \frac{n-s}{s}k$ can be constructed as follows. Choose a subset X of N_1 of size k, and a subset Y of N_2 of size $\lfloor \frac{n-s}{s}k \rfloor$, and forbid all edges between X and $N_2 \setminus Y$. Therefore, by conditioning on the size of N_1 and N_2, we have

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3)}\big)\\
&\quad = \sum_{s=1}^{n-1} \binom{n}{s}\left(\frac{1}{2}\right)^n \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3}') \,|\, |N_1| = s,\ |N_2| = n-s\big)\\
&\quad \leq 2 \sum_{s=\lceil n/2 \rceil}^{n-1} \binom{n}{s}\left(\frac{1}{2}\right)^n \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3}') \,|\, |N_1| = s,\ |N_2| = n-s\big)\\
&\quad \leq 2 \sum_{s=\lceil n/2 \rceil}^{n-1} \binom{n}{s}\left(\frac{1}{2}\right)^n \left(\sum_{k=1}^{s-1} \binom{s}{k}\binom{n-s}{\lfloor \frac{n-s}{s}k \rfloor}(1-q)^{k(n-s-\lfloor \frac{n-s}{s}k \rfloor)} + (n-s)(1-q)^s\right).
\end{aligned}$$

For the remainder of this proof, let $r = (1-q)^{-1}$, and $\nu(n) = 2\log_r((n-1)/2) + 3$.

Note that $r > 1$. We define

$$E_s = \binom{n}{s}\left(\frac{1}{2}\right)^n \sum_{k=1}^{s-1} \binom{s}{k}\binom{n-s}{\lfloor \frac{n-s}{s}k \rfloor}(1-q)^{k(n-s-\lfloor \frac{n-s}{s}k\rfloor)}$$

for $s = \lceil n/2 \rceil, \ldots, n$. We consider the expression E_s for two cases: (i) $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$, and (ii) $s = n - \nu(n) + 1, \ldots, n-1$.

First, let us consider E_s in the regime $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$. For all $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$ and $k = 1, \ldots, s-1$, define

$$F_{s,k} = \binom{s}{k}^2 (1-q)^{\frac{n-s}{s}k(s-k)},$$

and note that $F_{s,k} = F_{s,s-k}$. We would like to show that $F_{s,k} \geq F_{s,k+1}$ for all $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$ and $k = 1, \ldots, \lfloor (s-1)/2 \rfloor$, or equivalently,

$$2\log_r \frac{s-k}{k+1} \leq \frac{n-s}{s}(s-2k-1) \quad \text{for } k = 1, \ldots, \lfloor (s-1)/2 \rfloor \text{ and } s = \lceil n/2 \rceil, \ldots, n-\nu(n). \tag{3.3.7}$$

Define

$$\Delta(x) = \frac{n-s}{s}(s-2x-1) - 2\log_r(s-x) + 2\log_r(x+1).$$

As in the proof of Theorem 3.3.4, we can show that $\Delta(x)$ is concave on $[1, (s-1)/2]$. We have that $\Delta(1) \geq 0$ for all $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$, since

$$\begin{aligned}
\Delta(1) &= \frac{n-s}{s}(s-3) - 2\log_r(s-1) + 2\log_r 2 \\
&= n - s - \frac{3n}{s} + 3 - 2\log_r \frac{s-1}{2} \\
&\geq n - s - 3 - 2\log_r \frac{s-1}{2} && \text{(since } s \geq n/2) \\
&\geq n - s - 3 - 2\log_r \frac{n-1}{2} && \text{(since } s \leq n) \\
&\geq 0 && \text{(since } s \leq n - \nu(n)).
\end{aligned}$$

In addition, we have that $\Delta((s-1)/2) = 0$. Since $\Delta(x)$ is concave on $[1, (s-1)/2]$, it follows that when $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$, $\Delta(x) \geq 0$ for all $x \in [1, s/2]$, which establishes the identity (3.3.7). Therefore, $F_{s,k} \geq F_{s,k+1}$ for $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$ and $k = 1, \ldots, \lfloor s/2 \rfloor$.

Since $F_{s,k} = F_{s,s-k}$, it follows that $F_{s,1} \geq F_{s,k}$ for $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$ and $k = 1, \ldots, s$. By Lemma 3.3.3 (letting $a = \frac{n-s}{s}$), for all $s = \lceil n/2 \rceil, \ldots, n - \nu(n)$ and $k = 1, \ldots, s$, we have that

$$\binom{n-s}{\lfloor \frac{n-s}{s}k \rfloor} \leq \binom{s}{k}.$$

It follows that for $s = \lceil n/2 \rceil, \dots, n - \nu(n)$,

$$\begin{aligned} E_s &\le \binom{n}{s}\left(\frac{1}{2}\right)^n \sum_{k=1}^{s-1} \binom{s}{k}^2 (1-q)^{\frac{n-s}{s}k(s-k)} \\ &= \binom{n}{s}\left(\frac{1}{2}\right)^n \sum_{k=1}^{s-1} F_{s,k} \\ &\le \binom{n}{s}\left(\frac{1}{2}\right)^n sF_{s,1} \\ &= \binom{n}{s}\left(\frac{1}{2}\right)^n s^3(1-q)^{\frac{n-s}{s}(s-1)}. \end{aligned}$$

Therefore, for sufficiently large n,

$$\begin{aligned} \sum_{s=\lceil n/2 \rceil}^{n-\nu(n)} E_s &\le \sum_{s=\lceil n/2 \rceil}^{n-\nu(n)} \binom{n}{s}\left(\frac{1}{2}\right)^n s^3(1-q)^{\frac{n-s}{s}(s-1)} \\ &\le \left(\frac{1}{2}\right)^n (n-\nu(n))^3 \sum_{s=\lceil n/2 \rceil}^{n-\nu(n)} \binom{n}{s}(1-q)^{\frac{n-s}{2}} \qquad \left(\text{since } \tfrac{s-1}{s} \ge \tfrac{1}{2} \text{ for all } s \ge 2\right) \\ &= \left(\frac{1}{2}\right)^n (n-\nu(n))^3 \sum_{s=\lceil n/2 \rceil}^{n-\nu(n)} \binom{n}{n-s}\left(\sqrt{1-q}\right)^{n-s} \\ &\le \left(\frac{1}{2}\right)^n (n-\nu(n))^3 \left(1+\sqrt{1-q}\right)^n \\ &= \frac{(n-\nu(n))^3}{\left(\frac{2}{1+\sqrt{1-q}}\right)^n} \end{aligned}$$

Now we consider the regime $s = n - \nu(n) + 1, \dots, n-1$. Note that

$$\begin{aligned} E_s &= \binom{n}{s}\left(\frac{1}{2}\right)^n \sum_{k=1}^{s-1} \binom{s}{k}\binom{n-s}{\lfloor \frac{n-s}{s}k \rfloor}(1-q)^{\frac{n-s}{s}k(s-k)} \\ &\le \binom{n}{s}\left(\frac{1}{2}\right)^n 2^{n-s} \sum_{k=1}^{s-1} \binom{s}{k}(1-q)^k \\ &\le \binom{n}{s}\left(\frac{1}{2}\right)^s \left(\frac{1}{2}\right)^{n-s} 2^{n-s}(1+(1-q))^s \\ &= \binom{n}{s}\left(1-\frac{q}{2}\right)^s. \end{aligned}$$

It follows that

$$\begin{aligned}
\sum_{s=n-\nu(n)+1}^{n-1} E_s &\leq \sum_{s=n-\nu(n)+1}^{n-1} \binom{n}{s} \left(1-\frac{q}{2}\right)^s \\
&= \sum_{t=1}^{\nu(n)-1} \binom{n}{n-t} \left(1-\frac{q}{2}\right)^{n-t} \\
&= \left(1-\frac{q}{2}\right)^n \sum_{t=1}^{\nu(n)-1} \binom{n}{t} \left(\frac{2}{2-q}\right)^t \\
&\leq \left(1-\frac{q}{2}\right)^n \sum_{t=1}^{\nu(n)-1} \frac{n^t}{t!} \left(\frac{2}{2-q}\right)^t \\
&= \left(1-\frac{q}{2}\right)^n \sum_{t=1}^{\nu(n)-1} \frac{1}{t!} \left(\frac{2n}{2-q}\right)^t .
\end{aligned}$$

Let $H_t = \frac{1}{t!}\left(\frac{2n}{2-q}\right)^t$. Then, for $t = 1, \dots, \nu(n)-1$, we have

$$\begin{aligned}
\frac{H_{t+1}}{H_t} &= \frac{t!}{(t+1)!}\left(\frac{2n}{2-q}\right)^{t+1}\left(\frac{2-q}{2n}\right)^t \\
&= \frac{2n}{(2-q)(t+1)} \\
&\geq \frac{2n}{(2-q)\nu(n)} \\
&= \frac{2n}{(2-q)(2\log_r((n-1)/2)+3)} \\
&\geq 1 \qquad\qquad \text{for all sufficiently large } n.
\end{aligned}$$

Therefore, for sufficiently large n,

$$\begin{aligned}
\sum_{s=n-\nu(n)+1}^{n-1} E_s &\leq \left(1-\frac{q}{2}\right)^n \sum_{t=1}^{\nu(n)-1} \frac{1}{t!}\left(\frac{2n}{2-q}\right)^t \\
&\leq \left(1-\frac{q}{2}\right)^n (\nu(n)-1)\frac{1}{(\nu(n)-1)!}\left(\frac{2n}{2-q}\right)^{\nu(n)-1} \\
&= \frac{\left(\frac{2n}{2-q}\right)^{\nu(n)-1}}{(\nu(n)-2)!\left(\frac{2}{2-q}\right)^n}.
\end{aligned}$$

Putting all of this together implies that for n sufficiently large,

$\mathbb{P}\big(B \in \mathcal{B}(n;q)$ satisfies (SD3)$\big)$

$$
\begin{aligned}
&\leq 2 \sum_{s=\lceil n/2 \rceil}^{n-1} E_s + 2 \sum_{s=\lceil n/2 \rceil}^{n-1} \binom{n}{s} \left(\frac{1}{2}\right)^n (n-s)(1-q)^s \\
&\leq 2 \sum_{s=\lceil n/2 \rceil}^{n-\nu(n)} E_s + 2 \sum_{s=n-\nu(n)+1}^{n-1} E_s + 2 \binom{n}{\frac{n}{2}} \left(n - \frac{n}{2}\right) (1-q)^{n/2} \\
&\leq \frac{2(n-\nu(n))^3}{\left(\frac{2}{1+\sqrt{1-q}}\right)^n} + \frac{2\left(\frac{2n}{2-q}\right)^{\nu(n)-1}}{(\nu(n)-2)!\left(\frac{2}{2-q}\right)^n} + \frac{n^2}{2}(1-q)^{n/2} \\
&= \frac{2(n - 2\log_r \frac{n-1}{2} - 3)^2}{\left(\frac{2}{1+\sqrt{1-q}}\right)^n} + \frac{2\left(\frac{2n}{2-q}\right)^{2\log_r \frac{n-1}{2}+2}}{\left(2\log_r \frac{n-1}{2} + 1\right)!\left(\frac{2}{2-q}\right)^n} + \frac{n^2}{2}(1-q)^{n/2},
\end{aligned}
$$

and therefore $\lim_{n\to\infty} \mathbb{P}(B \in \mathcal{B}(n;q) \text{ satisfies (SD3)}) = 0$.

Finally, we put all the pieces together:

$$
\begin{aligned}
&\lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ is Sidney decomposable}\big) \\
&\quad = \lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD1)}\big) + \lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD2)}\big) \\
&\qquad\qquad + \lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies (SD3)}\big) = 0. \quad \square
\end{aligned}
$$

One interesting corollary of Theorem 3.3.4 and Theorem 3.3.5 is that for almost all 0-1 bipartite instances, under either $\mathcal{B}(n_1, n_2; q)$ or $\mathcal{B}(n;q)$, *every* feasible schedule is a 2-approximation. This result is quite interesting. However, as we are about to see in the next section, we can show something stronger.

3.4 Two-dimensional Gantt charts and 0-1 bipartite instances

Two-dimensional (2D) Gantt charts provide an elegant, geometric way of understanding single-machine completion-time-objective scheduling problems. They were initially introduced by Eastman et al. (1964), and recently revived by Goemans and Williamson (2000) to give alternate proofs for various results related to $1 \,|\, \text{prec} \,|\, \sum w_j C_j$, including Lawler's (1978) algorithm for $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ with series-parallel precedence constraints.

In a traditional Gantt chart, the horizontal axis corresponds to processing time. In a 2D Gantt chart, the horizontal axis corresponds to processing time, and the vertical axis corresponds to weight. Suppose we have an instance $(N, A, (p_i)_{i\in N}, (w_i)_{i\in N})$ of $1 \,||\, \sum w_j C_j$. The 2D Gantt chart is constructed for a permutation schedule $(1, \dots, n)$ as follows. Each job $j \in N$ is represented by a rectangle of width p_j and height w_j, whose position in the chart is defined by a startpoint and an endpoint. The startpoint of the first job (job 1) in the schedule is $(0, \sum_{j\in N} w_j)$, and its endpoint is $(p_1, \sum_{j\in N} w_j - w_1)$. For all subsequent jobs in the schedule, the startpoint (t, w) of job j is the endpoint of the previous job $j-1$, and its endpoint is $(t+p_j, w-w_j)$. The completion time of a job in this schedule is the time component of its endpoint. See Figure 3-1 for an example.

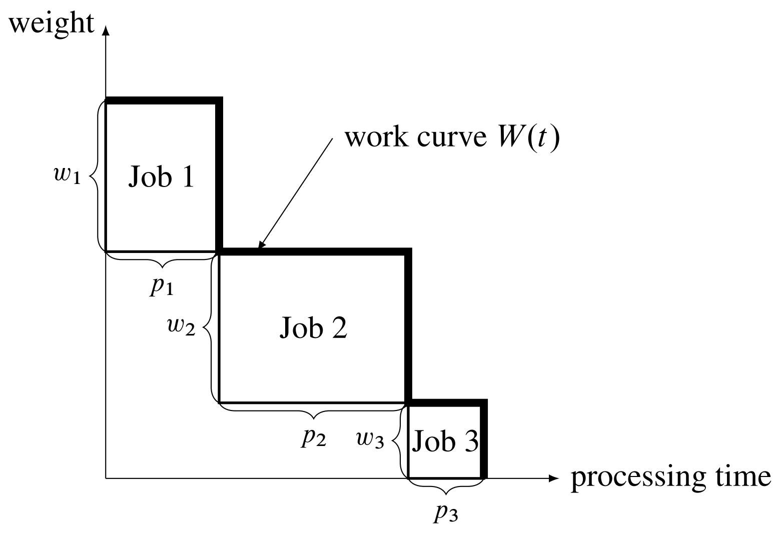

Figure 3-1: An example of a two-dimensional Gantt chart.

The *work curve* $W(t)$ formed by the upper side of each rectangle (the bold line in Figure 3-1) is the total weight of jobs that have not been completed at time t. The area under the work curve is equal to the sum of weighted completion times for the schedule represented by the 2D Gantt chart.

Interestingly, it turns out that when q is fixed, the area under the work curve of random 0-1 bipartite graphs under $\mathcal{B}(n_1, n_2; q)$ and $\mathcal{B}(n; q)$ is "large." We formalize this notion now. Consider the 2D Gantt chart for an optimal schedule of a 0-1 bipartite instance B with $|N_1| = n_1$ and $|N_2| = n_2$. Note that any 2D Gantt chart for such an instance starts at $(0, n_2)$ and ends at $(n_1, 0)$. Also observe that all jobs in N_1 are represented by a horizontal line segment of length 1, and that all jobs in N_2 are represented by a vertical line segment of length 1. Define R_B to be the region between the optimal work curve and the frontier formed by the lines $\{(t, w) : t = n_1\}$ and $\{(t, w) : w = n_2\}$ (see Figure 3-2). Also, define the following parameterized condition on a 0-1 bipartite instance B, for any $\alpha \in (0, 1)$:

(R-α) A rectangle of width αn_1 and height αn_2 cannot fit in R_B.

We now show that for any fixed $\alpha \in (0, 1)$, the condition (R-α) is satisfied for almost all 0-1 bipartite instances.

Theorem 3.4.1.

(a) *For any fixed* $c, d \in \mathbb{Z}_{>0}$, $\alpha \in (0, 1)$ *and* $q \in (0, 1)$,

$$\lim_{t\to\infty} \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \textit{ satisfies } (\text{R-}\alpha)\big) = 1.$$

(b) *For any fixed* $\alpha \in (0, 1)$ *and* $q \in (0, 1)$,

$$\lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n; q) \textit{ satisfies } (\text{R-}\alpha)\big) = 1.$$

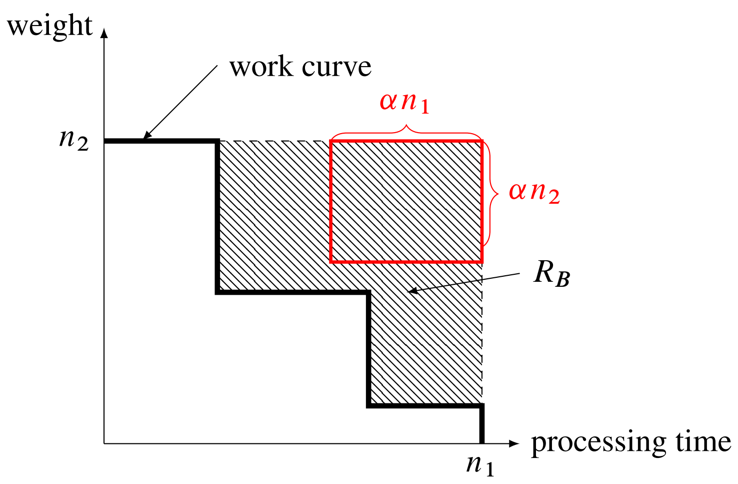

Figure 3-2: 2D Gantt chart depicting region R_B.

Proof. Fix a 0-1 bipartite instance $B = (N_1, N_2, A)$ with $|N_1| = n_1$ and $|N_2| = n_2$. If B does not satisfy (R-α), that is, a rectangle of width αn_1 and height αn_2 can fit in R_B, then there exists a set of $\lceil \alpha n_2 \rceil$ jobs from N_2 that has at most $n_1 - \lceil \alpha n_1 \rceil$ predecessors in N_1. In other words, if a rectangle of width αn_1 and height αn_2 can fit in R_B, then there exists a set of $\lceil \alpha n_2 \rceil$ jobs from N_2 and a set of $\lceil \alpha n_1 \rceil$ jobs from N_1 with no precedence constraints between them.

Applying the above discussion to $\mathcal{B}(ct, dt; q)$, we have that

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ does not satisfy (R-}\alpha\text{)}\big) \\
&\quad \le \mathbb{P}\left(\exists X \subseteq N_1, Y \subseteq N_2 : \begin{array}{c} |X| = \lceil \alpha ct \rceil, |Y| = \lceil \alpha dt \rceil, \\ \text{no precedence constraints between } X \text{ and } Y \end{array}\right) \\
&\quad \le \binom{ct}{\lceil \alpha ct \rceil}\binom{dt}{\lceil \alpha dt \rceil}(1-q)^{\lceil \alpha ct \rceil \lceil \alpha dt \rceil} \\
&\quad \le \left(\frac{ect}{\lceil \alpha ct \rceil}\right)^{\lceil \alpha ct \rceil}\left(\frac{edt}{\lceil \alpha dt \rceil}\right)^{\lceil \alpha dt \rceil}(1-q)^{\lceil \alpha ct \rceil \lceil \alpha dt \rceil} \\
&\quad \le \left(\frac{ect}{\alpha ct}\right)^{\alpha ct + 1}\left(\frac{edt}{\alpha dt}\right)^{\alpha dt + 1}(1-q)^{\alpha^2 cdt^2} \\
&\quad = \left(\frac{e}{\alpha}\right)^{\alpha(c+d)t+2}(1-q)^{\alpha^2 cdt^2}.
\end{aligned}$$

Therefore, $\lim_{t\to\infty} \mathbb{P}(B \in \mathcal{B}(ct, dt; q)$ does not satisfy (R-α)$) = 0$, which establishes (a).

Now, we turn to proving (b). Again, applying the above discussion to $\mathcal{B}(n; q)$, we have that

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(n; q) \text{ does not satisfy (R-}\alpha\text{)} \mid |N_1| = k, |N_2| = n-k\big) \\
&\quad \le \mathbb{P}\left(\begin{array}{c} \exists X \subseteq N_1, Y \subseteq N_2, \\ |N_1| = k, |N_2| = n-k \end{array} : \begin{array}{c} |X| = \lceil \alpha k \rceil, |Y| = \lceil \alpha(n-k) \rceil, \\ \text{no precedence constraints between } X \text{ and } Y \end{array}\right)
\end{aligned}$$

$$\leq \binom{k}{\lceil \alpha k \rceil}\binom{n-k}{\lceil \alpha(n-k)\rceil}(1-q)^{\lceil \alpha k\rceil\lceil \alpha(n-k)\rceil}$$
$$\leq \binom{k}{\lceil \alpha k \rceil}\binom{n-k}{\lceil \alpha(n-k)\rceil}(1-q)^{\alpha^2 k(n-k)}.$$

Therefore, by conditioning on the size of N_1 and N_2,

$$\begin{aligned}
&\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ does not satisfy (R-}\alpha\text{)}\big) \\
&= \sum_{k=1}^{n-1} \mathbb{P}\left(\begin{array}{c} B \in \mathcal{B}(n;q) \\ \text{does not satisfy (R-}\alpha\text{)} \end{array} \middle| |N_1| = k, |N_2| = n-k \right) \mathbb{P}\left(\begin{array}{c} |N_1| = k, \\ |N_2| = n-k \end{array} \right) \\
&\leq \sum_{k=1}^{n-1} \binom{n}{k}\left(\frac{1}{2}\right)^n \binom{k}{\lceil \alpha k\rceil}\binom{n-k}{\lceil \alpha(n-k)\rceil}(1-q)^{\alpha^2 k(n-k)} \\
&\leq \sum_{k=1}^{n-1} \binom{n}{k}\left(\frac{1}{2}\right)^k \binom{k}{\lceil \alpha k\rceil}\left(\frac{1}{2}\right)^{n-k}\binom{n-k}{\lceil \alpha(n-k)\rceil}(1-q)^{\alpha^2 k(n-k)} \\
&\leq \sum_{k=1}^{n-1} \binom{n}{k}(1-q)^{\alpha^2 k(n-k)}.
\end{aligned}$$

Let $r = (1-q)^{-1}$, and let $M_{\alpha,r}$ be some constant such that $\alpha^2(n-3) - \log_r((n-1)/2) \geq 0$ for all $n \geq M_{\alpha,r}$. Define

$$D_k = \binom{n}{k}(1-q)^{\alpha^2 k(n-k)} \qquad k = 1, \ldots, n-1.$$

We would like to show that $D_k \geq D_{k+1}$ for $k = 1, \ldots, \lfloor (n-1)/2 \rfloor$ and $n \geq M_{\alpha,r}$, or equivalently,

$$\log_r \frac{n-k}{k+1} \leq \alpha^2(n-2k-1) \quad \text{for } k = 1, \ldots, \lfloor (n-1)/2\rfloor \text{ and } n \geq M_{\alpha,r}.$$

Define the function

$$\Delta(x) = \alpha^2(n-2x-1) - \log_r(n-x) + \log_r(x+1).$$

As in the proof of Theorem 3.3.4, we can show that $\Delta(x)$ is concave on $[1, (n-1)/2]$. Evaluating $\Delta(x)$ at $x = 1$ and $x = (n-1)/2$ yields

$$\Delta(1) = \alpha^2(n-3) - \log_r \frac{n-1}{2} \qquad \Delta\left(\frac{n-1}{2}\right) = 0.$$

So $\Delta(1) \geq 0$ when $n \geq M_{\alpha,r}$, and so $\Delta(x) \geq 0$ for all $x \in [1, (n-1)/2]$ when $n \geq M_{\alpha,r}$. Therefore, we have that $D_k \geq D_{k+1}$ for $k = 1, \ldots, \lfloor (n-1)/2\rfloor$, for all $n \geq M_{a,r}$.

Since $D_k = D_{n-k}$ for all $k = 1, \ldots, \lfloor (n-1)/2 \rfloor$, it follows that $D_1 \geq D_k$ for all

$k = 1, \ldots, n-1$, when $n \geq M_{\alpha,r}$. Therefore, for $n \geq M_{\alpha,r}$,

$$\mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ does not satisfy (R-}\alpha\text{)}\big) \leq \sum_{k=1}^{n-1} D_k \leq nD_1 = n^2(1-q)^{\alpha^2(n-1)}.$$

It follows that $\lim_{n\to\infty} \mathbb{P}(B \in \mathcal{B}(n;q) \text{ does not satisfy (R-}\alpha\text{)}) = 0$, which establishes (b). □

3.4.1 Almost all 0-1 bipartite instances have all schedules almost optimal

Having the geometric property (R-α) satisfied by the 2D Gantt charts for almost all 0-1 bipartite instances translates into some interesting consequences. For example, with Theorem 3.4.1 in hand, we can show the following. Let OPT(B) denote the optimal value of instance B, and let val(B, S) denote the objective value of (feasible) schedule S for instance B.

Theorem 3.4.2.

(a) *For any fixed* $c, d \in \mathbb{Z}_{>0}$, $\alpha \in (0,1)$, *and* $q \in (0,1)$,

$$\lim_{t\to\infty} \mathbb{P}\left(B \in \mathcal{B}(ct, dt; q) \textit{ satisfies } \frac{\text{val}(B,S)}{\text{OPT}(B)} \leq (1-\alpha)^{-2} \begin{array}{c}\textit{for all feasible}\\ \textit{schedules } S\end{array} \right) = 1.$$

(b) *For any fixed* $\alpha \in (0,1)$ *and* $q \in (0,1)$,

$$\lim_{n\to\infty} \mathbb{P}\left(B \in \mathcal{B}(n; q) \textit{ satisfies } \frac{\text{val}(B,S)}{\text{OPT}(B)} \leq (1-\alpha)^{-2} \begin{array}{c}\textit{for all feasible}\\ \textit{schedules } S\end{array} \right) = 1.$$

Proof. Consider some 0-1 bipartite instance B with $|N_1| = n_1$ and $|N_2| = n_2$. By the 2D Gantt chart in Figure 3-3, we see that if (R-α) is satisfied—that is, if a rectangle of width αn_1 and height αn_2 cannot fit in the region R_B—then OPT(B) $> n_1 n_2 (1-\alpha)^2$. Since the objective value of any feasible schedule of an instance B is at most $n_1 n_2$, this implies that if (R-α) is satisfied,

$$\frac{\text{val}(B,S)}{\text{OPT}(B)} \leq \frac{n_1 n_2}{n_1 n_2 (1-\alpha)^2} = (1-\alpha)^{-2},$$

which implies the claim. □

So, for 0-1 bipartite instances, almost always, *all* feasible schedules are arbitrarily close to optimal! This seems especially paradoxical in light of the difficulty of obtaining an approximation algorithm with performance guarantee better than 2: for a given $\epsilon > 0$, when n is sufficiently large, the naïve algorithm—take any feasible schedule—yields a $(1+\epsilon)$-approximate schedule with high probability.

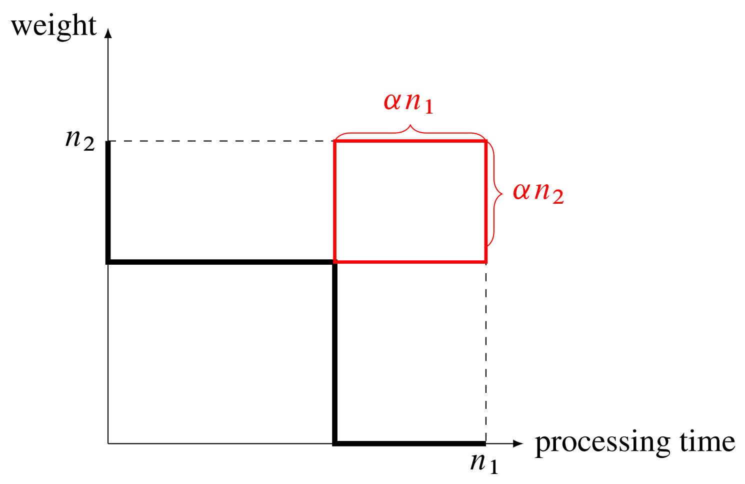

Figure 3-3: 2D Gantt chart that satisfies (R-α) with the smallest area under its work curve.

3.4.2 A lower bound on integrality gaps for almost all 0-1 bipartite instances

The asymptotic behavior of the 2D Gantt charts of random 0-1 bipartite instances also has some interesting consequences regarding the integrality gap of various LP relaxations of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. Let $\text{LP}(B)$ denote the optimal value of the LP relaxation [P-LP], [CH-LP], or [CS-LP]. Before proceeding, we need the following version of the Chernoff bound.

Lemma 3.4.3 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that for $i = 1, \ldots, n$, $\mathbb{P}(X_i = 1) = q$ and $\mathbb{P}(X_i = 0) = 1 - q$ with $q \in (0,1)$. Then for $S = \sum_{i=1}^n X_i$, $\mu = \mathbb{E}(S) = qn$, and any $\delta > 0$,*

$$\mathbb{P}\big(S > (1+\delta)\mu\big) < \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu .$$

Chekuri and Motwani (1999) showed that the integrality gap of [P-LP] is 2, using a family of 0-1 bipartite instances with $|N_1| = |N_2|$. In the following, we show that almost all 0-1 bipartite instances have integrality gap $2/(1+q)$, under both $\mathcal{B}(n_1, n_2; q)$ and $\mathcal{B}(n; q)$.

Theorem 3.4.4.

(a) *For any fixed $c, d \in \mathbb{Z}_{>0}$, $\alpha \in (0,1)$ and $\delta > 0$,*

$$\lim_{t \to \infty} \mathbb{P}\left(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } \frac{\text{OPT}(B)}{\text{LP}(B)} \geq \frac{2(1-\alpha)^2}{1 + (1+\delta)q} \right) = 1.$$

(b) *For any fixed $\alpha \in (0,1)$ and $\delta > 0$,*

$$\lim_{n \to \infty} \mathbb{P}\left(B \in \mathcal{B}(n; q) \text{ satisfies } \frac{\text{OPT}(B)}{\text{LP}(B)} \geq \frac{2(1-\alpha)^2}{1 + (1+\delta)q} \right) = 1.$$

Proof. Consider a 0-1 bipartite instance $B = (N_1, N_2, A)$ with $|N_1| = n_1$ and $|N_2| = n_2$. It is straightforward to show that

$$\delta_{ij} = \begin{cases} 1 & \text{if } (i, j) \in A \\ \frac{1}{2} & \text{otherwise} \end{cases}$$

is a feasible solution to [P-LP], [CH-LP], and [CS-LP], and that this solution has objective value $\frac{1}{2}(n_1 n_2 + |A|)$. Therefore, $\mathrm{LP}(B) \leq \frac{1}{2}(n_1 n_2 + |A|)$. By similar arguments to those in the proof of Theorem 3.4.2, if B satisfies (R-α) and $|A| \leq (1+\delta) q n_1 n_2$, then

$$\frac{\mathrm{OPT}(B)}{\mathrm{LP}(B)} > \frac{n_1 n_2 (1-\alpha)^2}{\frac{1}{2}(n_1 n_2 + |A|)} \geq \frac{n_1 n_2 (1-\alpha)^2}{\frac{1}{2}(n_1 n_2 + (1+\delta) q n_1 n_2)} = \frac{2(1-\alpha)^2}{1 + (1+\delta) q}.$$

First, we show (a). By the Chernoff bound in Lemma 3.4.3, we have that

$$\mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } |A| > (1+\delta) q (ct)(dt)\big) < \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{q(ct)(dt)}.$$

Since $\delta > 0$, we have that $e^{\delta} < (1+\delta)^{(1+\delta)}$. Therefore,

$$\lim_{t \to \infty} \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } |A| > (1+\delta) q c d t^2\big) = 0.$$

By the above discussion, we have that

$$\begin{aligned} \mathbb{P}\left(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } \frac{\mathrm{OPT}(B)}{\mathrm{LP}(B)} < \frac{2(1-\alpha)^2}{1+(1+\delta)q}\right) \\ \leq \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ does not satisfy (R-}\alpha)\big) \\ + \mathbb{P}\big(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } |A| > (1+\delta) q c d t^2\big), \end{aligned}$$

and so

$$\lim_{t \to \infty} \mathbb{P}\left(B \in \mathcal{B}(ct, dt; q) \text{ satisfies } \frac{\mathrm{OPT}(B)}{\mathrm{LP}(B)} < \frac{2(1-\alpha)^2}{1+(1+\delta)q}\right) = 0.$$

Now, we show (b). By conditioning on the size of N_1 and N_2, using the Chernoff bound from Lemma 3.4.3, and observing that $e^{\delta} < (1+\delta)^{(1+\delta)}$ for any $\delta > 0$, we obtain

$$\begin{aligned} &\mathbb{P}\big(B \in \mathcal{B}(n; q) \text{ satisfies } |A| > (1+\delta) q n_1 n_2\big) \\ &\quad = \sum_{k=1}^{n} \mathbb{P}\left(\begin{array}{c} B \in \mathcal{B}(n; q) \text{ satisfies} \\ |A| > (1+\delta) q n_1 n_2 \end{array} \middle| n_1 = k, n_2 = n-k \right) \mathbb{P}(n_1 = k, n_2 = n-k) \\ &\quad < \sum_{k=1}^{n} \binom{n}{k} \left(\frac{1}{2}\right)^{n} \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right)^{qk(n-k)} \end{aligned}$$

$$\leq \sum_{k=1}^{n} \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right)^{qk(n-k)}$$
$$\leq n \left(\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}} \right)^{q(n-1)}.$$

Therefore,

$$\lim_{n\to\infty} \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies } |A| > (1+\delta)qn_1n_2\big) = 0.$$

By the above discussion, we have that

$$\mathbb{P}\left(B \in \mathcal{B}(n;q) \text{ satisfies } \frac{\mathrm{OPT}(B)}{\mathrm{LP}(B)} < \frac{2(1-\alpha)^2}{1+(1+\delta)q} \right)$$
$$\leq \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ does not satisfy (R-}\alpha)\big)$$
$$+ \mathbb{P}\big(B \in \mathcal{B}(n;q) \text{ satisfies } |A| > (1+\delta)qcdt^2\big),$$

and so

$$\lim_{n\to\infty} \mathbb{P}\left(B \in \mathcal{B}(n;q) \text{ satisfies } \frac{\mathrm{OPT}(B)}{\mathrm{LP}(B)} < \frac{2(1-\alpha)^2}{1+(1+\delta)q} \right) = 0. \qquad \square$$

It is known that the linear programming relaxation in completion time variables [QW-LP] is weaker than [P-LP] (Schulz 1996); as a result, the analogue of Theorem 3.4.4 for [QW-LP] also holds.

3.5 Conclusion

In this work, we studied 0-1 bipartite instances of the scheduling problem $1 \,|\, \text{prec} \,|\, \sum w_j C_j$. Although this class of instances appears rather restricted, it has been shown that in fact, the approximability of these restricted instances is virtually identical to the approximability of arbitrary instances. Therefore, in order to design better approximation algorithms for arbitrary instances of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$, it suffices to restrict our attention to these simple 0-1 bipartite instances. In an effort to gain a better understanding of these instances, we study them from a probabilistic point of view. We showed that under probability distributions typically considered in random graph theory, almost all 0-1 bipartite instances of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ are non-Sidney-decomposable, and as a corollary, for almost all 0-1 bipartite instances, *any* feasible schedule is a 2-approximation. In fact, using the two-dimensional Gantt charts of Eastman et al. (1964), we can show something stronger: for almost all 0-1 bipartite instances, all feasible schedules are arbitrarily close to optimal. In addition, for almost all 0-1 bipartite instances, we give a lower bound on the integrality gap of the linear ordering LP relaxations of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ due to Potts (1980), Chudak and Hochbaum (1999), and Correa and Schulz (2005). This lower bound approaches 2 as the precedence constraints become sparser in expectation.

One natural direction for future research would be to investigate the behavior of other models of random 0-1 bipartite instances. In this work, we studied the models $\mathcal{B}(n_1, n_2; q)$ and $\mathcal{B}(n; q)$ where q is a constant independent of n; it would be interesting, for instance, to see what happens when $q = c/n$ for some constant c. It would also be interesting to see what happens under the random instance model $\mathcal{B}(n; q)$ when the probability of a job being in N_1 is not equal to the probability of a job being in N_2. Another natural direction for future research would be to consider models for random instances with arbitrary weights, processing times, and precedence constraints.

3.A Examples

Example 3.A.1. The following example gives a stiff 0-1 bipartite instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ without fixed costs that has a feasible schedule whose objective value is more than a factor of 2 away from the optimal objective value. Consider the following instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ with jobs $N = \{1, 2, 3, 4\}$. The processing times and weights are as follows:

i	1	2	3	4
p_i	1	1	0	0
w_i	0	0	1	1

The precedence constraints are displayed in Figure 3-4. Consider the sequence $(1, 2, 3, 4)$.

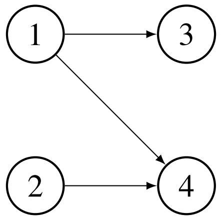

Figure 3-4: Precedence constraints for the instance in Example 3.A.1.

The objective value for this sequence is

$$\begin{aligned} p_1 w_2 \delta_{12} + p_2 w_1 \delta_{21} + p_2 w_3 \delta_{23} + p_3 w_2 \delta_{32} + p_3 w_4 \delta_{34} + p_4 w_3 \delta_{43} \\ = (0 \cdot 1) + (0 \cdot 0) + (1 \cdot 1) + (0 \cdot 0) + (0 \cdot 1) + (0 \cdot 0) \\ = 1. \end{aligned}$$

The objective value for the sequence $(1, 3, 2, 4)$ is

$$\begin{aligned} p_1 w_2 \delta_{12} + p_2 w_1 \delta_{21} + p_2 w_3 \delta_{23} + p_3 w_2 \delta_{32} + p_3 w_4 \delta_{34} + p_4 w_3 \delta_{43} \\ = (0 \cdot 1) + (0 \cdot 0) + (1 \cdot 0) + (0 \cdot 1) + (0 \cdot 1) + (0 \cdot 0) \\ = 0 \end{aligned}$$

Therefore, the maximum relative gap between the optimal value and any feasible schedule is unbounded. □

Example 3.A.2. The following example shows that there exists a 0-1 bipartite instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ that is non-Sidney-decomposable, for which the "all $1/2$" solution is not LP-optimal. Consider the following instance of $1 \,|\, \text{prec} \,|\, \sum w_j C_j$ with jobs $N = \{1, 2, 3, 4, 5\}$. The weights and processing times are shown in the table below.

i	1	2	3	4	5
p_i	1	1	0	0	0
w_i	0	0	1	1	1

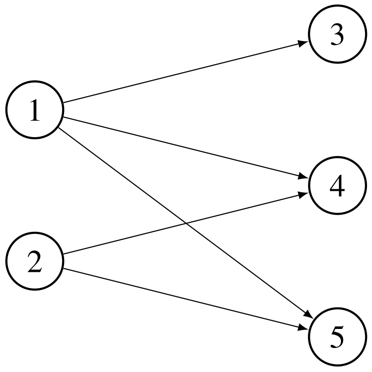

Figure 3-5: Precedence constraints for the instance in Example 3.A.2.

The precedence constraints are displayed in Figure 3-5. There are ten ordered pairs of incomparable jobs: $(1,2)$, $(2,1)$, $(3,4)$, $(4,3)$, $(4,5)$, $(5,4)$, $(3,5)$, $(5,3)$, $(2,3)$, $(3,2)$. The linear program [CS-LP] for this instance is

$$\begin{array}{lll}
\text{minimize} & 5+\delta_{23} & \\
\text{subject to} & \delta_{12}+\delta_{21}\geq 1 & \\
& \delta_{34}+\delta_{43}\geq 1 & \\
& \delta_{45}+\delta_{54}\geq 1 & \\
& \delta_{35}+\delta_{53}\geq 1 & \\
& \delta_{23}+\delta_{32}\geq 1 & \\
& \delta_{12}+\delta_{23}\geq 1 & \\
& \delta_{23}+\delta_{34}\geq 1 & \\
& \delta_{23}+\delta_{35}\geq 1 & \\
& \delta_{ij}\geq 0 & \text{for all } i,j\in N : i \parallel j.
\end{array}$$

An optimal solution to this linear program is

$$\begin{array}{lllll}
\delta_{12}=1 & \delta_{34}=1 & \delta_{45}=1 & \delta_{35}=1 & \delta_{23}=0\\
\delta_{21}=0 & \delta_{43}=0 & \delta_{54}=0 & \delta_{53}=0 & \delta_{32}=1,
\end{array}$$

and the optimal objective value is 5. However, the "all $1/2$" solution has an objective value of 5.5. □

Chapter 4

Scheduling in concurrent open shop environments

4.1 Introduction

Consider the following scheduling setting, sometimes known as the *concurrent open shop model*, or the *order scheduling model*. We have a set of machines $M = \{1, \dots, m\}$, with each machine capable of processing one component type. We have a set of jobs $N = \{1, \dots, n\}$, with each job requiring specific quantities of processing for each of the m component types. Each job $j \in N$ has a weight $w_j \in \mathbb{R}_{\geq 0}$, and the processing time that job j requires on machine i is $p_{ij} \in \mathbb{R}_{\geq 0}$. Components are independent of each other: in particular, components from the same job can be processed in parallel. A job is completed when all its components are completed. In this chapter, we focus on minimizing the sum of weighted completion times in a concurrent open shop. Following the notation of Leung et al. (2005), we denote this problem by PD $||$ $\sum w_j C_j$ in the standard classification scheme of Graham et al. (1979).

The concurrent open shop model can be considered as a variant of the classical open shop model in which operations belonging to the same job can be processed concurrently. This model has a variety of applications in manufacturing, including automobile and airplane maintenance and repair (Yang 1998), and orders with multiple components in manufacturing environments (Sung and Yoon 1998). This model also has applications in distributed computing (Garg et al. 2007).

The problem PD $||$ $\sum w_j C_j$ was first studied by Ahmadi and Bagchi (1990). A number of authors have since shown that various special cases of this problem are NP-hard (Ahmadi and Bagchi 1990; Sung and Yoon 1998; Chen and Hall 2001; Leung et al. 2005); it turns out that this problem is strongly NP-hard, even when all jobs have unit weight, and the number of machines m is fixed to be 2 (Roemer 2006). Recently, Garg et al. (2007) showed that PD $||$ $\sum w_j C_j$ is APX-hard, even when all jobs have unit weight and either zero or unit processing time. It is also known that this problem is inapproximable within a factor of $4/3 - \epsilon$ if the Unique Games Conjecture holds (see Appendix 4.A).

Quite a bit of attention has been devoted to designing heuristics for this problem. For example, Sung and Yoon (1998), Wang and Cheng (2003), and Leung et al. (2005) have proposed various priority rules for this problem; all of the priority rules they studied were shown to either have a performance guarantee of m, or have an unbounded performance guarantee. Ahmadi et al. (2005) also proposed various heuristics for this problem and showed that they all have a performance guarantee of m. Wang and Cheng (2003) used a time-indexed linear programming formulation of this problem to obtain a 5.83-approximation algorithm. Finally, several groups of authors have independently observed that a linear programming relaxation of this problem in completion time variables with the parallel inequalities of Wolsey (1985) and Queyranne (1993), combined with a result of Schulz (1996), yields a 2-approximation algorithm (Chen and Hall 2001; Garg et al. 2007; Leung et al. 2007). Note that when $m = 1$, or when each job consists of components all with equal processing time, PD $| \, |$ $\sum w_j C_j$ reduces to the classic problem of minimizing the sum of weighted completion times on a single machine.

4.1.1 Contributions of this work

We begin in Section 4.2 by presenting some interesting properties of various linear programming relaxations for this concurrent open shop problem. Then in Section 4.3, we present a simple combinatorial primal-dual approximation algorithm that has a performance guarantee of 2. Although the approximation algorithm independently discovered by Chen and Hall (2001), Garg et al. (2007), and Leung et al. (2007) achieves the same performance guarantee, their algorithm requires solving a linear program. Our algorithm, on the other hand, runs in $O(n(m + n))$ time. Finally, we conclude in Section 4.4 by mentioning some directions for future research.

4.2 Mathematical programming formulations and relaxations

The existing integer programming formulations and linear programming relaxations for various machine scheduling problems provide natural starting points for modeling the problem of minimizing the sum of weighed completion times in a concurrent open shop. We present two types of mathematical programming formulations for PD $| \, |$ $\sum w_j C_j$, one based on completion time variables, and the other based on linear ordering variables.

4.2.1 Completion time variables

Chen and Hall (2001) proposed the following linear programming relaxation of PD $| \, |$ $\sum w_j C_j$:

$$
\begin{array}{lllll}
\text{[CT-1]} & \text{minimize} & \displaystyle\sum_{j \in N} w_j C_j & & (4.2.1\text{a}) \\
& \text{subject to} & \displaystyle\sum_{j \in S} p_{ij} C_{ij} \geq f_i(S) & \text{for all } i \in M, S \subseteq N, & (4.2.1\text{b}) \\
& & C_j \geq C_{ij} & \text{for all } i \in M, j \in N, & (4.2.1\text{c})
\end{array}
$$

where C_{ij} represents the completion time of job j's component on machine i, C_j represents the completion time of job j, and

$$f_i(S) = \frac{1}{2}\sum_{j \in S} p_{ij}^2 + \frac{1}{2}\left(\sum_{j \in S} p_{ij}\right)^2 \quad \text{for all } i \in M, S \subseteq N.$$

The constraints (4.2.1b) are the so-called *parallel inequalities* (Wolsey 1985; Queyranne 1993) for each of the m machines. These inequalities are known to be valid for the completion time vectors of jobs on a single machine; in fact, they are sufficient to describe the convex hull of completion time vectors for jobs on a single machine. It immediately follows that [CT-1] is a valid relaxation for this scheduling problem.

By substituting the constraints (4.2.1c) into the constraints (4.2.1b), we obtain a further relaxation of PD $||$ $\sum w_j C_j$ in fewer completion time variables:

$$\begin{aligned} &\text{[CT-2]} \quad \text{minimize} && \sum_{j \in N} w_j C_j && (4.2.2a) \\ &\text{subject to} && \sum_{j \in S} p_{ij} C_j \geq f_i(S) \quad \text{for all } i \in M, S \subseteq N. && (4.2.2b) \end{aligned}$$

The relaxation [CT-2] will serve as the basis of our analysis for the algorithm presented in Section 4.3.

For any optimal solution $C = (C_j)_{j \in N}$ to [CT-2], we denote its family of *tight sets* as

$$\tau(C) = \left\{ S \subseteq N : \sum_{j \in S} p_{ij} C_j = f_i(S) \text{ for some } i \in M \right\}.$$

One interesting feature of [CT-2] is that when $p_{ij} > 0$ for all $i \in M$ and $j \in N$, its tight sets are nested. This is a generalization of a result known for supermodular polyhedra (Edmonds 1970), and has been observed for completion-time-variable LP relaxations of some scheduling problems (e.g. Margot et al. 2003). To show this, we make use of the following lemma from Margot et al. (2003).

Lemma 4.2.1 (Margot et al. 2003). *Suppose $p_{ij} > 0$ for all $i \in M$ and $j \in N$. Let $C = (C_j)_{j \in N}$ be an optimal solution to* [CT-2], *and let $i \in M$ and $S \subseteq N$ such that $\sum_{j \in S} p_{ij} C_j = f_i(S)$. Then,*

(a) $C_k \leq \sum_{j \in S} p_{ij}$ *for all $k \in S$,* (b) $C_k \geq \sum_{j \in S} p_{ij} + p_{ik}$ *for all $k \notin S$.*

Proof. Suppose $k \in S$. Then,

$$f_i(S) - f_i(S \setminus \{k\}) = \frac{1}{2} p_{ik} \left(2 \sum_{j \in S \setminus \{k\}} p_{ij} + p_{ik} \right) + \frac{1}{2} p_{ik}^2 = p_{ik} \sum_{j \in S} p_{ij}.$$

Therefore,

$$p_{ik}C_k = f_i(S) - \sum_{j \in S \setminus \{k\}} p_{ij}C_j \le f_i(S) - f_i(S \setminus \{k\}) = p_{ik} \sum_{j \in S} p_{ij},$$

which establishes (a). Now suppose $k \notin S$. Then,

$$f_i(S \cup \{k\}) - f_i(S) = \frac{1}{2} p_{ik} \left(2 \sum_{j \in S} p_{ij} + p_{ik} \right) + \frac{1}{2} p_{ik}^2 = p_{ik} \left(\sum_{j \in S} p_{ij} + p_{ik} \right).$$

Therefore,

$$p_{ik}C_k \ge f_i(S \cup \{k\}) - \sum_{j \in S} p_{ij}C_j = f_i(S \cup \{k\}) - f_i(S) = p_{ik} \left(\sum_{j \in S} p_{ij} + p_{ik} \right),$$

which establishes (b). □

Lemma 4.2.2. *Suppose $p_{ij} > 0$ for all $i \in M$ and $j \in N$. Let $C = (C_j)_{j \in N}$ be an optimal solution to* [CT-2]. *Then the tight sets of C are nested: for any two distinct sets $S, T \in \tau(C)$, either $S \subseteq T$, or $T \subseteq S$.*

Proof. For the purpose of finding a contradiction, suppose that for some pair of tight sets $S, T \in \tau(C)$, we have that $S \setminus T \ne \emptyset$ and $T \setminus S \ne \emptyset$. Suppose S is tight for machine h—that is, $\sum_{j \in S} p_{hj}C_j = f_h(S)$—and suppose T is tight for machine i. In addition, let k be some element in $S \setminus T$, and let l be some element in $T \setminus S$. By repeated application of Lemma 4.2.1, we have that

$$C_k \le \sum_{j \in S} p_{hj} < \sum_{j \in S} p_{hj} + p_{hl} \le C_l \le \sum_{j \in T} p_{ij} < \sum_{j \in T} p_{ij} + p_{ik} \le C_k,$$

which is a contradiction. □

When $m = 1$, Lemma 4.2.2 implies that the tight sets of a basic feasible solution to [CT-2] are properly nested: that is, there are n tight sets, $\{S_1, \dots, S_n\}$ where $|S_i| = i$ for $i = 1, \dots, n$. When $m \ge 2$, however, this no longer applies, as seen in the following example.

Example 4.2.3. In this example, we show for an optimal solution to [CT-2], the sets that are tight may correspond to multiple machines. Consider the following instance with $m = 2$ and $n = 2$:

j	1	2
w_j	1	1
p_{1j}	3	1
p_{2j}	1	3

An optimal solution to [CT-2] is $C_1^{\mathrm{LP}} = 3.25$, $C_2^{\mathrm{LP}} = 3.25$. It is straightforward to check that the only tight set for $(C_1^{\mathrm{LP}}, C_2^{\mathrm{LP}})$ is $N = \{1, 2\}$, for both machines 1 and 2. □

4.2.2 Linear ordering variables

Instead of explicitly modeling the completion times of each job on each machine, we can model the order in which the jobs are processed on each machine. For every machine $i \in M$, we define the decision variables δ^i_{jk}, where $\delta^i_{jk} = 1$ if job j precedes job k on machine i, and $\delta^i_{jk} = 0$ if job k precedes job j on machine i. These variables are known as *linear ordering variables*. Consider the following integer programming formulation for PD $| \, | \sum w_j C_j$:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j \in N} w_j C_j && && \text{(4.2.3a)}\\
\text{subject to} \quad & \delta^i_{jk} + \delta^i_{kj} = 1 && \text{for all } i \in M, j,k \in N : j \neq k, && \text{(4.2.3b)}\\
& \delta^i_{jk} + \delta^i_{kl} + \delta^i_{lj} \leq 2 && \text{for all } i \in M, && \\
& && \quad j,k,l \in N : j \neq k \neq l \neq j, && \text{(4.2.3c)}\\
& \delta^i_{jk} \in \{0,1\} && \text{for all } i \in M, j,k \in N : j \neq k, && \text{(4.2.3d)}\\
& C_{ij} \geq \sum_{k \in N : k \neq j} p_{ik} \delta^i_{kj} + p_{ij} && \text{for all } i \in M, j \in N, && \text{(4.2.3e)}\\
& C_j \geq C_{ij} && \text{for all } i \in M, j \in N. && \text{(4.2.3f)}
\end{aligned}
$$

For a given machine i, the set of vectors defined by the constraints (4.2.3b)-(4.2.3d) is known to define all permutations of N as described by these δ-variables (the convex hull of this set is known as the *linear ordering polytope*). It follows that the integer program (4.2.3a)-(4.2.3f) is a correct formulation of PD $| \, | \sum w_j C_j$.

Wagneur and Sriskandarajah (1993) showed that for PD $| \, | \sum w_j C_j$, there always exists an optimal schedule in which each machine processes the jobs in the same order. This kind of schedule is known as a *permutation schedule*. Therefore, we only need to find one common ordering of the jobs to find an optimal solution. As we did before, let us define the decision variables δ_{jk}, where $\delta_{jk} = 1$ if job j precedes job k, and $\delta_{jk} = 0$ otherwise. Consider the following integer programming formulation for PD $| \, | \sum w_j C_j$, now with only one set of linear ordering constraints:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j \in N} w_j C_j && && \text{(4.2.4a)}\\
\text{subject to} \quad & \delta_{jk} + \delta_{kj} = 1 && \text{for all } j,k \in N : j \neq k, && \text{(4.2.4b)}\\
& \delta_{jk} + \delta_{kl} + \delta_{lj} \leq 2 && \text{for all } j,k,l \in N : j \neq k \neq l \neq j, && \text{(4.2.4c)}\\
& \delta_{jk} \in \{0,1\} && \text{for all } j,k \in N : j \neq k, && \text{(4.2.4d)}\\
& C_j \geq \sum_{k \in N : k \neq j} p_{ik} \delta_{kj} + p_{ij} && \text{for all } i \in M, j \in N. && \text{(4.2.4e)}
\end{aligned}
$$

By the observation of Wagneur and Sriskandarajah (1993), it follows that the above integer programming formulation is also valid for PD $| \, | \sum w_j C_j$.

We consider the following linear programming relaxation of the integer program (4.2.3a)-

(4.2.3f), obtained by replacing the binary constraints with nonnegativity constraints:

$$\begin{array}{lll}
\text{[LO-1]} & \text{minimize} & (4.2.3a) \\
& \text{subject to} & (4.2.3b), (4.2.3c), (4.2.3e), (4.2.3f), \\
& & \delta^i_{jk} \geq 0 \quad \text{for all } i \in M \text{ and } j,k \in N : j \neq k.
\end{array}$$

We also consider the following linear programming relaxation of (4.2.4a)-(4.2.4e), obtained similarly:

$$\begin{array}{lll}
\text{[LO-2]} & \text{minimize} & (4.2.4a) \\
& \text{subject to} & (4.2.4b), (4.2.4c), (4.2.4e), \\
& & \delta_{jk} \geq 0 \quad \text{for all } j,k \in N : j \neq k.
\end{array}$$

4.2.3 Relative strength of LP relaxations

For any linear programming relaxation [X] of PD$\,||\sum w_j C_j$, let LP$_{[\text{X}]}$ be the optimal value of [X]. We show the following statement on the relative strength of the four linear programming relaxations presented above.

Lemma 4.2.4. *For any given instance of PD$\,||\sum w_j C_j$, we have that*

$$\text{LP}_{[\text{CT}-1]} = \text{LP}_{[\text{CT}-2]} \leq \text{LP}_{[\text{LO}-1]} \leq \text{LP}_{[\text{LO}-2]}.$$

Proof. Fix an instance of PD$\,||\sum w_j C_j$. Let $((C_{ij})_{i\in M, j\in N}, (C_j)_{j\in N})$ be an optimal solution to [CT-1], and let $(C'_j)_{j\in N}$ be an optimal solution to [CT-2]. Clearly, $(C_j)_{j\in N}$ is feasible in [CT-2], and so LP$_{[\text{CT}-2]} \leq$ LP$_{[\text{CT}-1]}$. Now define $C'_{ij} = C'_j$ for all $i \in M$ and $j \in N$. Clearly, $((C'_{ij})_{i\in M, j\in N}, (C'_j)_{j\in N})$ is feasible in [CT-1], and so LP$_{[\text{CT}-1]} \leq$ LP$_{[\text{CT}-2]}$. Therefore, LP$_{[\text{CT}-1]} =$ LP$_{[\text{CT}-2]}$.

Now suppose $((\bar{\delta}^i_{jk})_{i\in M, j,k\in N: j\neq k}, (\bar{C}_{ij})_{i\in M, j\in N}, (\bar{C}_j)_{j\in N})$ is an optimal solution to [LO-1]. Using techniques from Schulz (1996), it is straightforward to show that $(\bar{C}_j)_{j\in N}$ is feasible in [CT-2], and so LP$_{[\text{CT}-2]} \leq$ LP$_{[\text{LO}-1]}$.

Finally, suppose $((\hat{\delta}_{jk})_{j,k\in N: j\neq k}, (\hat{C}_j)_{j\in N})$ is an optimal solution to [LO-2]. Define $\hat{\delta}^i_{jk} = \hat{\delta}_{jk}$ for all $i \in M$ and $j,k \in N$ such that $j \neq k$. Also, define $\hat{C}_{ij} = \hat{C}_j$ for all $i \in M$ and $j \in N$. Clearly, $((\hat{\delta}^i_{jk})_{i\in M, j,k\in N: j\neq k}, (\hat{C}_{ij})_{i\in M, j\in N}, (\hat{C}_j)_{j\in N})$ is a feasible solution to [LO-1], and so LP$_{[\text{LO}-1]} \leq$ LP$_{[\text{LO}-2]}$. □

The following example shows that the last inequality in Lemma 4.2.4 can be strict.

Example 4.2.5. In this example, we provide an instance for which LP$_{[\text{LO}-1]} <$ LP$_{[\text{LO}-2]}$. Consider the following instance with $m = 2$ and $n = 2$:

j	1	2
w_j	1	1
p_{1j}	1	4
p_{2j}	2	2

The optimal objective value of [LO-1] is 6.75, and the optimal objective value of [LO-2] is 7. □

4.2.4 Integrality gaps for LP relaxations

Chen and Hall (2001), Leung et al. (2007), and Garg et al. (2007) independently observed that scheduling jobs in order of nondecreasing optimal C_j to the linear program [CT-1] is a 2-approximation algorithm for the problem $\mathrm{PD}\,|\,|\sum w_j C_j$. The proof technique used to show this implies that [CT-1] is in fact a 2-relaxation of $\mathrm{PD}\,|\,|\sum w_j C_j$; that is, the integrality gap[1] of [CT-1] is at most 2. Similar proof techniques also show that scheduling jobs in order of nondecreasing optimal C_j to the linear programs [CT-2], [LO-1], and [LO-2] are also 2-approximation algorithms, and that these linear programs are all 2-relaxations. In this subsection, we present a proof showing that the analyses of these LP relaxations are tight: the integrality gap is 2 for [CT-1], [CT-2], [LO-1], and [LO-2].

Theorem 4.2.6 (Mastrolilli 2008)**.** *The integrality gap is* 2 *for the following linear programming relaxations:* [CT-1], [CT-2], [LO-1] *and* [LO-2].

Proof. We show that the integrality gap of [CT-1], [CT-2], [LO-1], and [LO-2] is at least 2.

Let (N, E) be a complete r-uniform hypergraph on N (in other words, let E the family of all $\binom{n}{r}$ subsets of N with cardinality r). We construct an instance of $\mathrm{PD}\,|\,|\sum w_j C_j$ as follows. Each node $j \in N$ corresponds to a job. Each hyperedge $i \in E$ corresponds to a machine, so $m = \binom{n}{r}$. The processing times are

$$p_{ij} = \begin{cases} 1 & \text{if } j \in \text{hyperedge } i, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in M,\ j \in N.$$

All jobs have unit weight. Note that every machine needs to process jobs for exactly r time units.

We first show that in any feasible schedule without idle time, there are at least $n - r + 1$ jobs that complete at time r. We consider two cases.

1. *There are at most* $r - 2$ *jobs that complete at or before time* $r - 1$. Therefore, at least $n - r + 2$ jobs complete at time r, which directly implies the claim.
2. *There are at least* $r - 1$ *jobs that complete at or before time* $r - 1$. Let A be a set of $r - 1$ jobs that completes at or before time $r - 1$. Since (N, E) is a complete r-uniform hypergraph, for any job $j \in N \setminus A$, we have that $A \cup \{j\}$ is a hyperedge in (N, E). Since there are $r - 1$ jobs in A, this implies that every job $j \in N \setminus A$ cannot complete until at least time r on the machine corresponding to the hyperedge $A \cup \{j\}$. Since $|N \setminus A| = n - r + 1$, there are at least $n - r + 1$ jobs that complete at time r.

[1] In this chapter, we slightly abuse terminology: for a minimization problem Π, let $\mathrm{OPT}(I)$ denote the optimal value of the problem Π under instance I, and let $\mathrm{OPT}_R(I)$ denote the optimal value of the relaxation R of problem Π under instance I. We say that the integrality gap of the relaxation R of problem Π is $\sup\{\mathrm{OPT}(I)/\mathrm{OPT}_R(I) : I \text{ is an instance of } \Pi\}$.

Let OPT denote the optimal value of this instance. It follows from the above observation that OPT $\geq r(n - r + 1)$. Now consider the following solution to [LO-2]:

$$\begin{aligned} \delta_{jk} &= 1/2 && \text{for all } j, k \in N : j \neq k, \\ C_j &= \max_{i \in M} \left\{ \sum_{k \in N : k \neq j} p_{ik} \delta_{kj} + p_{ij} \right\} && \text{for all } j \in N. \end{aligned}$$

It is straightforward to show that this solution is feasible. Also, note that $C_j \leq (r-1)/2 + 1$, and so $\mathrm{LP}_{[\mathrm{LO-2}]} \leq n(r+1)/2$. Letting $r = n^{3/4}$, we have that

$$\frac{\mathrm{OPT}}{\mathrm{LP}_{[\mathrm{LO-2}]}} \geq \frac{2n^{3/4}(n - n^{3/4} + 1)}{n(n^{3/4} + 1)},$$

which approaches 2 as n goes to infinity.

By Lemma 4.2.4, the result follows for the other three LP relaxations. □

4.3 A combinatorial primal-dual 2-approximation algorithm

In this section, we present a simple primal-dual 2-approximation algorithm for PD $| \, | \sum w_j C_j$. Independently, Queyranne (2008) proposed essentially the same algorithm, framed as a greedy algorithm, and also showed that it has a performance guarantee of 2. Unlike the LP-based approximation algorithms discussed in Section 4.2.4, our algorithm does not require the solution of a linear program; in fact, our algorithm runs in $O(n(m+n))$ time. Although it does not require solving the linear program [CT-2], we use this linear program and its dual in the analysis of our algorithm. Note that the dual of [CT-2] is

$$\begin{aligned} \text{maximize} \quad & \sum_{i \in M} \sum_{S \subseteq N} f_i(S) y(i, S) && && (4.3.1a) \\ \text{subject to} \quad & \sum_{i \in M} p_{ij} \sum_{S \subseteq N : j \in S} y(i, S) = w_j && \text{for all } j \in N, && (4.3.1b) \\ & y(i, S) \geq 0 && \text{for all } i \in M, S \subseteq N. && (4.3.1c) \end{aligned}$$

Our algorithm works as follows. We find a permutation schedule by working in reverse. We determine the *last* job to be scheduled by observing that its completion time is achieved on the machine with the maximum load when all jobs are scheduled; we choose the job with the minimum weight-to-processing time ratio on that machine. We adjust the dual variables as well as the weights of the jobs, and we proceed in determining the next-to-last job in a similar manner. A full description of the algorithm is below.

Algorithm 4.3.1. Primal-dual algorithm for PD $| \, | \sum w_j C_j$

Input: instance of PD $| \, | \sum w_j C_j$: number of jobs n; number of machines m; processing times p_{ij} for all $i \in M$ and $j \in N$; weights w_j for all $j \in N$.

Output: permutation schedule of jobs $\sigma : \{1, \dots, n\} \mapsto N$.

```
1. Initialize:

   $C_j \leftarrow +\infty$                     for all $j \in N$,                   (completion times)
   $y(i,S) \leftarrow 0$                        for all $i \in M$, $S \subseteq N$,  (dual variables)
   $S^k \leftarrow N$                           for all $k = 1, \dots, n$,           (unscheduled jobs at iteration $k$)
   $L_i^k \leftarrow \sum_{j \in N} p_{ij}$     for all $k = 1, \dots, n$, $i \in M$, (load of machine $i$ at iteration $k$)
   $\bar{w}_j^k \leftarrow w_j$                 for all $k = 1, \dots, n$, $j \in N$. (adjusted weights)

2. For $k = n, n-1, \dots, 2, 1$ :

   $\mu(k) \leftarrow \arg\max_{i \in M} L_i^k$         (determine the machine on which job $\sigma(k)$ completes)
   $\sigma(k) \leftarrow \arg\min_{j \in S^k} \frac{\bar{w}_j^k}{p_{\mu(k),j}}$   (determine job $\sigma(k)$)
   $C_{\sigma(k)} \leftarrow L_{\mu(k)}^k$               (completion time of job $\sigma(k)$)
   $y(\mu(k), S^k) \leftarrow \frac{\bar{w}_{\sigma(k)}^k}{p_{\mu(k),\sigma(k)}}$   (update dual variables)
   $S^{k-1} \leftarrow S^k \setminus \{\sigma(k)\}$      (update unscheduled jobs)
   for all $i \in M$: $L_i^{k-1} \leftarrow L_i^k - p_{i,\sigma(k)}$   (update machine loads)
   for all $j \in S^{k-1}$: $\bar{w}_j^{k-1} \leftarrow \bar{w}_j^k - p_{\mu(k),j}\, y(\mu(k), S^k)$   (adjust weights)

   When computing $\mu(k)$ and $\sigma(k)$, break ties arbitrarily.
```

To show the performance guarantee of Algorithm 4.3.1, we need the following useful property of the set function f_i, first proved by Schulz (1996) in the context of completion-time-variable LP relaxations for other scheduling problems.

Lemma 4.3.2 (Schulz 1996)**.** *For any $i \in M$, and $S \subseteq N$, we have that*

$$\Big(\sum_{j \in S} p_{ij}\Big)^2 \le \Big(2 - \frac{2}{n+1}\Big) f_i(S).$$

Proof. Note that

$$\begin{aligned}\Big(\sum_{j \in S} p_{ij}\Big)^2 &= 2\bigg(\frac{1}{2}\Big(\sum_{j \in S} p_{ij}\Big)^2 + \frac{1}{2}\sum_{j \in S} p_{ij}^2\bigg) - \sum_{j \in S} p_{ij}^2 \\ &= 2 f_i(S) - \frac{2\sum_{j \in S} p_{ij}^2}{\big(\sum_{j \in S} p_{ij}\big)^2 + \sum_{j \in S} p_{ij}^2} f_i(S)\end{aligned}$$

$$\leq \left(2 - \frac{2}{|S|+1}\right) f_i(S)$$
$$\leq \left(2 - \frac{2}{n+1}\right) f_i(S),$$

which proves the claim. □

Now we show the main result of this section.

Theorem 4.3.3. *Algorithm 4.3.1 is a* $\left(2 - \frac{2}{n+1}\right)$*-approximation algorithm for* $PD\,||\,\sum w_j C_j$.

Proof. First, we show that the vector $y = (y(i,S))_{i\in M, S\subseteq N}$ computed by the algorithm is a feasible solution to the dual linear program (4.3.1a)-(4.3.1c). Note that the algorithm assigns non-zero values to at most n components of y: $y(\mu(k), S^k)$ for $k = 1,\dots,n$. Since $\sigma(k) \in S^k \subseteq S^{k+1}$ for $k = 1,\dots,n-1$, we have

$$\begin{aligned}
\bar{w}^k_{\sigma(k)} &= \bar{w}^{k+1}_{\sigma(k)} - p_{\mu(k+1),\sigma(k)} y(\mu(k+1), S^{k+1}) \\
&= \bar{w}^{k+1}_{\sigma(k)} - p_{\mu(k+1),\sigma(k)} \min_{j\in S^{k+1}} \frac{\bar{w}^{k+1}_j}{p_{\mu(k+1),j}} \\
&\geq 0
\end{aligned}$$

for $k = 1,\dots,n-1$. We also have that $\bar{w}^n_{\sigma(n)} = w_{\sigma(n)} \geq 0$. It follows that $y(i,S) \geq 0$ for all $i \in M$ and $S \subseteq N$. In addition, constraints (4.3.1b) are satisfied: for job $\sigma(k)$, we have

$$\begin{aligned}
\sum_{i\in M} p_{i,\sigma(k)} \sum_{S\subseteq N:\sigma(k)\in S} y(i,S) &= \sum_{l=k}^{n} p_{\mu(l),\sigma(k)} y(\mu(l), S^l) \\
&= \sum_{l=k+1}^{n} p_{\mu(l),\sigma(k)} y(\mu(l), S^l) + p_{\mu(k),\sigma(k)} y(\mu(k), S^k) \\
&= w_{\sigma(k)} - \bar{w}^k_{\sigma(k)} + \bar{w}^k_{\sigma(k)} \\
&= w_{\sigma(k)}.
\end{aligned}$$

Next, note that by how $\mu(k)$ is chosen for $k = 1,\dots,n$, we know that $(C_j)_{j\in N}$ is the completion time vector of the jobs scheduled according to the permutation σ computed by the algorithm.

We now show that the schedule constructed by the algorithm is a $(2 - 2/(n+1))$-approximation. Note that by construction, $L^k_i = \sum_{j\in S^k} p_{ij}$ for all $k = 1,\dots,n$ and $i \in M$. For the remainder of this proof, we assume without loss of generality that $\sigma(k) = k$ for $k = 1,\dots,n$. By this assumption, we have that $S^k = \{1,\dots,k\}$ for $k = 1,\dots,n$, and $C_1 \leq C_2 \leq \dots \leq C_n$. Let $(C^{\mathrm{LP}}_j)_{j\in N}$ be an optimal solution to [CT-2], and let $(C^*_j)_{j\in N}$ be an optimal completion time vector. We have that

$$\sum_{j=1}^{n} w_j C_j = \sum_{j=1}^{n} \left(\sum_{i\in M} p_{ij} \sum_{S\subseteq N: j\in S} y(i,S) \right) C_j$$

$$
\begin{aligned}
&= \sum_{i\in M}\sum_{S\subseteq N} y(i,S)\sum_{j\in S} p_{ij}C_j \\
&= \sum_{k=1}^{n} y(\mu(k),S^k)\sum_{j\in S_k} p_{\mu(k),j}C_j \\
&= \sum_{k=1}^{n} y(\mu(k),S^k)\sum_{j=1}^{k} p_{\mu(k),j}C_j \\
&\le \sum_{k=1}^{n} y(\mu(k),S^k)\Bigl(C_k\sum_{j=1}^{k} p_{\mu(k),j}\Bigr) && \text{(since } C_k \ge C_j \text{ for all } j = 1,\dots,k) \\
&= \sum_{k=1}^{n} y(\mu(k),S^k)\Bigl(\sum_{j=1}^{k} p_{\mu(k),j}\Bigr)^2 && \text{(since } C_k = L^k_{\mu(k)} = \textstyle\sum_{j=1}^{k} p_{\mu(k),j}) \\
&\le \Bigl(2-\frac{2}{n+1}\Bigr)\sum_{k=1}^{n} y(\mu(k),S^k) f_{\mu(k)}(S^k) \\
&\le \Bigl(2-\frac{2}{n+1}\Bigr)\sum_{j=1}^{n} w_j C_j^{\mathrm{LP}} && \text{(since } y \text{ is dual feasible)} \\
&\le \Bigl(2-\frac{2}{n+1}\Bigr)\sum_{j=1}^{n} w_j C_j^{*}.
\end{aligned}
$$

Finally, we analyze the running time of the algorithm. First, as mentioned above, note that we only need to keep track of n of the dual variables $(y(i,S))_{i\in M,S\subseteq N}$; the exponential representation of the dual variables in the algorithm is for presentation's sake. Also, note that throughout the algorithm, we only need to keep track of the unscheduled jobs, machine loads, and adjusted weights of the current iteration. In other words, in the description of Algorithm 4.3.1, the variables $(S^k)_{k=1,\dots,n}$ can be replaced by a single variable S; for each $i \in M$, the variables $(L_i^k)_{k=1,\dots,n}$ can be replaced by a single variable L_i; for all $j \in N$, the variables $(\bar{w}_j^k)_{k=1,\dots,n}$ can be replaced by a single variable $\bar{w}_j$. The algorithm runs through an initialization and n iterations. With the modifications mentioned above, each step in the initialization of the algorithm takes at most nm operations. Each step in each iteration of the algorithm takes either at most m operations or at most n operations. Therefore, the algorithm runs in $O(n(m+n))$ time. □

The above analysis of Algorithm 4.3.1 is tight, as the following example shows.

Example 4.3.4. In this example, we show that Algorithm 4.3.1 has a performance guarantee of at least $2-2/(n+1)$.

Consider the following instance, with $m = n$, and

$$
p_{ij} = \begin{cases} \frac{n}{i} & \text{if } j \le i, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i = 1,\dots,n \text{ and } j = 1,\dots,n.
$$

All jobs have unit weights. Note that when all jobs are scheduled, the load on all machines is n.

Consider the permutation schedule $(n, n-1, \dots, 2, 1)$. In this case, the completion time of job j on machine i is:

$$C_{ij} = \begin{cases} 0 & \text{if } j \geq i+1, \\ \left(\frac{n}{i}\right)(i-j+1) & \text{otherwise.} \end{cases}$$

It is straightforward to show that the completion time of job j under the permutation schedule $(n, n-1, \dots, 2, 1)$ is

$$C_j = \max_{i=1,\dots,n} C_{ij} = \max_{i=j,\dots,n} \left(\frac{n}{i}\right)(i-j+1) = n-j+1.$$

Therefore, the total completion time under the permutation schedule $(n, n-1, \dots, 2, 1)$ is $\frac{n(n+1)}{2}$.

Suppose that when computing $\mu(k)$ and $\sigma(k)$, the algorithm breaks ties by always choosing the machine or job with the highest index. We show that when using this tiebreaking rule, at iteration k:

- $S^k = \{1, \dots, k\}$.
- The load of machine i is

$$L_i^k = \sum_{j \in S^k} p_{ij} \begin{cases} = n & \text{if } i = 1, \dots, k \\ < n & \text{if } i = k+1, \dots, n \end{cases}$$

  $\Rightarrow \mu(k) = k$.
- $p_{\mu(k),j} = p_{k,j} = n/k$ for all jobs $j \in S^k \Rightarrow \sigma(k) = k$.
- $\bar{w}_j^{k-1} = 0$ for all jobs $j \in S^{k-1}$.

It is straightforward to check that these conditions hold at iteration n; in particular, we have that

$$y(\mu(n), S^n) = \frac{w_{\sigma(n)}}{p_{\mu(n),\sigma(n)}} = \frac{w_n}{p_{nn}} = 1$$
$$\Rightarrow \quad \bar{w}_j^{n-1} = w_j - p_{nj}\, y(\mu(n), S^n) = 1 - 1 \cdot 1 = 0 \quad \text{for all } j \in S^{n-1}.$$

Now suppose that these conditions hold at iteration k. What happens at iteration $k-1$?

- Since $\sigma(k) = k$ (job k is scheduled at iteration k), we have that the set of unscheduled jobs at iteration $k-1$ is $S^{k-1} = \{1, \dots, k-1\}$.
- We consider three cases.
  - Consider machine $i \in \{1, \dots, k-1\}$. Since the load of machine i at iteration k is $\sum_{j \in S^k} p_{ij} = n$ and $p_{ik} = 0$ (since $i \leq k-1$), it follows that the load of machine i at iteration $k-1$ is $\sum_{j \in S^{k-1}} p_{ij} = n$.
  - Now, consider machine $i \in \{k+1, \dots, n\}$. Since the load of machine i at

iteration k is $\sum_{j \in S^k} p_{ij} < n$, clearly then the load of machine i at iteration $k-1$ is $\sum_{j \in S^{k-1}} p_{ij} < n$.

  – Finally, consider machine k. The load of machine k at iteration k is $\sum_{j \in S^k} p_{kj} = n$. Since $p_{kk} > 0$, the load of machine k at iteration $k-1$ is $\sum_{j \in S^{k-1}} p_{kj} < n$.

  It follows from above that the tiebreaking rule chooses $\mu(k-1) = k-1$.

- By definition, $p_{\mu(k-1),j} = p_{k-1,j} = n/(k-1)$ for all jobs $j \in S^{k-1} = \{1, \dots, k-1\}$. In addition, all jobs $j \in S^{k-1}$ have weight $\bar{w}_j^{k-1} = 0$. Therefore, $\sigma(k-1) = k-1$.
- The algorithm computes:

$$\begin{aligned} y(\mu(k-1), S^{k-1}) &= \frac{\bar{w}^{k-1}_{\sigma(k-1)}}{p_{\mu(k-1),\sigma(k-1)}} \\ &= \frac{\bar{w}^{k-1}_{k-1}}{p_{k-1,k-1}} \\ &= 0 \\ \Rightarrow \quad \bar{w}_j^{k-2} &= \bar{w}_j^{k-1} - p_{k-1,j}\, y(\mu(k-1), S^{k-1}) \\ &= 0 - p_{k-1,j} \cdot 0 \\ &= 0 \qquad\qquad \text{for all } j \in S^{k-2}. \end{aligned}$$

It follows that the permutation schedule Algorithm 4.3.1 constructs is $(1, \dots, n)$. Since the maximum load of any machine is n at each iteration, it follows that the total completion time under the permutation schedule $(1, \dots, n)$ constructed by the modified greedy algorithm is n^2. As a result, using the objective value of the permutation schedule $(n, n-1, \dots, 2, 1)$ as an upper bound on the optimal value, the performance guarantee of Algorithm 4.3.1 is at least $2 - 2/(n+1)$.

The instance used above can be modified so that all processing times are strictly positive. In particular, changing the instance so the processing times are

$$p_{ij} = \begin{cases} \frac{n-\epsilon(n-i)}{i} & \text{if } j \le i, \\ \epsilon & \text{otherwise} \end{cases} \qquad \text{for all } i = 1, \dots, n \text{ and } j = 1, \dots, n$$

for some sufficiently small $\epsilon > 0$ will still induce similar behavior. □

4.4 Conclusion

In this chapter, we studied the problem of minimizing the sum of weighted completion times in a concurrent open shop environment. We presented some interesting properties of various linear programming relaxations for this problem. We also showed how to obtain a primal-dual 2-approximation algorithm. Although the performance guarantee of our algorithm matches the performance guarantee of the currently best known approximation algorithms for this problem, our algorithm does not require solving a linear program; in fact, it can be viewed as a greedy algorithm.

There are many interesting potential directions for future research that extend from this work. The most natural direction would be to design an approximation algorithm for $\mathrm{PD} \,|\,|\, \sum w_j C_j$ with a performance guarantee better than 2. It also seems plausible that we can design approximation algorithms with performance guarantees better than 2 when the number of machines m is fixed. However, no such results are currently known. One idea is as follows: observe that when $m = 2$, the polyhedron of the linear program [CT-2], on the surface, has some similarities with the intersection of two polymatroids, for which a great deal is known. This gives us some indication that we might be able to uncover a richer theory behind the structure of these polyhedra, which may in turn lead to better approximation algorithms for the case $m = 2$. Along these lines, it stands to reason that we should be able to design an approximation algorithm whose performance guarantee depends on m as well as n, as in some algorithms for other parallel machine scheduling problems. Of course, it may be the case that we actually need to prove stronger inapproximability results for this problem.

Another interesting direction for future research would be to establish structural characterizations of optimal solutions to this problem. Taking inspiration from a result by Correa and Schulz (2005), one may be tempted to conjecture that the tight sets of an optimal LP solution to [CT-2] yield a decomposition of optimal schedules, much like Sidney's (1975) decomposition for $1 \,|\, \mathrm{prec} \,|\, \sum w_j C_j$. For example, if the tight sets of some optimal solution are $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_k$, then one might conjecture that there exists an optimal schedule in which jobs in $S_{i+1} \setminus S_i$ come before $S_{i+2} \setminus S_{i+1}$. However, Example 4.B.1 shows that this is not necessarily the case.

4.A Hardness of approximation

The hardness of approximation results for PD $| \, | \sum w_j C_j$ in this appendix were communicated to us by Ola Svensson, and are due to Uriel Feige and some unknown co-authors.

Using the Unique Games Conjecture[2], Khot and Regev (2003) proved the following theorem on the inapproximability of the maximum cardinality independent set problem.

Theorem 4.A.1 (Khot and Regev 2003)**.** *Assuming the Unique Games Conjecture is true, for any $\delta \in (0, 1/2)$ the following problem is NP-hard: given an undirected graph $G = (N, E)$, decide whether*
(i) *G contains an independent set of size $(\frac{1}{2} - \delta)|N|$, or*
(ii) *all independent sets of G have size strictly less than $\delta|N|$.*

Using the above result, we can show the following.

Theorem 4.A.2. *Assuming the Unique Games Conjecture is true, PD $| \, | \sum w_j C_j$ is hard to approximate within a factor of $4/3 - \epsilon$ for any $\epsilon > 0$, unless P = NP.*

Proof. Let $G = (N, E)$ be an undirected graph. We construct an instance of PD $| \, | \sum w_j C_j$ as follows. Each node $j \in N$ corresponds to a job. Each edge $i \in E$ corresponds to a machine. The processing times are

$$p_{ij} = \begin{cases} 1 & \text{if edge } i \in E \text{ is incident to vertex } j \in N, \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } i \in M,\ j \in N.$$

Note that each machine needs to process jobs for exactly 2 time units.

This is the key observation. Suppose that $I \subseteq N$ is an independent set in G. Then, each job in I can have all its components processed simultaneously. Therefore, all jobs in I can be completed by time 1.

Let OPT denote the optimal value of this instance of PD $| \, | \sum w_j C_j$. Suppose that condition (i) holds from Theorem 4.A.1. Let I be such an independent set. By the observation in the previous paragraph, we know that all jobs in I can be completed by time 1, and that all the remaining jobs $N \setminus I$ can be completed by time 2. Therefore, in this case, $\text{OPT} \leq 1 \cdot (1/2 - \delta)|N| + 2 \cdot (1/2 + \delta)|N| = (3/2 + \delta)|N|$. Now suppose that condition (ii) holds from Theorem 4.A.1. This implies that in any schedule, at least $(1 - \delta)|N|$ jobs are

[2]The *Unique Games Conjecture* (Khot 2002) is a statement on the hardness of the Unique Label Cover problem. In the Unique Label Cover problem, we are given a bipartite graph $(V \cup W, E)$ with $V \cap W = \emptyset$, a set of allowed labels $\{1, \dots, M\}$, and bijective maps $\sigma_{v,w} : \{1, \dots, M\} \mapsto \{1, \dots, M\}$ for every edge $\{v, w\} \in E$. A labeling assigns one label to every vertex of $V \cup W$. A labeling satisfies an edge $\{v, w\} \in E$ if $\sigma_{v,w}(\text{label}(w)) = \text{label}(v)$. The objective is to find a labeling that maximizes the fraction of edges that are satisfied. The Unique Games Conjecture asserts that this problem is hard.

Conjecture (Unique Games Conjecture (Khot 2002))**.** *For any $\eta, \gamma \in \mathbb{R}_{>0}$, there exists a constant $M = M(\eta, \gamma)$ such that it is NP-hard to decide whether the Unique Label Cover problem with label set $\{1, \dots, M\}$ has optimum at least $1 - \eta$ or at most γ.*

The Unique Games Conjecture has been used to obtain inapproximability results for several problems (e.g. Khot 2002; Khot and Regev 2003; Khot et al. 2007).

forced to be completed at time 2. Therefore, in this case, OPT $\geq 2(1-\delta)|N|$. It follows that a $(4/3-\epsilon)$-approximation algorithm for PD $||\sum w_j C_j$ can solve the decision problem in Theorem 4.A.1. $\square$

Dinur and Safra (2002) showed the following hardness of approximation result for the independent set problem, which does not assume the Unique Games Conjecture is true.

Theorem 4.A.3 (Dinur and Safra 2002)**.** *For any $\delta > 0$, the following problem is NP-hard: given an undirected graph $G = (N, E)$, decide whether*

(i) *G contains an independent set of size strictly greater $(\frac{1}{3} - \delta)|N|$, or*

(ii) *all independent sets of G have size strictly less than $(\frac{1}{9} + \delta)|N|$.*

Using the ideas mentioned in the proof of Theorem 4.A.2 in conjunction with Theorem 4.A.3, one can show the following theorem.

Theorem 4.A.4. *PD $||\sum w_j C_j$ is hard to approximate within a factor of $16/15 - \epsilon$ for any $\epsilon > 0$, unless $P = NP$.*

4.B Additional Examples

Example 4.B.1. In this example, we show that the tight sets of an optimal solution to [CT-2] do not necessarily yield an optimal sequence decomposition of jobs. Consider the following instance with $m = 2$ and $n = 2$:

j	1	2
w_j	2	7
p_{1j}	1	4
p_{2j}	2	2

The optimal objective value of [CT-2] is 37.25. The unique optimal solution to [CT-2] is $C_1^{\text{LP}} = 2$, $C_2^{\text{LP}} = 4.75$. The optimal objective value of this instance is 38, which can be achieved by the permutation schedule $(2, 1)$. However, the tight sets of $(C_1^{\text{LP}}, C_2^{\text{LP}})$ imply the permutation schedule $(1, 2)$, which has an objective value of 39. □

Example 4.B.2. In this example, we show that the *simple greedy algorithm* (Algorithm 4.B.3, below) has a performance guarantee of at least 2.

The simple greedy algorithm works similarly to the primal-dual algorithm presented in Section 4.3 (Algorithm 4.3.1), except that the job weights are not modified to compute a dual-feasible solution.

```
Algorithm 4.B.3. Simple greedy algorithm
  Input: instance of PD || $\sum w_j C_j$: number of jobs $n$; number of machines
    $m$; processing times $p_{ij}$ for all $i \in M$ and $j \in N$; weights $w_j$ for all
    $j \in N$.
  Output: permutation schedule of jobs $\sigma : \{1, \dots, n\} \mapsto N$.
  1. Initialize:
       $C_j \leftarrow +\infty$                     for all $j \in N$,                    (completion times)
       $S^k \leftarrow N$                           for all $k = 1, \dots, n$,            (unscheduled jobs at iteration $k$)
       $L_i^k \leftarrow \sum_{j \in N} p_{ij}$     for all $k = 1, \dots, n, i \in M$.   (load of machine $i$ at iteration $k$)
  2. For $k = n, n-1, \dots, 2, 1$:
       $\mu(k) \leftarrow \arg\max_{i \in M} L_i^k$                    (determine the machine on which job $\sigma(k)$ completes)
       $\sigma(k) \leftarrow \arg\min_{j \in S^k} \frac{w_j}{p_{\mu(k),j}}$   (determine job $\sigma(k)$)
       $C_{\sigma(k)} \leftarrow L_{\mu(k)}^k$                        (completion time of job $\sigma(k)$)
       $S^{k-1} \leftarrow S^k \setminus \{\sigma(k)\}$               (update unscheduled jobs)
       for all $i \in M$: $L_i^{k-1} \leftarrow L_i^k - p_{i,\sigma(k)}$   (update machine loads)
  When computing $\mu(k)$ and $\sigma(k)$, break ties arbitrarily.
```

Consider the instance in Example 4.3.4. What does the simple greedy algorithm do with this instance? Suppose that when computing $\mu(k)$ and $\sigma(k)$, the algorithm breaks ties by always choosing the machine or job with the highest index. Using techniques similar to those in Example 4.3.4, we can show that using this tiebreaking rule, that at iteration k:

- $S^k = \{1, \ldots, k\}$.
- The load of machine i is
$$L_i^k = \sum_{j \in S^k} p_{ij} \begin{cases} = n & \text{if } i = 1, \ldots, k \\ < n & \text{if } i = k+1, \ldots, n \end{cases}$$
$\Rightarrow \mu(k) = k$.
- $p_{\mu(k),j} = p_{k,j} = n/k$ for all jobs $j \in S^k \Rightarrow \sigma(k) = k$.

It follows that the permutation schedule the simple greedy algorithm constructs is $(1, \ldots, n)$. Since the maximum load of any machine is n at each iteration, it follows that the total completion time under the permutation schedule $(1, \ldots, n)$ constructed by the simple greedy algorithm is n^2. As a result, using the objective value of the permutation schedule $(n, n-1, \ldots, 2, 1)$ as an upper bound on the optimal value, the performance guarantee of the simple greedy algorithm is at least $2 - 2/(n+1)$. □

Example 4.B.4. In this example, we show that the *m-WSPT algorithm* (Algorithm 4.B.5, below) has a performance guarantee of at least 2.

In the m-WSPT algorithm, we consider the m permutation schedules that correspond to the m permutations determined by WSPT (order the jobs according to nonincreasing weight-to-processing-time ratio) on each of the m machines.

Algorithm 4.B.5. m-WSPT algorithm.

Input: instance of PD | | $\sum w_j C_j$: number of jobs n; number of machines m; processing times p_{ij} for all $i \in M$ and $j \in N$; weights w_j for all $j \in N$.

Output: permutation schedule of jobs $\sigma : \{1, \ldots, n\} \mapsto N$.

1. Compute m permutations, determined by nonincreasing w_j / p_{ij} for each machine $i = 1, \ldots, m$. Break ties arbitrarily.
2. Construct m permutation schedules, each based on one of the m permutations computed in the previous step. Output the permutation schedule with the best objective value.

Consider the following instance, with $r \in \mathbb{Z}_{>0}$, $n = rm$, and

$$p_{ij} = \begin{cases} p & \text{if } j \equiv i \pmod{m}, \\ 1 & \text{otherwise,} \end{cases} \qquad \text{for all } i = 1, \ldots, m \text{ and } j = 1, \ldots, n.$$

where $p \gg 1$. All jobs have unit weight.

Consider the permutation schedule $(1,\ldots,m,m+1,\ldots,2m,\ldots,(r-1)m+1,\ldots,rm)$. We use the objective value of this permutation schedule as an upper bound on the objective value of optimal schedule. The total completion time of this schedule is

$$\begin{aligned}
& & p + (p+1) + \cdots + (p+m-1) \\
&+(p+m-1)m & +p + (p+1) + \cdots + (p+m-1) \\
&+2(p+m-1)m & +p + (p+1) + \cdots + (p+m-1) \\
& & \vdots \\
&+(r-1)(p+m-1)m & +p + (p+1) + \cdots + (p+m-1) \\
=\ & \frac{r(r-1)}{2}\cdot(p+m-1)m & +r\left(pm + \frac{m(m-1)}{2}\right).
\end{aligned} \tag{4.B.1}$$

For $i = 1,\ldots,m$, let $N_i = \{j \in N : p_{ij} = p\}$. Note that $|N_i| = r$ for all $i = 1,\ldots,m$, and that the sets $\{N_i : i = 1,\ldots,m\}$ form a partition of the jobs in N. Fix some machine i, and consider a schedule that is consistent with the ordered partition $(N_{i+1},\ldots,N_m,N_1,\ldots,N_i)$. Note that this is a WSPT schedule for machine i. In the corresponding permutation schedule, the total completion time of jobs on machine $i+1$ (or machine 1, if $i = m$) is

$$\begin{aligned}
&\big(p + 2p + \cdots + rp\big) + \big(rp \cdot r(m-1) + 1 + \cdots + r(m-1)\big) \\
&\qquad = \frac{pr(r+1)}{2} + pr^2(m-1) + \frac{r(m-1)\big(r(m-1)+1\big)}{2}.
\end{aligned} \tag{4.B.2}$$

Therefore, the total completion time of the permutation schedule output by the m-WSPT algorithm using the WSPT sequences prescribed above is at least (4.B.2).

Setting $p = r = m^2$, we have that the performance ratio of the m-WSPT algorithm is at least

$$\frac{m^7 - m^5 + m^4 + \frac{1}{2}m^3 - \frac{1}{2}m^2}{\frac{1}{2}m^7 + \frac{1}{2}m^6},$$

which approaches 2 as m tends to infinity. $\square$

Appendix A

Review of Essential Background

In this appendix, we point out some references that may be useful to the reader, and review some basic terminology related to approximation algorithms and machine scheduling that is used throughout this thesis.

We refer the reader to the textbooks by Schrijver (1986), Nemhauser and Wolsey (1988), Grötschel et al. (1988), Bertsimas and Tsitsiklis (1997), Korte and Vygen (2002), and Schrijver (2003) for comprehensive treatments of topics in linear and integer programming and combinatorial optimization. Garey and Johnson (1978) and Ausiello et al. (1999) provide useful introductions to computational complexity and the theory of NP-completeness.

A.1 Approximation algorithms

For NP-hard optimization problems, it is unlikely that we can design exact algorithms that run in polynomial time. In this light, a natural question that arises is whether we can obtain approximately good solutions in polynomial time. We define some terms from the literature on approximation algorithms; for an in-depth survey of this topic, see Hochbaum (1997), Ausiello et al. (1999), or Vazirani (2001).

A *ρ-approximation algorithm* ($\rho \geq 1$) for an optimization problem Π is an algorithm that finds a solution whose objective value is within a factor of ρ of the optimal value, and whose running time is polynomial in the size of the input to Π. That is, if $\mathrm{OPT}(I)$ is the optimal value of the problem Π under instance I, and if $\mathrm{A}(I)$ is the value of the solution returned by a ρ-approximation algorithm for Π under instance I, then,

$$\begin{aligned} &\mathrm{A}(I) \leq \rho \cdot \mathrm{OPT}(I) && \text{for minimization problems, or} \\ &\mathrm{A}(I) \geq \frac{1}{\rho} \cdot \mathrm{OPT}(I) && \text{for maximization problems.} \end{aligned}$$

The parameter ρ is called the *performance guarantee* of the algorithm. The closer the performance guarantee is to 1, the better the algorithm.

A *polynomial-time approximation scheme* (PTAS) for an optimization problem Π is an algorithm that, for any given $\epsilon > 0$, finds a solution whose objective value is within a factor $(1 + \epsilon)$ of the optimal value, and whose running time is polynomial in the size of the input to Π. Note that the running time of a PTAS can be exponential in $1/\epsilon$.

A *fully polynomial-time approximation scheme* (FPTAS) for an optimization problem Π is an algorithm that, for any given $\epsilon > 0$, finds a solution whose objective value is within a factor $(1 + \epsilon)$ of the optimal value, and whose running time is polynomial in the size of the input to Π, and $1/\epsilon$.

A.2 Notation for machine scheduling

The number of different types of machine scheduling problems that have been studied in the literature is tremendous. As such, it is convenient to describe machine scheduling problems using the three-field notation of Graham et al. (1979), in which the features of a scheduling problem is captured in the three-field abbreviation $\alpha \mid \beta \mid \gamma$.

- The field α represents the machine environment. Some common examples include

1	for a single machine.
P	for identical parallel machines.
Pm	for m identical parallel machines, where m is fixed.
Q	for uniform parallel machines: job j has processing requirement p_j and machine i has speed s_i; the processing time of job j on machine i is p_j/s_i.
PD	for concurrent open shop environments: m parallel machines, each dedicated to one component type; jobs consist of processing requirements for each of the m component types, which can be performed in parallel.

- The field β describes job characteristics. For instance,

$p_j = 1$	indicates that all jobs have unit processing time.
r_j	indicates that jobs have release dates.
pmtn	indicates that preemption of jobs is allowed.
prec	indicates that there are precedence constraints between jobs.

- The field γ denotes the objective function to be minimized. For example,

$\sum w_j C_j$	refers to the sum of weighted completion times objective.
$\sum C_j$	refers to the sum of completion times (unit weights) objective.
$\sum w_j M_j$	refers to the sum of weighted mean busy times objective.
$C_{\max}$	refers to the makespan objective.

So for example, $\mathrm{P} \mid r_j, \mathrm{prec} \mid \sum w_j C_j$ is the problem of minimizing the sum of weighted completion times on identical parallel machines, subject to release dates and precedence constraints.

Bibliography

R. H. Ahmadi, U. Bagchi. 1990. Scheduling of multi-job customer orders in multi-machine environments. ORSA/TIMS, Philadelphia.

R. H. Ahmadi, U. Bagchi, T. Roemer. 2005. Coordinated scheduling of customer orders for quick response. *Naval Research Logistics* 52:493–512.

C. Ambühl, M. Mastrolilli. 2006. Single machine precedence constrained scheduling is a vertex cover problem. In *Proceedings of the 14th Annual European Symposium on Algorithms (ESA 2006)*, vol. 4168 of *Lecture Notes in Computer Science*, pp. 28–39. Springer, Berlin.

C. Ambühl, M. Mastrolilli, N. Mutsanas, O. Svensson. 2007a. Scheduling with precedence constraints of low fractional dimension. In M. Fischetti, D. P. Williamson, eds., *Proceedings of the 12th Conference on Integer Programming and Combinatorial Optimization (IPCO 2007)*, vol. 4513 of *Lecture Notes in Computer Science*, pp. 130–144. Springer, Berlin.

C. Ambühl, M. Mastrolilli, O. Svensson. 2006. Approximating precedence-constrained single machine scheduling by coloring. In J. Diaz, K. Jansen, J. D. P. Rolim, U. Zwick, eds., *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX-RANDOM 2006)*, vol. 4110 of *Lecture Notes in Computer Science*, pp. 15–26. Springer, Berlin.

C. Ambühl, M. Mastrolilli, O. Svensson. 2007b. Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pp. 329–337.

G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi. 1999. *Complexity and Approximation: Combinatorial Optimization Problems and their Approximability Properties*. Springer, Berlin.

D. Bertsimas, J. N. Tsitsiklis. 1997. *Introduction to Linear Optimization*. Athena Scientific, Belmont.

G. Birkhoff. 1935. On the structure of abstract algebras. In *Proceedings of the Cambridge Philosophical Society*, vol. 31, pp. 433–454.

B. Bollobás. 2001. *Random Graphs*. Cambridge University Press, Cambridge, 2nd edition.

J. Bruno, E. G. Coffman, R. Sethi. 1974. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM* 17:382–387.

C. Chekuri, R. Motwani. 1999. Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. *Discrete Applied Mathematics* 98:29–38.

X. Chen, J. Zhang. 2006. Duality approaches to economic lot sizing games. Manuscript.

Z.-L. Chen, N. G. Hall. 2001. Supply chain scheduling: assembly systems. Working paper, Department of Systems Engineering, University of Pennsylvania.

F. A. Chudak, D. S. Hochbaum. 1999. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters* 25:199–204.

J. R. Correa, A. S. Schulz. 2005. Single-machine scheduling with precedence constraints. *Mathematics of Operations Research* 30:1005–1021.

I. Curiel, G. Pederzoli, S. Tijs. 1989. Sequencing games. *European Journal of Operational Research* 40:344–351.

I. Dinur, S. Safra. 2002. The importance of being biased. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pp. 33–42.

W. L. Eastman, S. Even, I. M. Isaacs. 1964. Bounds for the optimal scheduling of n jobs on m processors. *Management Science* 11:268–279.

J. Edmonds. 1970. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, J. Schönheim, eds., *Combinatorial Structures and Their Applications (Calgary International Conference on Combinatorial Structures and Their Applications)*, pp. 69–87.

J. Edmonds. 1971. Matroids and the greedy algorithm. *Mathematical Programming* 1:127–136.

E. Einy, R. Holzman, D. Monderer. 1999. On the least core and the Mas-Colell Bargaining set. *Games and Economic Behavior* 28:181–188.

E. Einy, D. Monderer, D. Moreno. 1998. The least core, kernel and bargaining sets of large games. *Economic Theory* 11:585–601.

P. Erdös, A. Rényi. 1959. On random graphs. *Publicationes Mathematicae* 6:290–297.

U. Faigle, S. P. Fekete, W. Hochstättler, W. Kern. 1998. On approximately fair cost allocation for Euclidean TSP games. *OR Spektrum* 20:29–37.

U. Faigle, W. Kern. 1993. On some approximately balanced combinatorial cooperative games. *Methods and Models of Operations Research* 38:141–152.

U. Faigle, W. Kern. 1998. Approximate core allocation for binpacking games. *SIAM Journal on Discrete Mathematics* 11:387–399.

U. Faigle, W. Kern, J. Kuipers. 2001. On the computation of the nucleolus of a cooperative game. *International Journal of Game Theory* 30:79–98.

U. Faigle, W. Kern, D. Paulusma. 2000. Note on the computational complexity of least core concepts for min-cost spanning tree games. *Mathematical Methods of Operations Research* 52:23–38.

U. Feige, V. Mirrokni, J. Vondrák. 2007. Maximizing non-monotone submodular functions. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007)*, pp. 461–471.

L. Fleischer, M. X. Goemans, V. S. Mirrokni, M. Sviridenko. 2006. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006)*, pp. 611–620.

H. Gabow, R. Tarjan. 1984. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms* 5:80–131.

M. R. Garey, D. S. Johnson. 1978. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, N.Y.

M. R. Garey, D. S. Johnson, L. Stockmeyer. 1976. Some simplified NP-complete graph problems. *Theoretical Computer Science* 1:237–267.

N. Garg, A. Kumar, V. Pandit. 2007. Order scheduling models: hardness and algorithms. In V. Arvind, S. Prasad, eds., *Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2007)*, vol. 4855 of *Lecture Notes in Computer Science*, pp. 96–107. Springer, Berlin.

Y. Gerchak, D. Gupta. 1991. On apportioning costs to customers in centralized continuous review inventory systems. *Journal of Operations Management* 10:546–551.

D. B. Gillies. 1959. Solutions to general non-zero-sum games. In A. W. Tucker, R. D. Luce, eds., *Contributions to the Theory of Games, Volume IV*, vol. 40 of *Annals of Mathematics Studies*, pp. 47–85. Princeton University Press, Princeton.

M. X. Goemans, M. Queyranne, A. S. Schulz, M. Skutella, Y. Wang. 2002. Single machine scheduling with release dates. *SIAM Journal on Discrete Mathematics* 15:165–192.

M. X. Goemans, M. Skutella. 2004. Cooperative facility location games. *Journal of Algorithms* 50:194–214.

M. X. Goemans, D. P. Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM* 42:1115–1145.

M. X. Goemans, D. P. Williamson. 2000. Two-dimensional Gantt charts and a scheduling algorithm of Lawler. *SIAM Journal on Discrete Mathematics* 13:281–294.

R. L. Graham, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* 5:287–326.

D. Granot, G. Huberman. 1981. Minimum cost spanning tree games. *Mathematical Programming* 21:1–18.

M. Grötschel, L. Lovász, A. Schrijver. 1988. *Geometric Algorithms and Combinatorial Optimization*. Springer.

L. A. Hall, A. S. Schulz, D. B. Shmoys, J. Wein. 1997. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. *Mathematics of Operations Research* 22:513–544.

P. Hall. 1935. On representatives of subsets. *Journal of the London Mathematical Society* 10:26–30.

B. Hartman, M. Dror, M. Shaked. 2000. Cores of inventory centralization games. *Games and Economic Behavior* 31:26–49.

J. Håstad. 2001. Some optimal inapproximability results. *Journal of the ACM* 48:798–859.

D. S. Hochbaum. 1983. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Applied Mathematics* 6:243–254.

D. S. Hochbaum, ed. 1997. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston.

N. Immorlica, M. Mahdian, V. Mirrokni. 2005. Limitations of cross-monotonic cost sharing schemes. In *Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA 2005)*, pp. 602–611.

K. Jain, M. Mahdian, M. Salavatipour. 2003. Packing Steiner trees. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003)*, pp. 266–274.

K. Jansen. 2003. Approximate strong separation with application in fractional graph coloring and preemptive scheduling. *Theoretical Computer Science* 302:239–256.

W. Kern, D. Paulusma. 2003. Matching games: the least core and the nucleolus. *Mathematics of Operations Research* 28:294–308.

S. Khot. 2002. On the power of unique 2-prover 1-round games. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pp. 767–775.

S. Khot, G. Kindler, E. Mossel, R. O'Donnell. 2007. Optimal inapproximability results for MAX-CUT and other 2-variable CSPs? *SIAM Journal on Computing* 37:319–357.

S. Khot, O. Regev. 2003. Vertex cover might be hard to approximate within $2 - \epsilon$. In *Proceedings of the 18th IEEE Conference on Computational Complexity*, pp. 379–386.

B. Korte, J. Vygen. 2002. *Combinatorial Optimization: Theory and Algorithms*, vol. 21 of *Algorithms and Combinatorics*. Springer, Berlin, 2nd edition.

E. L. Lawler. 1978. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Annals of Discrete Mathematics* 2:75–90.

E. Lehrer. 2002. Allocations processes in cooperative games. *International Journal of Game Theory* 31:341–351.

J. K. Lenstra, A. H. G. Rinnooy Kan. 1978. Complexity of scheduling under precedence constraints. *Operations Research* 26:22–35.

J. Y.-T. Leung, H. Li, M. Pinedo. 2007. Scheduling orders for multiple product types to minimize total weighted completion time. *Discrete Applied Mathematics* 155:945–970.

J. Y.-T. Leung, H. Li, M. L. Pinedo. 2005. Order scheduling in an environment with dedicated resources in parallel. *Journal of Scheduling* 8:355–386.

F. Maniquet. 2003. A characterization of the Shapley value in queueing problems. *Journal of Economic Theory* 109:90–103.

F. Margot, M. Queyranne, Y. Wang. 2003. Decompositions, network flows, and a precedence constrained single-machine scheduling problem. *Operations Research* 51:981–992.

M. Maschler, B. Peleg, L. S. Shapley. 1979. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research* 4:303–338.

M. Mastrolilli. 2008. Private communication.

D. Mishra, B. Rangarajan. 2005. Cost sharing in a job scheduling problem using the Shapley value. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, pp. 232–239.

H. Nagamochi, D.-Z. Zeng, N. Kabutoya, T. Ibaraki. 1997. Complexity of the minimum base game on matroids. *Mathematics of Operations Research* 22:146–164.

G. L. Nemhauser, L. E. Trotter. 1975. Vertex packings: structural properties and algorithms. *Mathematical Programming* 8:232–248.

G. L. Nemhauser, L. A. Wolsey. 1988. *Integer and Combinatorial Optimization*. Wiley, New York, N.Y.

G. L. Nemhauser, L. A. Wolsey, M. L. . Fisher. 1978. An analysis of algorithms for maximizing submodular set functions–I. *Mathematical Programming* 14:265–294.

G. Owen. 1975. On the core of linear production games. *Mathematical Programming* 9:358–370.

M. Pál, E. Tardos. 2003. Group strategyproof mechanisms via primal-dual algorithms. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2003)*, pp. 584–593.

B. Peleg, J. Rosenmüller. 1992. The least core, nucleolus, and kernel of homogeneous weighted majority games. *Games and Economic Behavior* 4:588–605.

B. Peleg, P. Sudhölter. 2007. *Introduction to the Theory of Cooperative Games*. Springer, Berlin, 2nd edition.

J. Potters, I. Curiel, S. Tijs. 1991. Traveling salesman games. *Mathematical Programming* 53:199–211.

C. N. Potts. 1980. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Studies* 13:78–87.

W. R. Pulleyblank. 1979. Minimum node covers and 2-bicritical graphs. *Mathematical Programming* 17:91–103.

M. Queyranne. 1993. Structure of a simple scheduling polyhedron. *Mathematical Programming* 58:263–285.

M. Queyranne. 2008. Private communication.

M. Queyranne, A. S. Schulz. 1995. Scheduling unit jobs with compatible release dates on parallel machines with nonstationary speeds. In E. Balas, J. Clausen, eds., *Integer Programming and Combinatorial Optimization (IPCO 1995)*, vol. 920 of *Lecture Notes in Computer Science*, pp. 307–320. Springer, Berlin.

M. Queyranne, A. S. Schulz. 2006. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. *SIAM Journal on Computing* 35:1241–1253.

M. Queyranne, Y. Wang. 1991. Single-machine scheduling polyhedra with precedence constraints. *Mathematics of Operations Research* 16:1–20.

R. Rado. 1957. Note on independence functions. In *Proceedings of the London Mathematical Society*, vol. 7, pp. 300–320.

H. Reijnierse, M. Maschler, J. Potters, S. Tijs. 1996. Simple flow games. *Games and Economic Behavior* 16:238–260.

T. A. Roemer. 2006. A note on the complexity of the concurrent open shop problem. *Journal of Scheduling* 9:389–396.

A. E. Roth, ed. 1988. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*. Cambridge University Press.

S. Sahni. 1976. Algorithms for scheduling independent tasks. *Journal of the ACM* 23:116–127.

D. Schmeidler. 1969. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics* 17:1163–1170.

A. Schrijver. 1986. *Theory of Linear and Integer Programming*. Wiley, Chichester.

A. Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, Berlin.

A. S. Schulz. 1996. Scheduling to minimize total weighted completion time: performance guarantees of LP-based heuristics and lower bounds. In W. H. Cunningham, S. T. McCormick, M. Queyranne, eds., *Integer Programming and Combinatorial Optimization (IPCO 1996)*, vol. 1084 of *Lecture Notes in Computer Science*, pp. 301–315. Springer, Berlin.

A. S. Schulz, N. A. Uhan. 2007. Encouraging cooperation in sharing supermodular costs. In M. Charikar, K. Jansen, O. Reingold, J. D. P. Rolim, eds., *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques (APPROX-RANDOM 2007)*, vol. 4627 of *Lecture Notes in Computer Science*, pp. 271–285. Springer, Berlin.

P. Schuurman, G. J. Woeginger. Approximation schemes - a tutorial. Preliminary version of a chapter for *Lectures on Scheduling*, edited by R. H. Möhring, C. N. Potts, A. S. Schulz, G. J. Woeginger, and L. A. Wolsey.

P. Schuurman, G. J. Woeginger. 1999. Polynomial time approximation algorithms for machine scheduling: ten open problems. *Journal of Scheduling* 2:203–213.

L. S. Shapley. 1953. A value for n-person games. In H. W. Kuhn, A. W. Tucker, eds., *Contributions to the Theory of Games, Volume II*, vol. 28 of *Annals of Mathematics Studies*, pp. 307–317. Princeton University Press, Princeton.

L. S. Shapley. 1971. Cores of convex games. *International Journal of Game Theory* 1:11–26.

L. S. Shapley, M. Shubik. 1966. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica* 34:805–827.

L. S. Shapley, M. Shubik. 1971. The assignment game I: the core. *International Journal of Game Theory* 1:111–130.

J. B. Sidney. 1975. Decomposition algorithms for single-machine sequencing with precedence constraints and deferral costs. *Operations Research* 23:283–298.

M. Skutella. 2001. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM* 48:206–242.

W. E. Smith. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3:59–66.

C. S. Sung, S. H. Yoon. 1998. Minimizing total weighted completion time at a pre-assembly stage composed of two feeding machines. *International Journal of Production Economics* 54:247–255.

W. T. Trotter. 1992. *Combinatorics and Partially Ordered Sets: Dimension Theory*. Johns Hopkins Series in the Mathematical Sciences. The Johns Hopkins University Press.

W. van den Heuvel, P. Borm, H. Hamers. 2005. Economic lot-sizing games. *European Journal of Operational Research* 176:1117–1130.

V. Vazirani. 2001. *Approximation Algorithms*. Springer, Berlin.

E. Wagneur, C. Sriskandarajah. 1993. Open shops with jobs overlap. *European Journal of Operational Research* 71:366–378.

G. Wang, T. C. E. Cheng. 2003. Customer order scheduling to minimize total weighted completion time. In *Proceedings of the First Multidisciplinary Conference on Scheduling Theory and Applications*, pp. 409–416.

H. Whitney. 1935. On the abstract properties of linear dependence. *American Journal of Mathematics* 57:509–533.

G. J. Woeginger. 2003. On the approximability of average completion time scheduling under precedence constraints. *Discrete Applied Mathematics* 131:237–252.

L. A. Wolsey. 1985. Mixed integer programming formulations for production planning and scheduling problems. Invited talk at the 12th International Symposium on Mathematical Programming, MIT, Cambridge, MA.

J. Yang. 1998. *Scheduling with batch objectives*. Ph.D. thesis, Industrial and Systems Engineering, The Ohio State University.