Algorithms for Discrete, Non-Linear and Robust Optimization Problems with Applications in Scheduling and Service Operations

by

Shashi Mittal

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

September 2011

Author . . . .
Sloan School of Management
August 12, 2011

Certified by . . . .
Andreas S. Schulz
Patrick J. McGovern (1959) Professor of Management
and Professor of Mathematics of Operations Research
Thesis Supervisor

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

Algorithms for Discrete, Non-Linear and Robust Optimization Problems with Applications in Scheduling and Service Operations

by

Shashi Mittal

Submitted to the Sloan School of Management
on August 12, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

This thesis presents efficient algorithms that give optimal or near-optimal solutions for problems with non-linear objective functions that arise in discrete, continuous and robust optimization.

First, we present a general framework for designing approximation schemes for combinatorial optimization problems in which the objective function is a combination of more than one function. Examples of such problems include those in which the objective function is a product or ratio of two or more linear functions, parallel machine scheduling problems with the makespan objective, robust versions of weighted multi-objective optimization problems, and assortment optimization problems with logit choice models. For many of these problems, we give the first fully polynomial time approximation scheme using our framework.

Next, we present approximation schemes for optimizing a rather general class of non-linear functions of low rank over a polytope. In contrast to existing results in the literature, our approximation scheme does not require the assumption of quasi-concavity of the objective function. For the special case of minimizing a quasi-concave function of low-rank, we give an alternative algorithm which always returns a solution which is an extreme point of the polytope. This algorithm can also be used for combinatorial optimization problems where the objective is to minimize a quasi-concave function of low rank. We also give complexity-theoretic results with regards to the inapproximability of minimizing a concave function over a polytope.

Finally, we consider the problem of appointment scheduling in a robust optimization framework. The appointment scheduling problem arises in many service operations, for example health care. For each job, we are given its minimum and maximum possible execution times. The objective is to find an appointment schedule for which the cost in the worst case scenario of the realization of the processing times of the jobs is minimized. We present a global balancing heuristic, which gives an easy to compute closed form optimal schedule when the underage costs of the jobs are non-decreasing. In addition, for the case where we have the flexibility of changing the order of execution of the jobs, we give simple heuristics to find a near-optimal sequence of the jobs.

Thesis Supervisor: Andreas S. Schulz
Title: Patrick J. McGovern (1959) Professor of Management
and Professor of Mathematics of Operations Research

Acknowledgments

I am deeply indebted to my advisor Andreas Schulz for everything that he has taught me in the past five years, starting from the geometry of linear programming to how to actually teach students in a classroom and be patient with them. Andreas gave me the full freedom to pursue my own research agenda, yet he was always ready to help me whenever I got stuck in my research. He instilled in me the confidence that I had the ability to solve any problem, no matter how difficult it appeared. Apart from research, he also helped me with developing many soft skills such as writing good papers, giving good talks and giving a good lecture in a classroom. Working with him has been my best learning experience at MIT, and I will forever be grateful to him for that.

I am also thankful to my colleague Sebastian Stiller for his collaboration in the past one year. The last chapter of this thesis owes much to him. His knack for finding the right models and the right ideas for solving a problem were immensely useful. Thank you Sebastian for all the discussions we had during the lunches at Whitehead, and for the great barbecue parties at your home.

I am thankful to my committee members, Jim Orlin and David Shmoys, for their useful suggestions that helped in making this thesis better. Jim was also the one who convinced me to join the Operations Research Center five years ago, and I thank him for helping me make the best decision of my life. David agreed to be on my committee despite being a visiting member at MIT for only a year, and I am grateful to him for his feedback on my thesis.

My thanks also go to several other faculty members of the institute, who have had a direct or indirect influence on my research. Michel Goemans' undergraduate class on Combinatorial Optimization stimulated my interest in this field; taking two more courses with him helped me learn a lot about algorithms and optimization. I am also thankful to all the professors with whom I was a teaching assistant in various courses: Asuman Ozdaglar, Georgia Perakis and Rama Ramakrishnan. Thanks also to Retsef Levi for being my advisor for the first three terms at MIT. Finally, I would like to thank the co-directors of the Operations Research Center, Dimitris Bertsimas and Patrick Jaillet, for always being willing to listen to the concerns of the students, and for making the ORC an exciting place for doing research.

I am thankful to several past and present colleagues at the ORC for their help. The fourth chapter of this thesis is a direct outcome of a discussion with Vineet Goyal. I am thankful to him for sharing a preliminary draft of his paper on a similar topic. Claudio Telha proofread some of the chapters of this thesis, as did Nikhil Padhye; any remaining errors in the thesis are of course mine. Thanks to

Chaithanya Bandi for all the fun we had on the Cricmetric project. I also benefited enormously from all the numerous informal discussions with the ORC students over lunch, coffee, beer and poker, and I am thankful to all of them for making this five year journey so much intellectually satisfying.

The thesis also benefited from the feedback received by numerous researchers during the presentation of my research work at several conferences and seminars, and many anonymous referees who gave their feedback in the course of peer review of papers submitted to various conferences and journals based on the different chapters of this thesis. I thank all of them for helping me in whatever little ways in my research.

I am also thankful to several staff members at MIT who made my work so much easier. The superb staff at ORC, Andrew Carvalho, Paulette Mosley and Laura Rose, handled all the academic and administrative affairs with great efficiency. I am thankful to David Merrill at the Sloan School of Management for helping me with all the necessary arrangements in the new Sloan building. Thanks also to Janet Rankin at MIT Teaching and Learning Lab, and Marilyn Levine at MIT Writing and Communication Center, for their help in improving my teaching and writing skills respectively.

I would also like to express my gratitude to my undergraduate mentors at the Indian Institute of Technology, Kanpur who were instrumental in stimulating my research interests. I am grateful to Kalyanmoy Deb for introducing me to the field of multi-objective optimization; his influence is evident in chapters 3 and 4 of this thesis. The third chapter of this thesis is an outcome of finding a solution to a problem in parallel computer architecture that came up in a research project with Mainak Chaudhuri; I am thankful to him for suggesting the problem. I also thank Harish Karnick and Anil Seth for guiding me at various stages of my undergraduate studies.

I would like to thank my family members for everything that they have done for me. This thesis is as much a hallmark of their success as it is of mine. It is impossible for me to sum up in words the love, dedication and sacrifice of my parents that have simply made me a better human being. My elder brother and sister were always encouraging in my career goals, and helped me in all possible ways to achieve them. Even though we are all miles apart from each other, the bond of love and affection between us has always been a pillar of strength in my life.

Finally, I dedicate this thesis to two very close people of my life, who passed away earlier this year and who will always be deeply missed: my grandmother, and my long time school friend Prashanth. May their souls rest in peace.

Cambridge, August 2011 *Shashi Mittal*

कर्मण्येवाधिकारस्ते मा फलेषु कदाचन ।
मा कर्मफलहेतुर्भूर्मा ते संगोऽस्त्वकर्मणि ॥ ४७॥

द्वितीयोऽध्यायः, भगवद्गीता

To work alone have you the right and never to the fruits of works. Don't be impelled by the fruits of works, at the same time don't be tempted to withdraw from the works.

Ch. 2, verse 47, Bhagavad Gita

Contents

1 Introduction 17

1.1 Motivation . . . 17

1.2 Contributions of this Thesis . . . 18

1.3 Reading this Thesis . . . 22

1.4 Bibliographic Information . . . 22

2 Preliminaries 23

2.1 Approximation Algorithms and Approximation Schemes . . . 23

2.2 Preliminaries on Multi-objective optimization . . . 24

3 Approximation Schemes for Combinatorial Optimization Problems 29

3.1 Introduction . . . 29

3.1.1 Examples of Problems . . . 30

3.1.2 Related Work . . . 33

3.1.3 Overview of Our Framework . . . 33

3.2 Solving the Gap Problem for the Discrete Case . . . 34

3.3 The General Formulation of the FPTAS . . . 35

3.4 FPTAS for Scheduling and Resource Allocation Problems . . . 37

3.4.1 The $Rm||C_{max}$ Problem and the Max-Min Resource Allocation Problem 37

3.4.2 The Vector Scheduling Problem . . . 39

3.5 FPTAS for Minimizing the Product of Two Linear Objective Functions . . 39

3.5.1 Formulation of the FPTAS . . . 40

3.5.2 FPTAS for Some Problems with the Product Objective Function . . 40

3.5.3 Products of More Than Two Linear Objective Functions . . . 42

3.6 FPTASes for Robust Weighted Multi-Objective Optimization Problems . . 42

3.7 FPTASes for Problems with Rational Objective Functions 45
3.7.1 FPTAS for Assortment Optimization Problems 47
3.8 Conclusion 50

4 Approximation Schemes for Optimizing a Class of Low-Rank Functions Over a Polytope 53
4.1 Introduction 53
4.2 The Approximation Scheme 59
4.2.1 Formulation of the FPTAS 60
4.2.2 Outline of the FPTAS 61
4.3 Applications of the Approximation Scheme 62
4.3.1 Multiplicative Programming Problems 62
4.3.2 Low Rank Bi-Linear Programming Problems 62
4.3.3 Sum-of-Ratios Optimization 63
4.4 The Special Case of Minimizing Quasi-Concave Functions 64
4.4.1 Multiplicative Programming Problems in Combinatorial Optimization 69
4.4.2 Mean-risk Minimization in Combinatorial Optimization 70
4.5 Inapproximability of Minimizing a Concave Function over a Polytope 71
4.6 Open Problems 73

5 Robust Appointment Scheduling 75
5.1 Introduction 75
5.2 Model Description 79
5.2.1 The Robust Model 80
5.3 The Global Balancing Heuristic 81
5.3.1 Analysis of the Global Balancing Heuristic 82
5.3.2 Key Insights 85
5.3.3 Computational Results 87
5.3.4 Comparison of Robust Optimal and Stochastic Optimal Schedules . 88
5.3.5 Impact of Various Factors on the Average Cost of the Robust Schedule 89
5.4 The Ordering Problem 91
5.4.1 A Linear Ordering Formulation for the Ordering Problem 92
5.4.2 An Exact Formulation of the Ordering Problem 94

5.4.3 KKT Conditions for Local Optimality 94
5.4.4 An Approximation Algorithm for the Ordering Problem 96
5.5 Conclusion and Future Directions 99

List of Figures

2-1 Figure illustrating the concept of Pareto-optimal front (shown by the thick boundary) and approximate Pareto-optimal front (shown as solid black points) for two objectives. 25

2-2 Figure illustrating the concept of convex Pareto-optimal front CP (shown by solid black points) and approximate convex Pareto-optimal front CP_ϵ (shown by solid gray points) for two objectives. The dashed lines represent the lower envelope of the convex hull of CP_ϵ 26

5-1 Worst-case scenarios for an optimal solution for the case of 3 jobs. 87

5-2 Comparing the schedules obtained using robust model and stochastic model. 89

5-3 Impact of the number of jobs on the relative performance of the robust schedule. 90

5-4 Impact of the underage cost on the relative performance of the robust schedule. 90

5-5 Impact of the standard deviation on the relative performance of the robust schedule. 91

List of Tables

1.1 Classification of the problems studied in this thesis. 21

Chapter 1

Introduction

1.1 Motivation

Optimization is ubiquitous in the field of engineering and management today. Optimization problems arise in diverse areas such as production and manufacturing, service operations, health care, revenue management, transportation, among others. A glimpse of an array of optimization problems that arise in practise can be found in the survey book by Horst and Pardalos (1995). Because of their importance, optimization problems have been extensively studied - both from a theoretical point of view as well as their practical implementation.

An optimization problem has two parts: the first is the *objective function* that we are trying to optimize. For example, it can correspond to minimizing the cost of a given production schedule, or it could be maximizing the revenue of an airline company. The other is the set of constraints which define the permissible solutions for the optimization problem, also called the *feasible set*. For a given optimization problem, the main theoretical questions of interest is: Can this problem be solved *exactly* in an *efficient* manner? While the meaning of the exactness aspect of this question is fairly obvious, there are well-accepted theoretical notions of efficiency as well. The most commonly used concept of efficiency for algorithms is that for a given class of problems, the number of computational steps required to solve the problem should be a polynomial in the input size of the problem. A well known class of optimization problems that can be solved exactly in polynomial time are *linear programming problems*, in which the objective function is linear and the feasible set is a polyhedron. Linear programming problems belong to a more general class of optimization problem, called *convex optimization problems*, in which the objective function as well as the feasible set are convex. Convex optimization problems have several nice properties, which can be exploited to design polynomial time algorithms for solving these problems almost exactly (Nesterov and Nemirovskii 1961).

However, in real life there are several aspects of optimization problems which make them hard to solve:

1. **The objective function or the feasible set can be non-convex.** In this case, the neat properties associated with the convex optimization problems are lost. Moreover, such problems can have *local optima*, which means that there might be solutions which are good in a local region around the solution, but are worse-off than a *global optimum*, which is the best solution among all the solutions in the feasible set.

2. **The feasible set may be discrete.** In many optimization problems, the variables take only discrete values. Because the set of solutions is no more a continuous set, finding an optimal solution in this discrete set (which in many cases is huge, making an exhaustive search of all the solutions an impractical proposition) becomes difficult.

3. **The parameters of the optimization problem may be uncertain.** This can arise due to two main reasons. Firstly, the process of gathering data for the problem may be noisy, which can lead to uncertainties in the parameters of the optimization problem. Secondly, the parameter itself may have inherent randomness. As an example, consider the generic problem of scheduling patients in a health care facility. The time each patient needs for treatment is random, and this must be taken into account when solving for an optimal schedule for this problem.

Unfortunately, it turns out that for many problems with one or more of the nasty aspects mentioned above, there may not be efficient algorithms for solving them exactly. In fact, the existence of efficient algorithms for these problems is closely related to the P versus NP question in complexity theory, which is a well known open problem (Garey and Johnson 1979). Therefore, we need to relax the exactness and/or the efficiency criteria and design specialized algorithm for solving such optimization problems. This is what we attempt in this thesis.

1.2 Contributions of this Thesis

In this thesis, we look at a gamut of optimization problems with one or more of the features mentioned above and develop algorithms for solving such problems.

Chapter 2 covers the preliminary topics which are subsequently used in the rest of the thesis. Included in this chapter is the notion of *approximation algorithms*, in which we relax the exactness condition on the algorithms by specifying that the algorithm must return a solution which is within a given factor of the optimal solution, and that it must be efficient. We also define the notion of approximation schemes and multi-objective optimization in this chapter.

In Chapter 3, we present a general framework for designing approximation schemes for combinatorial optimization problems in which the objective function is a combination of more than one function. Examples of such problems include those in which the objective function is a product or ratio of two or more linear functions, parallel machine scheduling problems with the makespan objective, robust versions of weighted multi-objective optimization problems, and assortment optimization problems with logit choice models. For many of these problems, we give the first fully polynomial time approximation scheme using our framework.

In Chapter 4, we present approximation schemes for optimizing a rather general class of non-linear functions of low rank over a polytope. The main contribution of this chapter is that unlike the existing results in the literature, our approximation scheme does not require the assumption of quasi-concavity of the objective function. For the special case of minimizing a quasi-concave function of low-rank, we give an alternative algorithm which always returns a solution which is an extreme point of the polytope. This algorithm can also be used for combinatorial optimization problems where the objective is to minimize a quasi-concave function of low rank. We also give complexity-theoretic results with regards to the inapproximability of minimizing a concave function over a polytope.

In Chapter 5, we look at the problem of appointment scheduling, which arises in many service operations, for example health care. In this problem, the main challenge is to deal with the uncertain processing times of the jobs. The traditional approach in the literature to deal with uncertainty is by formulating the problem as a stochastic program. However, stochastic models are usually complicated and computationally intensive to solve. In contrast, we look at this problem in a robust optimization framework, and derive a simple closed-form solution for the optimal duration that should be assigned to each job. Moreover, for the case where we have the flexibility of changing the order of execution of the jobs, we give simple heuristics to find a near-optimal sequence of the jobs as well.

A list of problems studied in Chapters 3 to 5 is given below. This is not an exhaustive list; its aim is to give the reader a flavor of the optimization problems that we tackle in this thesis.

Max-min resource allocation problem: Santa Claus has to distribute a number of gifts among children on Christmas Day. A gift cannot be divided among more than one child, and the happiness of a child is the sum of the happiness derived from the individual gifts given to that child. Santa wants to be fair to all the children, so he wants to distribute the gifts in such a manner so as to maximize the happiness of the least happy child. How should Santa hand out the gifts to the children? (Chapter 3)

Minimum makespan scheduling problems: A user wants to perform multiple queries on a parallel database system. Each query uses multiple resources on a server, for example multiple cores of a processor, memory, hard disks, etc. The performance of the system is governed by the "weakest link" in the database system - that is, the resource with the maximum load on it. How should the queries be assigned to the servers so as to minimize the load on the bottleneck resource? (Chapter 3)

Assortment optimization problems: A cellphone company has to decide what kind of handsets it should display in its showroom. In its inventory the company has many handsets, ranging from the cheap ones with no advanced features, to more costly smart phones with Internet and e-mail facilities, and the really expensive ones with 4G network, large memory and faster processors. If the showroom displays only the cheap handsets, it may turn away customers looking to buy the smart phones. If it displays only the advanced handsets, customers with limited budget may not be interested in buying them. The showroom cannot display all the phones because it has a limited capacity. What assortment of the cellphones should the showroom offer so as to maximize its sales revenue? (Chapter 3)

Multiplicative programming problems: Suppose I have to drive from my home in Albany Street in Cambridge to the Logan airport. Naturally, I want to take the shortest path possible, but in order to avoid traffic, I would also like to take a path which has as few intersections as possible. One way to trade-off between these two objectives is to find a path that minimizes the product of the the length of the path and the total number of intersections on that path. How do I find such a path? (Chapter 3 and 4)

Mean-risk minimization problems: In the problem mentioned above, the speed at which I can drive on a street is uncertain, since it depends on the traffic conditions. Suppose that I know the average time it takes to cover a street, and I also have information about the variance of the time taken. An alternative way to choose a path would be to find a path that minimizes the sum of the average time and the standard deviation of the time taken to drive to the destination. How can I find a path that provides a reasonable trade-off between the average time and the standard deviation? (Chapter 4)

Appointment scheduling problem: A hospital manager needs to schedule surgeries for outpatients in an operation room of the hospital. The time a surgery will take is uncertain. However, the manager needs to plan in advance the time slot that should be assigned to each surgery. If the manager assigns a large interval for a surgery, then it is likely that the surgery will finish early and the operation room will be left underutilized till the next surgery commences. On the other hand, if the manger assigns a small duration for the surgery, then it will likely overshoot its deadline, thereby delaying the next surgery and causing inconvenience to the patients as well as the medical staff. How much duration should the manager assign to each surgery to achieve the right trade-off between these two scenarios? Moreover, if it is possible to change the order in which the surgeries are performed, then what should be the sequence of the surgeries? (Chapter 5)

Table 1.1 gives a classification of the problems according to whether the objective function is linear or non-convex, the underlying feasible set is discrete or continuous, and whether the parameters of the problem are uncertain. Note that although all the problems given in this table have a discrete feasible set, in some cases (for example, multiplicative programming problems) we also look at the corresponding problem with a continuous feasible set. It turns out that algorithms for the continuous case are more efficient and much simpler as compared to the corresponding discrete case (and in some cases, the algorithm can be used for the discrete case as well), hence the reason for considering the continuous case separately.

Problems	**Non-convexity**	**Discreteness**	**Uncertainty**
Max-min resource allocation		✓	
Makespan scheduling		✓	
Assortment optimization	✓	✓	
Multiplicative programming	✓	✓	
Mean-risk minimization	✓	✓	✓
Appointment scheduling	✓	✓	✓

Table 1.1: Classification of the problems studied in this thesis.

1.3 Reading this Thesis

The material of this thesis should be accessible to anyone with a basic knowledge of optimization, particularly in linear programming and to some extent in combinatorial optimization. Chapter 2 covers some of the basic concepts used in the rest of the chapters. Apart from that, Chapters 3 to 5 can be read independently of each other. Instead of discussing the existing literature for the specific problems and the improvements achieved with respect to the current state of the art in this chapter, we defer those materials to the later chapters. For the more industrious reader willing to take a challenge, at the end of each chapter we present a few problems which still remain open.

1.4 Bibliographic Information

Chapter 1 is based on work done in collaboration with Andreas S. Schulz. An extended abstract of this work has appeared as Mittal and Schulz (2008); a journal version is currently under review. Chapter 2 is also joint work with Andreas S. Schulz and is largely based on the technical report Mittal and Schulz (2010). Chapter 3 is joint work with Sebastian Stiller. An extended abstract of this work has appeared as Mittal and Stiller (2011).

Chapter 2

Preliminaries

In this chapter, we discuss some of the basic concepts that will be used in the subsequent chapters of this thesis.

2.1 Approximation Algorithms and Approximation Schemes

An instance π of a single-objective optimization problem Π is given by an objective function $f : X \to \mathbb{R}_+$, where X is some subset of $\mathbb{R}^n$. In this thesis, we consider problems in which either X is a polytope whose concise description is known to us (in terms of linear inequalities, or a separation oracle), or X is a discrete set for which a concise description of its convex hull is known.

If the problem Π is NP-hard, then it is unlikely that there is an algorithm which returns an optimal solution for every instance π of Π, and has a running time which is polynomial in the input size of the problem. In that case, one usually looks for an approximation algorithm for the problem, which is defined below.

Definition 2.1.1 *For a minimization (resp. maximization) problem* Π*, an* α-approximation algorithm *for* $\alpha > 1$ *(resp.* $\alpha < 1$*) is an algorithm* A *which, given any instance* π *of the problem returns a solution* $x_\pi^A \in X$ *such that* $f(x_\pi^A) \leq \alpha \cdot f(x_\pi^*)$ *(resp.* $f(x_\pi^A) \geq \alpha \cdot f(x_\pi^*)$*), where* x_π^* *is an optimal solution to the problem instance* π*. The running time of the algorithm* A *is polynomial in the input size of the problem.*

For certain problems, it is possible to get an approximation algorithm for any factor α arbitrarily close to one. Such a family of algorithms is called an approximation scheme, and is defined below.

Definition 2.1.2 *For a minimization (resp. maximization) problem* Π, *a* polynomial time approximation scheme (PTAS) *is a family of algorithms parametrized by* ϵ *such that for all* $\epsilon > 0$, *there is an algorithm* A_ϵ *which is a* $(1+\epsilon)$*-approximation algorithm (resp.* $(1-\epsilon)$*-approximation algorithm) for the problem, and whose running time is polynomial in the input size of the problem.*

A stronger notion of an approximation scheme is a fully polynomial time approximation scheme.

Definition 2.1.3 *For a minimization (resp. maximization) problem* Π, *a* fully polynomial time approximation scheme (FPTAS) *is a family of algorithms parametrized by* ϵ *such that for all* $\epsilon > 0$, *there is an algorithm* A_ϵ *which is a* $(1+\epsilon)$*-approximation algorithm (resp.* $(1-\epsilon)$*-approximation algorithm) for the problem, and whose running time is polynomial in the input size of the problem, as well as in* $1/\epsilon$.

Theoretically speaking, the existence of an FPTAS for an NP-hard optimization problem is in some sense the strongest possible result one can get for that problem.

2.2 Preliminaries on Multi-objective optimization

An instance π of a multi-objective optimization problem Π is given by a set of k functions $f_1, \ldots, f_k$. Each $f_i : X \to \mathbb{R}_+$ is defined over the same set of feasible solutions, X. Here, X is some subset of $\mathbb{R}^n$ (more specifically, we will consider the case when X or the convex hull of X is a polytope whose concise description is known to us), and k is significantly smaller than n. Let $|\pi|$ denote the binary-encoding size of the instance π. Assume that each f_i takes values in the range $[m, M]$, where $m, M > 0$. We first define the Pareto-optimal frontier for multi-objective optimization problems.

Definition 2.2.1 *Let* π *be an instance of a multi-objective minimization problem. The* Pareto-optimal frontier, *denoted by* $P(\pi)$, *is a set of solutions* $x \in X$, *such that for each* $x \in P(\pi)$, *there is no* $x' \in X$ *such that* $f_i(x') \leq f_i(x)$ *for all* i *with strict inequality for at least one* i.

In other words, $P(\pi)$ consists of all the undominated solutions. If f_i are all linear functions and the feasible set X is a polytope, then the set of function values $(f_1(x), \ldots, f_k(x))$ for $x \in X$ is a polytope in $\mathbb{R}^k$. Then $P(\pi)$ in this case is the set of points on the "lower" boundary of this polytope.

For continuous multi-objective minimization problems, in general, $P(\pi)$ may have infinitely many points, and so we need a more compact representation of the Pareto-optimal frontier. One such way is to use the convex Pareto-optimal frontier, whose definition is given below.

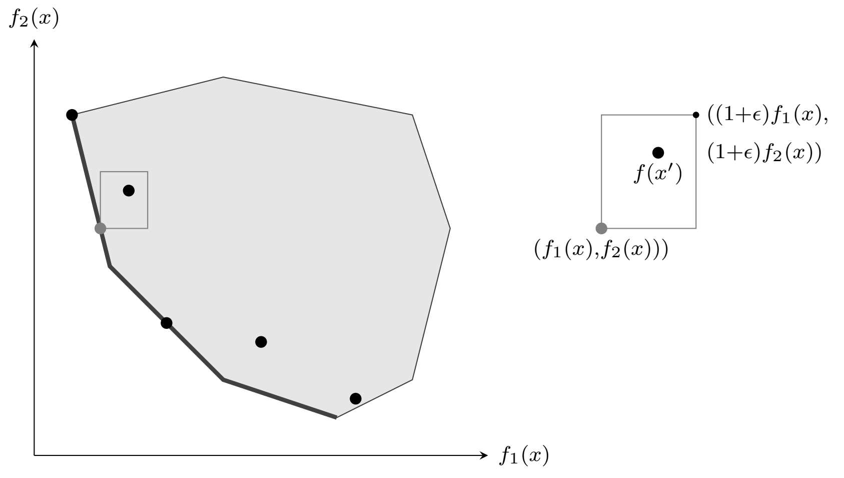

Figure 2-1: Figure illustrating the concept of Pareto-optimal front (shown by the thick boundary) and approximate Pareto-optimal front (shown as solid black points) for two objectives.

Definition 2.2.2 *Let π be an instance of a multi-objective minimization problem. The* convex Pareto-optimal set, *denoted by $CP(\pi)$, is the set of extreme points of $conv(P(\pi))$.*

In many cases, it may not be tractable to compute $P(\pi)$ or even $CP(\pi)$. For example, determining whether a point belongs to the Pareto-optimal frontier for the two-objective shortest path problem is NP-hard (Hansen 1979). Also, the number of undominated solutions for the two-objective shortest path can be exponential in the input size of the problem. This means that $CP(\pi)$ can have exponentially many points, as the shortest path problem can be formulated as a min-cost flow problem, which has a linear programming formulation. This necessitates the idea of using an approximation of the Pareto-optimal frontier. One such notion of an approximate Pareto-optimal frontier is as follows. It is illustrated in Figure 2-1

Definition 2.2.3 *Let π be an instance of a multi-objective minimization problem. For $\epsilon > 0$, an ϵ-*approximate Pareto-optimal frontier, *denoted by $P_\epsilon(\pi)$, is a set of solutions, such that for all $x \in X$, there is $x' \in P_\epsilon(\pi)$ such that $f_i(x') \leq (1+\epsilon)f_i(x)$, for all i.*

Similar to the notion of an approximate Pareto-optimal frontier, we need to have a notion of an approximate convex Pareto-optimal frontier, defined below. The concept of convex Pareto-optimal set and approximate Pareto-optimal set is illustrated in Figure 2-2.

Definition 2.2.4 *Let π be an instance of a multi-objective minimization problem. For $\epsilon > 0$, an ϵ-*approximate Pareto-optimal set, *denoted by $CP_\epsilon(\pi)$, is a set of solutions such that for any $x \in X$, there is x' in $conv(CP_\epsilon(\pi))$ such that $f_i(x') \leq (1+\epsilon)f_i(x)$, for all i.*

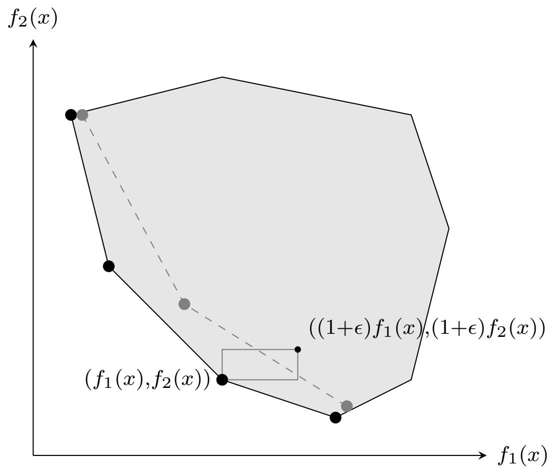

Figure 2-2: Figure illustrating the concept of convex Pareto-optimal front CP (shown by solid black points) and approximate convex Pareto-optimal front CP_ϵ (shown by solid gray points) for two objectives. The dashed lines represent the lower envelope of the convex hull of CP_ϵ

In the rest of the paper, whenever we refer to an (approximate) Pareto-optimal frontier or its convex counterpart, we mutually refer to both its set of solutions and their vectors of objective function values. Even though $P(\pi)$ may contain exponentially many (or even uncountably many) solution points, there is always an approximate Pareto-optimal frontier that has polynomially many elements, provided k is fixed. The following theorem gives one possible way to construct such an approximate Pareto-optimal frontier in polynomial time. We give a proof of this theorem here, as the details will be needed for designing the approximation schemes in the later chapters.

Theorem 2.2.5 (Papadimitriou and Yannakakis (2000)) *Let k be fixed, and let $\epsilon, \epsilon' > 0$ be such that $(1-\epsilon')(1+\epsilon)^{1/2} = 1$. One can determine a $P_\epsilon(\pi)$ in time polynomial in $|\pi|$ and $1/\epsilon$ if the following 'gap problem' can be solved in polynomial-time: Given a k-vector of values $(v_1, \ldots, v_k)$, either*
(i) *return a solution $x \in X$ such that $f_i(x) \leq v_i$ for all $i = 1, \ldots, k$, or*
(ii) *assert that there is no $x \in X$ such that $f_i(x) \leq (1-\epsilon')v_i$ for all $i = 1, \ldots, k$.*

Proof. Suppose we can solve the gap problem in polynomial time. An approximate Pareto-optimal frontier can then be constructed as follows. Consider the hypercube in $\mathbb{R}^k$ of possible function values given by $\{(v_1, \ldots, v_k) : m \leq v_i \leq M \text{ for all } i\}$. We divide this hypercube into smaller hypercubes, such that in each dimension, the ratio of successive divisions is equal to $1+\epsilon''$, where $\epsilon'' = \sqrt{1+\epsilon} - 1$. For each corner point of all such smaller hypercubes, we solve the gap problem. Among all solutions returned by solving the gap problems, we keep only those solutions that are not Pareto-dominated by any other solution. This is the required $P_\epsilon(\pi)$. To see this, it suffices to prove that every point $x^* \in P(\pi)$ is approximately dominated by some point in $P_\epsilon(\pi)$. For such a solution point x^*, there is a corner point $v = (v_1, \ldots, v_k)$ of some hypercube such that $f_i(x^*) \leq v_i \leq (1+\epsilon'')f_i(x^*)$ for $i = 1, \ldots, k$. Consider the solution of the gap problem for $y = (1+\epsilon'')v$. For the point y, the algorithm for solving the gap problem cannot assert (ii) because the point x^* satisfies $f_i(x^*) \leq (1-\epsilon')y_i$ for all i. Therefore, the algorithm must return a solution x' satisfying $f_i(x') \leq y_i \leq (1+\epsilon)f_i(x^*)$ for all i. Thus, x^* is approximately dominated by x', and hence by some point in $P_\epsilon(\pi)$ as well. Since we need to solve the gap problem for $O((\log{(M/m)}/\epsilon)^k)$ points, this can be done in polynomial time. □

We will refer to the above theorem as the gap theorem. Solving the gap problem will be the key to designing the approximation schemes in the later chapters. For the case where $f_i(x)$ are continuous linear functions, the gap problem can be solved using a linear program. For the discrete case, however, solving the gap problem requires more effort (see Chapter 3 for more details).

Further Reading

The standard reference on NP-hardness is Garey and Johnson (1979). For readers interested in approximation algorithms, the book by Williamson and Shmoys (2011) is an excellent text. An extensive discussion on computing approximate Pareto-optimal fronts for multi-objective combinatorial optimization problems can be found in Safer and Orlin (1995a), Safer and Orlin (1995b) and Safer, Orlin, and Dror (2004).

Chapter 3

Approximation Schemes for Combinatorial Optimization Problems

3.1 Introduction

In many combinatorial optimization problems, the objective function is a combination of more than one function. For example, consider the problem of finding a spanning tree in a graph $G = (V, E)$ with two edge weights c_1 and c_2, where c_1 may correspond to the failure probability of the edges, and c_2 to the cost of the edges. The objective is to find a spanning tree T of the graph for which $c_1(T) \cdot c_2(T)$ is minimized (Kuno 1999; Kern and Woeginger 2007). In this problem, the objective function is a combination of two linear objective functions combined together using the product function.

Another example of a problem whose objective function subsumes more than one function is the max-min resource allocation problem (Asadpour and Saberi 2007). Here, there are several resources which have to be distributed among agents. The utility of each agent is the sum of the utilities of the resources assigned to the agent. The objective is to maximize the utility of the agent with the lowest utility. In this problem, one can look at the utility of each agent as a separate objective function. Thus, the objective function of the problem is a combination of the objective functions of the individual agents using the minimum function.

This chapter presents a unified approach for solving combinatorial optimization problems in which the objective function is a combination of more than one (but a fixed number) of objective functions. Usually, these problems turn out to be NP-hard. We show that under very general conditions, we can obtain FPTASes for such problems. Our technique turns out be surprisingly versatile: it can be applied to a variety of scheduling problems (e.g. unrelated parallel machine scheduling and vector scheduling), combinatorial optimization problems with a non-linear objective function such as a product or a ratio of two linear functions, and robust versions of weighted multi-objective optimization problems. We first give examples of some of the problems for which we can get FPTASes using our framework.

3.1.1 Examples of Problems

Combinatorial optimization problems with a rational objective: Consider the problem in which the objective function is a ratio of discrete linear functions:

$$\begin{aligned} \text{minimize } g(x) \quad &= \quad \frac{f_1(x)}{f_2(x)} = \frac{a_0 + a_1x_1 + \ldots + a_dx_d}{b_0 + b_1x_1 + \ldots + b_dx_d}, \\ \text{s.t.} \quad & \quad x \in X \subseteq \{0,1\}^d. \end{aligned} \tag{3.1}$$

We assume that $f_1(x) > 0, f_2(x) > 0$ for all $x \in X$. In this case, there are two linear objective functions that have been combined by using the quotient function. A well known example is the computation of a minimum mean cost circulation in graphs. Megiddo (1979) showed that any polynomial time algorithm for the corresponding linear objective problem can be used to obtain a polynomial time algorithm for the same problem with a rational objective function. Extensions of this result have been given by Hashizume, Fukushima, Katoh, and Ibaraki (1987), Billionnet (2002) and Correa, Fernandes, and Wakabayashi (2010) to obtain approximation algorithms for the case where the corresponding linear objective problem is NP-hard. The main idea behind all these approaches is to convert the problem with a rational objective function to a parametric linear optimization problem, and then perform a binary search on the parameter to get an approximate solution for the problem. The main drawback of parametric methods is that they do not generalize to the case where we have a sum of ratios of linear functions.

In Section 3.7, we give a fairly general sufficient condition for the existence of an FPTAS for this problem. It can be used to obtain an FPTAS, for example, for the knapsack problem with a rational objective function. In contrast to the methods described above, our algorithm uses a non-parametric approach to find an approximate solution. One distinct advantage of our technique is that it easily generalizes to more general rational functions as well, for example the sum of a fixed number of ratios of linear functions. Such a form often arises in assortment optimization in the context of retail management, and in Section 3.7.1, we show how to obtain FPTASes using our framework for assortment optimization problems under two different choice models.

Resource allocation and scheduling problems: The best known approximation algorithm for the general max-min resource allocation problem has an approximation ratio of $O\left(\frac{1}{\sqrt{m}\log^3 m}\right)$, where m is the number of agents (Asadpour and Saberi 2007). In Section 3.4.1, we obtain the first FPTAS for this problem when the number of agents is fixed.

Scheduling problems can be thought of as an inverse of the resource allocation problem, where we want to assign jobs to machines, and attempt to minimize the load on individual machines. Corresponding to the max-min resource allocation problem, we have the problem of scheduling jobs on unrelated parallel machines to minimize the makespan (i.e. the time at which the last job finishes its execution). When the number of machines m is fixed, this problem is referred to as the $Rm||C_{\max}$ problem. Another objective function that has been considered in the literature is the one in which the total load on different machines are combined together using an l_p norm (Azar, Epstein, Richter, and Woeginger 2004). In Section 3.4.1, we give FPTASes for both of these scheduling problems. It should be noted that approximation schemes for the $R||C_{\max}$ problem already exist in the literature (e.g. Sahni (1976) and Lenstra, Shmoys, and Tardos (1990)).

A generalization of the $Rm||C_{\max}$ problem is the vector scheduling problem. In this problem, a job uses multiple resources on each machine, and the objective is to assign the jobs to the machines so as to minimize the maximum load over all the resources of the machines. A practical situation where such a problem arises is query optimization in parallel database systems (Garofalakis and Ioannidis 1996). In this case, a job is a database query, which uses multiple resources on a computer - for example, multiple cores of a processor, memory, hard disk etc. A query can be assigned to any one of the multiple servers in the database system. Since the overall performance of a system is governed by the resource with the maximum load, the objective is to minimize over all the resources, the maximum load. Chekuri and Khanna (2004) give a PTAS for the problem when the number of resources on each machine is fixed. Moreover, they only consider the case where each job has the same requirement for a particular resource on all the machines. In Section 3.4.2, we show that when both the number of machines and resources are fixed, we can get an FPTAS for the problem, even when each job can use different amounts of a resource on different machines.

Combinatorial optimization problems with a product objective: For the product versions of the minimum spanning tree problem and the shortest s-t path problem, Kern and Woeginger (2007) and Goyal, Genc-Kaya, and Ravi (2011) give an FPTAS. Both these methods are based on linear programming techniques, and do not generalize to the case where we have more than two functions in the product. Moreover, their techniques do not extend to the case where the corresponding problem with a linear objective function is NP-hard.

In Section 3.5.3 of this chapter, we give FPTASes for the product version of the s-t path problem and the spanning tree problem using our framework. A big advantage of our method is that it easily generalizes to the case where the objective function is a product of a fixed number of linear functions. It can also be used to design approximation schemes for the product version of certain NP-hard problems, such as the knapsack problem.

Robust weighted multi-objective optimization problems: Consider once again the spanning tree problem with two cost functions c_1 and c_2 on the edges, as given in the introduction. One way to combine the two costs is to find a spanning tree T which minimizes the weighted sum $w_1c_1(T) + w_2c_2(T)$ for some positive weights w_1 and w_2. However, in many cases it is not clear a-priori which weights should be used to combine the two objectives. An alternative is to allow the weights $w = (w_1, w_2)$ to take values in a set W, and find a spanning tree that minimizes the cost of the weighted objective for the worst case scenario weight in the set $W \subseteq \mathbb{R}^2_+$. This ensures a fair trade-off of the two cost functions. More generally, we consider the following robust version of a weighted multi-objective optimization problem:

$$\text{minimize } g(x) = \max_{w \in W} w^T f(x), \qquad x \in X \subseteq \{0,1\}^d. \tag{3.2}$$

Here, $f(x) = (f_1(x), \ldots, f_m(x))$ is a vector of m function values and $W \subseteq \mathbb{R}^m$ is a compact convex weight set. For the spanning tree problem and the shortest path problem, the above robust version is NP-hard even for the case of two objectives.

The robust version of weighted multi-objective optimization problems has been studied by Hu and Mehrotra (2010) for the case when each f_i is a continuous function. For discrete optimization problems, this formulation is a generalization of the robust discrete optimization model with a fixed number of scenarios (see e.g. Kouvelis and Yu (1997)). This problem is NP-hard, but admits an FPTAS for the robust version of many problems when the number of scenarios is fixed (Aissi, Bazgan, and Vanderpooten 2007). In Section 3.6, we generalize this result and show that we can get FPTASes for the robust version of weighted multi-objective optimization problems when the number of objectives is fixed, for the case of the spanning tree problem, the shortest path problem and the knapsack problem.

3.1.2 Related Work

There are two well known general methods for obtaining approximation schemes for combinatorial optimization problems. In the first method, the input parameters of the problem are rounded and then dynamic programming is applied on the modified instance to find an approximate solution for the original problem. This idea has been extensively used to find approximation schemes for a number of machine scheduling problems (e.g. Sahni (1976), Horowitz and Sahni (1976), Hochbaum and Shmoys (1987), Lenstra, Shmoys, and Tardos (1990)). The other method is shrinking the state space of the dynamic programs that solve the problem in pseudo-polynomial time. This idea was first used by Ibarra and Kim (1975) to obtain an approximation scheme for the knapsack problem. Woeginger (2000) gives a very general framework where such dynamic programs can be converted to an FPTAS, and using this framework he derives FPTASes for several scheduling problems. Another example is the work of Halman, Klabjan, Mostagir, Orlin, and Simchi-Levi (2009), who adopt the same methodology to get FPTASes for inventory management problems.

3.1.3 Overview of Our Framework

We present a general framework which we use to design FPTASes for the problems given in Section 3.1.1. The main idea behind this framework is to treat each objective function as a separate objective, and compute the approximate Pareto-optimal front corresponding to these objective functions. It is possible to get an approximate Pareto-optimal front for many combinatorial optimization problems under the general condition that the corresponding "exact" problem is solvable in pseudo-polynomial time. The exact problem, for example, for the spanning tree problem is, given a vector of non-negative integer edge weights c and a non-negative integer K, does there exist a spanning tree T such that $c(T) = K$? For many combinatorial optimization problems (e.g. the spanning tree problem, the shortest path problem, and the knapsack problem) the exact problem can indeed be solved in pseudo-polynomial time. For the resource allocation and scheduling problems, the exact problem is a variant of the partition problem, and we show that it is also solvable in pseudo-polynomial time.

Our framework works in the following three stages:

1. Show that the optimal solution of the problem lies on the *Pareto-optimal front* of the corresponding multi-objective optimization problem.

2. Show that there is at least one solution in the *approximate Pareto-optimal front* which is an approximate solution of the given optimization problem.

3. To construct the approximate Pareto-optimal front, in many cases it is sufficient to solve the *exact problem* corresponding to the original optimization problem in pseudo-polynomial time.

In Section 3.2, we first show how to solve the gap problem to construct an approximate Pareto-optimal front for a multi-objective discrete optimization problem. We then give our general framework for designing FPTAS for combinatorial optimization problems in which several objective functions are combined into one, and state the conditions needed for the framework to work in the main theorem of this chapter in Section 3.3. Subsequently, we derive FPTASes for the problems mentioned in Section 3.1.1 as corollaries to the main theorem.

3.2 Solving the Gap Problem for the Discrete Case

From Theorem 2.2.5, we know that it suffices to solve the gap problem to compute an approximate Pareto-optimal front. We give a procedure here for solving the gap problem with respect to minimization problems, but it can be extended to maximization problems as well (see Section 3.7).

We restrict our attention to the case when $X \subseteq \{0,1\}^d$, since many combinatorial optimization problems can be framed as $0/1$-integer programming problems. Further, we consider linear objective functions; that is, $f_i(x) = \sum_{j=1}^{d} a_{ij}x_j$, and each a_{ij} is a non-negative integer. Suppose we want to solve the gap problem for the m-vector $(v_1, \ldots, v_m)$. Let $r = \lceil d/\epsilon' \rceil$. We first define a "truncated" objective function. For all $j = 1, \ldots, d$, if for some i, $a_{ij} > v_i$, we set $x_j = 0$, and drop the variable x_j from each of the objective functions. Let V be the index set of the remaining variables. Thus, the coefficients in each objective function are now less than or equal to v_i. Next, we define a new objective function $f'_i(x) = \sum_{j \in V} a'_{ij}x_j$, where $a'_{ij} = \lceil a_{ij}r/v_i \rceil$. In the new objective function, the maximum value of a coefficient is now r. For $x \in X$, by Lemma 3.8.1 (see appendix) the following two statements hold.

- If $f'_i(x) \leq r$, then $f_i(x) \leq v_i$.

- If $f_i(x) \leq v_i(1 - \epsilon')$, then $f'_i(x) \leq r$.

Therefore, to solve the gap problem, it suffices to find an $x \in X$ such that $f'_i(x) \leq r$, for $i = 1, \ldots, m$, or assert that no such x exists. Since all the coefficients of $f'_i(x)$ are non-negative integers, there are $r+1$ ways in which $f'_i(x) \leq r$ can be satisfied. Hence there are $(r+1)^m$ ways overall in which all inequalities $f'_i(x) \leq r$ can be simultaneously satisfied. Suppose we want to find if there is an $x \in X$ such that $f'_i(x) = b_i$ for $i = 1, \ldots, m$. This is equivalent to finding an x such that $\sum_{i=1}^{m} M^{i-1} f'_i(x) = \sum_{i=1}^{m} M^{i-1} b_i$, where $M = dr + 1$ is a number greater than the maximum value that $f'_i(x)$ can take.

Given an instance π of a multi-objective linear optimization problem over a discrete set $X \subseteq \{0,1\}^d$, the exact version of the problem is: Given a non-negative integer C and a vector $(c_1, \ldots, c_d) \in \mathbb{Z}_+^d$, does there exist a solution $x \in X$ such that $\sum_{j=1}^{d} c_j x_j = C$? The following theorem establishes the connection between solving the exact problem and the construction of an approximate Pareto-optimal front.

Theorem 3.2.1 *Suppose we can solve the exact version of the problem in pseudo-polynomial time, then there is an FPTAS for computing the approximate Pareto-optimal curve $P_\epsilon(\pi)$.*

Proof. The gap problem can be solved by making at most $(r+1)^m$ calls to the pseudo-polynomial time algorithm for the exact problem, and the input to each call has numerical values of order O$((d^2/\epsilon)^{m+1})$. Therefore, all calls to the algorithm take polynomial time, hence the gap problem can be solved in polynomial time. The theorem now follows from Theorem 2.2.5. $\square$

3.3 The General Formulation of the FPTAS

In this section, we present a general formulation of the FPTAS based on the ideas given in Section 2.2. We then show how this general framework can be adapted to obtain FPTASes for the problems given in Section 3.1.1.

Let $f_1, \ldots, f_m$, for m fixed, be functions which satisfy the conditions given in the beginning of Section 2.2. Let $h : \mathbb{R}_+^m \to \mathbb{R}_+$ be any function that satisfies the following two conditions.

1. $h(y) \leq h(y')$ for all $y, y' \in \mathbb{R}_+^m$ such that $y_i \leq y'_i$ for all $i = 1, \ldots, m$, and
2. $h(\lambda y) \leq \lambda^c h(y)$ for all $y \in \mathbb{R}_+^m$ and $\lambda > 1$, for some fixed $c > 0$.

In particular, h includes all the l_p norms (with $c = 1$), and the product of a fixed number (say, k) of linear functions (with $c = k$). We denote by $f(x)$ the vector $(f_1(x), \ldots, f_m(x))$.

Consider the following general optimization problem:

$$\text{minimize } g(x) = h(f(x)), \qquad x \in X. \tag{3.3}$$

We show that if we can solve the corresponding exact problem (with a singe linear objective function) in polynomial time, then there is an FPTAS to solve this general optimization problem as well. Also, even though we state all our results for minimization problems, there is a straightforward extension of the method to maximization problems as well.

Lemma 3.3.1 *There is at least one optimal solution x^* to* (3.3) *such that $x^* \in P(\pi)$.*

Proof. Let $\hat{x}$ be an optimal solution of (3.3). Suppose $\hat{x} \notin P(\pi)$. Then there exists $x^* \in P(\pi)$ such that $f_i(x^*) \leq f_i(\hat{x})$ for $i = 1, \ldots, m$. By Property 1 of $h(x)$, $h(f(x^*)) \leq h(f(\hat{x}))$. Thus x^* minimizes the function g and is in $P(\pi)$. □

Lemma 3.3.2 *Let $\hat{\epsilon} = (1+\epsilon)^{1/c} - 1$. Let $\hat{x}$ be a solution in $P_{\hat{\epsilon}}(\pi)$ that minimizes $g(x)$ over all the points $x \in P_{\hat{\epsilon}}(\pi)$. Then $\hat{x}$ is a $(1+\epsilon)$-approximate solution of* (3.3)*; that is, $g(\hat{x})$ is at most $(1+\epsilon)$ times the value of an optimal solution to* (3.3).

Proof. Let x^* be an optimal solution of (3.3) that is in $P(\pi)$. By the definition of an ϵ-approximate Pareto-optimal frontier, there exists $x' \in P_{\hat{\epsilon}}(\pi)$ such that $f_i(x') \leq (1+\hat{\epsilon}) f_i(x^*)$, for all $i = 1, \ldots, m$. Therefore,

$$\begin{aligned} g(x') = h(f_1(x'), \ldots, f_m(x')) &\leq h((1+\hat{\epsilon})f_1(x^*), \ldots, (1+\hat{\epsilon})f_m(x^*)) \\ &\leq (1+\hat{\epsilon})^c h(f_1(x^*), \ldots, f_m(x^*)) = (1+\epsilon) g(x^*), \end{aligned}$$

where the first inequality follows from Property 1 and the second inequality follows from Property 2 of h. Since $\hat{x}$ is a minimizer of $g(x)$ over all the solutions in $P_{\hat{\epsilon}}(\pi)$, $g(\hat{x}) \leq g(x') \leq (1+\epsilon) g(x^*)$. □

From these two lemmata and Theorem 3.2.1, we get the main theorem of this chapter regarding the existence of an FPTAS for solving (3.3).

Theorem 3.3.3 *Suppose the exact problem corresponding to the functions given in* (3.3) *can be solved in pseudo-polynomial time. Then there is an FPTAS for solving the general optimization problem* (3.3) *when m is fixed.*

The FPTAS can now be summarized as follows.

1. Sub-divide the space of objective function values $[2^{-p(|\pi|)}, 2^{p(|\pi|)}]^m$ into hypercubes, such that in each dimension, the ratio of two successive divisions is $1 + \epsilon''$, where $\epsilon'' = (1 + \epsilon)^{1/2c} - 1$.

2. For each corner of the hypercubes, solve the corresponding gap problem, and keep only the set of non-dominated solutions obtained from solving each of the gap problems.

3. Among all the solutions in the non-dominated front, return the one with the minimum function value.

Finally, we establish the running time of the above algorithm.

Lemma 3.3.4 *The running time of the algorithm is polynomial in $|\pi|$ and $1/\epsilon$.*

Proof. There are $O((\frac{p(|\pi|)}{\epsilon})^m)$ corner points for which we need to solve the gap problem. Solving each gap problem requires calling the algorithm for solving the exact problem $O(r^m)$ times, which is $O((\frac{d}{\epsilon})^m)$. The magnitude of the largest number input to the algorithm for the exact problem is $O((\frac{d^2}{\epsilon})^{m+1})$. Hence the running time of the algorithm is $O((\frac{p(|\pi|)d}{\epsilon^2}) \cdot PP((\frac{d^2}{\epsilon})^{m+1}, m, d))$, where $PP(M, m, d)$ is the running time of the pseudo-polynomial time algorithm for the exact problem with maximum magnitude of an input number equal to M. $\square$

3.4 FPTAS for Scheduling and Resource Allocation Problems

Using the framework presented in Section 3.3, we give FPTASes for the max-min resource allocation problem, the $Rm||C_{\max}$ problem and the vector scheduling problem.

3.4.1 The $Rm||C_{max}$ Problem and the Max-Min Resource Allocation Problem

Recall the $Rm|\,|C_{\max}$ scheduling problem defined in the introduction. There are m machines and n jobs, and the processing time of job k on machine i is p_{ik}. The objective is to schedule the jobs to minimize the makespan. The max-min resource allocation problem is similar to this scheduling problem, except that the objective here is to maximize the minimum completion time over all the machines. Observe that this corresponds to h being the l_∞-norm with $c = 1$ in the formulation given by (3.3).

We first give an integer programming formulation for the two problems. Let x_{ik} be the variable which is 1 if job k is assigned to machine i, 0 otherwise. The m objective functions in this case (corresponding to each agent/machine) are given by $f_i(x) = \sum_{k=1}^{n} p_{ik} x_{ik}$, and the set X is given by

$$\sum_{i=1}^{m} x_{ik} = 1 \qquad \text{for } k = 1, \ldots n, \tag{3.4a}$$

$$x_{ik} \in \{0,1\} \qquad \text{for } i = 1, \ldots, m, \quad k = 1, \ldots, n. \tag{3.4b}$$

The exact version for both the problems is this: Given an integer C, does there exist a $0/1$-vector x such that $\sum_{k=1}^{n} \sum_{j=1}^{m} c_{jk} x_{jk} = C$, subject to the constraints (3.4a) and (3.4b)? The following lemma establishes that the exact problem can be solve in pseudo-polynomial time.

Lemma 3.4.1 *The exact problem for the max-min resource allocation problem and the* $Rm||C_{\max}$ *problem can be solved in pseudo-polynomial time.*

Proof. The exact problem can be viewed as a reachability problem in a directed graph. The graph is an $(n+1)$-partite directed graph; let us denote the partitions of this digraph by $V_0, \ldots, V_n$. The partition V_0 has only one node, labeled as $v_{0,0}$ (the source node), all other partitions have $C+1$ nodes. The nodes in V_i for $1 \le i \le n$ are labeled as $v_{i,0}, \ldots, v_{i,C}$. The arcs in the digraph are from nodes in V_i to nodes in V_{i+1} only, for $0 \le i \le n-1$. For all $c \in \{c_{1,i+1}, \ldots, c_{m,i+1}\}$, there is an arc from $v_{i,j}$ to $v_{i+1,j+c}$, if $j + c \le C$. Then there is a solution to the exact version if and only if there is a directed path from the source node $v_{0,0}$ to the node $v_{n,C}$. Finding such a path can be accomplished by doing a depth-first search from the node $v_{0,0}$. The corresponding solution for the exact problem (if it exists) can be obtained using the path found by the depth-first search algorithm. □

Therefore, we obtain FPTASes for both the $Rm||C_{\max}$ problem as well as the max-min resource allocation problem. For the max-min resource allocation problem with a fixed number of agents, we give the first FPTAS, though approximation schemes for the $R||C_{\max}$ problem already exist in the literature (e.g. Sahni (1976) and Lenstra, Shmoys, and Tardos (1990)). Further, Theorem 3.3.3 implies that we get an FPTAS even when the objectives for different agents/machines are combined together using any norm. We therefore have the following corollary to Theorem 3.3.3.

Corollary 3.4.2 *There is an FPTAS for the max-min resource allocation problem with a fixed num-*

ber of agents. Further, we also get an FPTAS for the min-max resource allocation problem with a fixed number of agents and the unrelated parallel machine problem when the objectives for different agents/machines are combined by some norm.

3.4.2 The Vector Scheduling Problem

The vector scheduling problem is a generalization of the $Rm||C_{\max}$ problem. In this problem, each job requires d resources for execution on each machine. Job k consumes an amount p_{ik}^j of a resource j on machine i. Suppose J_i is the set of jobs assigned to machine i. Thus the total usage of resource j on machine i is $\sum_{k \in J_i} p_{ik}^j$. The objective is to minimize over all the machines i and all the resources j, the value $\sum_{k \in J_i} p_{ik}^j$. We assume that both d and m are fixed.

Similar to the $Rm||C_{\max}$ problem, let x_{ik} be a variable that is 1 if job k is assigned to machine i, 0 otherwise. In this case, we have a total of md functions and $f_{ij}(x) = \sum_{k=1}^{n} p_{ik}^j x_{ik}$, for $i = 1, \dots m$ and $j = 1, \dots, d$. The md objective function are combined together using the l_∞ norm in this problem. The underlying set of constraints is the same as given by (3.4a)-(3.4b). Therefore, the exact algorithm for the $Rm||C_{\max}$ problem works for the vector scheduling problem as well, and since we have a fixed number of objective functions, we get an FPTAS for the vector scheduling problem as well. Hence we have the following corollary to Theorem 3.3.3.

Corollary 3.4.3 *There is an FPTAS for the vector scheduling problem when the number of machines as well as the number of resources are fixed, even for the case when each job can use a different amount of a particular resource on different machines.*

3.5 FPTAS for Minimizing the Product of Two Linear Objective Functions

In this section, we give a general framework for designing FPTASes for problems in which the objective is to minimize the product of two linear cost functions. We then apply this technique to some product combinatorial optimization problems on graphs, and then extend it to the case where the objective function is a product of a fixed number of linear functions.

3.5.1 Formulation of the FPTAS

Consider the following optimization problem.

$$\text{minimize } g(x) = f_1(x) \cdot f_2(x), \qquad x \in X, \tag{3.5}$$

where $f_i : X \to \mathbb{Z}_+$ are linear functions and $X \subseteq \{0,1\}^d$. In our general formulation given by (3.3), the corresponding function h for this case is $h(y_1, y_2) = y_1 y_2$, and so $c = 2$. Thus, if we can construct an approximate Pareto-optimal front for $f_1(x)$ and $f_2(x)$ in polynomial time, we will be able to design an FPTAS for the product optimization problem. Therefore, we get the following corollary to Theorem 3.3.3.

Corollary 3.5.1 *There is an FPTAS for the problem given by* (3.5) *if the following exact problem can be solved in pseudo-polynomial time: Given* $(c_1, \ldots, c_d) \in \mathbb{Z}_+^d$ *and* $K \in \mathbb{Z}_+$*, does there exist* $x \in X$ *such that* $\sum_{i=1}^{d} c_i x_i = K$*?*

3.5.2 FPTAS for Some Problems with the Product Objective Function

Using the above theorem, we now construct FPTASes for several combinatorial optimization problems involving the product of two objective functions.

1. **Spanning tree problem:** In this case, the exact problem is: given a graph $G = (V, E)$ with cost function $c : E \to \mathbb{Z}_+$ and a positive integer K, does there exist a spanning tree $T \subseteq E$ whose cost is equal to exactly k? Barahona and Pulleyblank (1987) give an $O((n^3 + p^2)p^2 \log p)$ algorithm for solving the exact problem, where n is the number of vertices in the graph and $p = n \cdot \max_e (c(e))$. Thus we have an FPTAS for the spanning tree problem with the product of two cost functions as the objective.

2. **Shortest s-t path problem:** The exact problem in this case is: given a graph $G = (V, E)$, vertices $s, t \in V$, a distance function $d : E \to \mathbb{Z}_+$ and an integer K, is there an s-t path with length equal to exactly K? Note that for the shortest path problem, the exact problem is strongly NP-complete, since it includes the Hamiltonian path problem as a special case. To circumvent this issue, we relax the original problem to that of finding a walk (in which a vertex can be visited more than once) between the vertices s and t that minimizes the product objective. The optimal solution of the relaxed problem will have the same objective function value as that of an optimal solution of the original problem, since any s-t walk can

be truncated to get an s-t path. Therefore, it suffices to get an approximate solution for the relaxed problem.

The corresponding exact s-t walk problem is: Does there exist an s-t walk in the graph whose length is equal to exactly a given number $K \in \mathbb{Z}_+$? Since we are dealing with non-negative weights, this problem can be solved in $O(mnK)$ time by dynamic programming, where n is the number of vertices and m is the number of edges in the graph. If the solution given by the algorithm is a walk instead of a path, we remove the cycles from the walk to get a path. Hence we obtain an FPTAS for the shortest s-t path problem with the product of two distance functions as the objective.

3. **Knapsack problem:** The exact problem for the knapsack problem is: given a set I of items with profit $p : I \to \mathbb{Z}_+$, size $s : I \to \mathbb{Z}_+$ and a capacity constraint C, does there exist a subset of I satisfying the capacity constraint and having total profit exactly equal to a given integer K? Again, this exact problem can be solved in $O(nK)$ time by dynamic programming, where n is the number of objects. Therefore we get an FPTAS for the product version of the knapsack problem.

4. **Minimum cost flow problem:** The problem we have is: given a directed graph $G = (V, A)$, vertices $s, t \in V$, an amount of flow $d \in \mathbb{Z}_+$ to send from s to t, capacities $u : A \to \mathbb{Z}_+$, and two cost functions $c_1, c_2 : A \to \mathbb{Z}_+$, find a feasible s-t flow x of total amount d such that $c_1(x) \cdot c_2(x)$ is minimized. The minimum cost flow problem is different from the above two problems, since it can be formulated as a linear program, instead of an integer linear program. In this case, the gap problem as stated in Theorem 2.2.5 can be solved directly using linear programming. Therefore we obtain an FPTAS for the minimum cost flow problem with the product objective function as well.

   Note that in this case, the approximate solution that we obtain may not necessarily be integral. This is because when we solve the gap problem, we introduce constraints of the form $f_i(x) \leq (1 - \epsilon')v_i$ corresponding to each of the two objectives, in addition to the flow conservation and capacity constraints. This means that the constraint set may not be totally unimodular, and hence the solution obtained can possibly be non-integral.

A big advantage of our method is that it can be used to get an approximation scheme for the product version of an optimization problem even if the original problem is NP-hard, for example in the case of the knapsack problem, whereas previously existing methods cannot handle this case.

3.5.3 Products of More Than Two Linear Objective Functions

Another advantage of our technique over existing methods for designing FPTASes for product optimization problems (Kern and Woeginger 2007; Goyal, Genc-Kaya, and Ravi 2011) is that it can be easily extended to the case where the objective function is a product of more than two linear functions, as long as the total number of functions involved in the product is a constant. Consider the following generalization of the problem given by (3.5).

$$\text{minimize } g(x) = f_1(x) \cdot f_2(x) \cdot \ldots \cdot f_m(x), \qquad x \in X, \tag{3.6}$$

where $f_i : X \to \mathbb{Z}_+$ are linear functions, for $i = 1, \ldots, m$, $X \subseteq \{0,1\}^d$ and m is a fixed number. This again fits into our framework given by (3.3), with $c = m$. Thus our technique yields an FPTAS for the problem given by (3.6) as well. We have therefore established the following corollary to Theorem 3.3.3.

Corollary 3.5.2 *There is an FPTAS for the problem given by* (3.6) *if m is fixed and if the following exact problem can be solved in pseudo-polynomial time: Given $(c_1, \ldots, c_d) \in \mathbb{Z}_+^d$ and $K \in \mathbb{Z}_+$, does there exist $x \in X$ such that $\sum_{i=1}^d c_i x_i = K$?*

3.6 FPTASes for Robust Weighted Multi-Objective Optimization Problems

Consider the following robust version of a weighted multi-objective optimization problem given by Hu and Mehrotra (2010):

$$\text{minimize } g(x) = \max_{w \in W} w^T f(x), \qquad x \in X \subseteq \{0,1\}^d. \tag{3.7}$$

Here, $f(x) = (f_1(x), \ldots, f_m(x)) \in \mathbb{R}_+^m$ is a vector of m function values, $W \subseteq W_f$, where $W_f = \{w \in \mathbb{R}_+^m : w_1 + \ldots + w_m = 1\}$ (i.e. the weights are non-negative and they sum up to one) and W is a compact convex set. We assume that we can optimize a linear function over the set W in polynomial time; this ensures that the function $g(x)$ can be computed efficiently. Examples of some of the forms that the weight set W can take are as follows:

1. Simplex weight set: $W = W_f = \{w \in \mathbb{R}_+^m : w_1 + \ldots + w_m = 1\}$.

2. Ellipsoidal weight set: $W = \{w \in W_f : (w - \hat{w})^T S^{-1}(w - \hat{w}) \leq \gamma^2\}$, where $\hat{w}, \gamma > 0$, and S is a $m \times m$ positive semi-definite matrix.

3. Box weight set: $W = \{w \in W_f : ||w||_\infty \leq k\}$, where $k > 0$.

In particular, the model with the simplex weight set can be considered to be a generalization of the robust optimization model with a fixed number of scenarios. The robust optimization with a fixed number of scenarios has the following form.

$$\text{minimize } h(x) = \max_{c \in \{c_1,\ldots,c_m\}} c^T x, \qquad x \in X \subseteq \{0,1\}^d. \tag{3.8}$$

The connection between the problems given by (3.7) and (3.8) is established in the following lemma.

Lemma 3.6.1 *The problem given by* (3.8) *is equivalent to the problem given by* (3.7) *when* $f_i(x) = c_i^T x$ *for* $i = 1, \ldots, m$ *and the weight set is the simplex weight set* W_f.

Proof. For a given solution $x \in X$, its objective function value $h(x)$ in the formulation (3.8) is given by

$$\begin{aligned} h(x) &= \max\{c^T x : c \in \{c_1, \ldots, c_m\}\} \\ &= \max\{c^T x : c \in \text{conv}(\{c_1, \ldots, c_m\})\} \\ &= \max\{w_1 c_1^T x + \ldots + w_m c_m^T x : w \in W_f\} \\ &= \max\{w^T f(x) : w \in W_f\} \\ &= g(x), \end{aligned}$$

where $g(x)$ is the objective function value in the formulation given by (3.7), conv$(\{c_1, \ldots, c_m\})$ denotes the convex hull of the m points $c_1, \ldots, c_m$ and $f_i(x) = c_i^T x$ for $i = 1, \ldots, m$. This establishes the equivalence between the optimization problem given by (3.7) with the simplex weight set and the optimization problem given by (3.8). □

Using this observation, we establish the NP-hardness of the optimization problem given by (3.7).

Lemma 3.6.2 *The optimization problem given by* (3.7) *is NP-hard for the shortest path problem and the spanning tree problem.*

Proof. The following 2-scenario robust optimization problem is known to be NP-hard for the shortest path problem and the spanning tree problem (Kouvelis and Yu 1997):

$$\text{minimize } h(x) = \max_{c \in \{c_1, c_2\}} c^T x, \qquad x \in X \subseteq \{0,1\}^d. \tag{3.9}$$

Problem (3.9) is equivalent to the form given by (3.7) with $f_1(x) = c_1^T x$, $f_2(x) = c_2^T x$ and $W = \{(w_1, w_2) \in \mathbb{R}_+^2 : w_1 + w_2 = 1\}$. Therefore the optimization problem given by (3.7) is also NP-hard in general. □

Next, we establish that when m, the number of objectives, is fixed, the optimization problem given by (3.7) admits an FPTAS.

Lemma 3.6.3 *There is an optimal solution to* (3.7) *that lies on $P(\pi)$, the Pareto-optimal frontier of the m functions $f_1(x), \ldots, f_m(x)$.*

Proof. Let x^* be the optimal solution to the problem given by (3.7). Suppose x^* is not on the Pareto-optimal front. By definition, there exists $\hat{x} \in P(\pi)$ such that $f_i(\hat{x}) \le f_i(x^*)$ for $i = 1, \ldots, m$. Let $\hat{w} \in W$ be the weight vector which maximizes $w^T f(\hat{x})$. Then,

$$\begin{aligned} g(\hat{x}) &= \hat{w}^T f(\hat{x}) \\ &\le \hat{w}^T f(x^*) \\ &\le \max_{w \in W} w^T f(x^*) = g(x^*). \end{aligned}$$

Hence $\hat{x}$ minimizes $g(x)$ and lies on the Pareto-optimal frontier. □

Lemma 3.6.4 *There is a solution $\hat{x}$ on $P_\epsilon(\pi)$ that is a $(1+\epsilon)$-approximate solution of the optimization problem* (3.7).

Proof. Let x^* be the optimal solution to the problem given by (3.7). By definition of $P_\epsilon(\pi)$, there exists $\hat{x} \in P_\epsilon(\pi)$ such that $f_i(\hat{x}) \le (1+\epsilon) f_i(x^*)$ for $i = 1, \ldots, m$. Let $\hat{w} \in W$ be the weight which maximizes $w^T f(\hat{x})$. Therefore,

$$\begin{aligned} g(\hat{x}) &= \hat{w}^T f(\hat{x}) \\ &\le (1+\epsilon) \hat{w}^T f(x^*) \\ &\le (1+\epsilon) \max_{w \in W} w^T f(x^*) = (1+\epsilon) g(x^*). \end{aligned}$$

Therefore $\hat{x}$ is a $(1+\epsilon)$ approximate solution to the problem given by (3.7). $\square$

Together with the above two lemmata, we get the following corollary to Theorem 3.3.3, which establishes the existence of FPTASes for the robust version of the shortest path problem, the spanning tree problem and the knapsack problem.

Corollary 3.6.5 *There is an FPTAS for the problem given by* (3.7) *when m is fixed if the following exact problem can be solved in pseudo-polynomial time: Given $(c_1, \ldots, c_d) \in \mathbb{Z}_+^d$ and $K \in \mathbb{Z}_+$, does there exist $x \in X$ such that $\sum_{i=1}^{d} c_i x_i = K$?*

3.7 FPTASes for Problems with Rational Objective Functions

In this section, we consider combinatorial optimization problems involving a ratio of two linear objectives as given in the introduction:

$$\begin{aligned} \text{minimize } g(x) &= \frac{f_1(x)}{f_2(x)} = \frac{a_0 + a_1 x_1 + \ldots + a_d x_d}{b_0 + b_1 x_1 + \ldots + b_d x_d} \\ \text{s.t.} \quad & x \in X \subseteq \{0,1\}^d. \end{aligned} \tag{3.10}$$

We assume that $f_1(x) > 0, f_2(x) > 0$ for all $x \in X$. The situation here is different from the problems we have considered previously, since in this case we are attempting to minimize f_1, while simultaneously maximizing f_2. Therefore we cannot use Theorem 3.3.3 directly for obtaining an FPTAS. We need to modify the definition of the Pareto-optimal front and the approximate Pareto-optimal front for this problem, and re-state the gap theorem for the modified definition. We first give the appropriate definition of the Pareto-optimal and the approximate Pareto-optimal front for this problem.

Definition 3.7.1 *Consider the problem given by* (3.10). *For this problem, the* Pareto-optimal frontier *$P(\pi)$ is the set of all points x for which there is no x' such that $f_1(x') \leq f_1(x)$ and $f_2(x') \geq f_2(x)$ with strict inequality for at least one of them.*

Definition 3.7.2 *For the problem given by* (3.10)*, for $\epsilon > 0$, an* approximate Pareto-optimal frontier *$P_\epsilon(\pi)$ is a set of solutions such that for all $x \in X$, there is $x' \in P_\epsilon(\pi)$ such that $f_1(x') \leq (1+\epsilon) f_1(x)$ and $f_2(x') \geq f_2(x)/(1+\epsilon)$.*

We now state the modified gap theorem for this problem. The proof of this theorem is same as the one for Theorem 2.2.5, so we omit it.

Theorem 3.7.3 (Modified gap theorem) *Let $\epsilon, \epsilon'_1, \epsilon'_2 > 0$ be such that $(1-\epsilon'_1)(1+\epsilon)^{1/2} = 1$ and $(1+\epsilon'_2) = (1+\epsilon)^{1/2}$. One can determine a $P_\epsilon(\pi)$ in time polynomial in $|\pi|$ and $1/\epsilon$ if the following 'gap problem' can be solved in polynomial time: Given a vector of values (v, w), either*
(i) *return a solution $x \in X$ such that $f_1(x) \leq v$ and $f_2(x) \geq w$, or*
(ii) *assert that there is no $x \in X$ such that $f_1(x) \leq (1-\epsilon'_1)v$ and $f_2(x) \geq (1+\epsilon'_2)w$.*

It is easy to see that Lemma 3.3.1 holds in this case, with the modified definition of the Pareto-optimal front. The analog of Lemma 3.3.2 is given below.

Lemma 3.7.4 *Let $P_\epsilon(\pi)$ denote the approximate Pareto-optimal front of the functions f_1 and f_2 corresponding to minimizing f_1 and maximizing f_2. Let $\hat{x}$ be the solution in $P_\epsilon(\pi)$ that minimizes $g(x)$ over all points $x \in P_\epsilon(\pi)$. Then $\hat{x}$ is a $(1+\epsilon)^2$-approximate solution for* (3.10).

Proof. Let x^* be an optimal solution of (3.10) that is in $P(\pi)$. By the definition of an ϵ-approximate Pareto-optimal frontier, there exists $x' \in P_\epsilon(\pi)$ such that $f_1(x') \leq (1+\epsilon)f_1(x^*)$ and $f_2(x') \geq (1+\epsilon)^{-1}f_2(x^*)$. Therefore,

$$g(x') \leq \frac{(1+\epsilon)f_1(x^*)}{(1+\epsilon)^{-1}f_2(x^*)} = (1+\epsilon)^2 g(x^*).$$

Since $\hat{x}$ is a minimizer of $g(x)$ over all the solutions in $P_\epsilon(\pi)$, $g(\hat{x}) \leq g(x') \leq (1+\epsilon)^2 g(x^*)$. $\square$

The following theorem is an analog of Theorem 3.3.3 for this case.

Theorem 3.7.5 *There is an FPTAS for the problem given by* (3.10) *if the following exact problem can be solved in pseudo-polynomial time: Given $(c_1, \ldots, c_d) \in \mathbb{Z}^d_+$ and $K \in \mathbb{Z}_+$, does there exist $x \in X$ such that $\sum_{i=1}^d c_i x_i = K$?*

We give a proof of this theorem here, as it involves both maximization and minimization of the underlying objective functions.

Proof. From Theorem 3.7.3, it suffices to give a polynomial time algorithm to solve the gap problem. Suppose we want to solve the gap problem for the 2-vector (v_1, v_2). Let $r_1 = \lceil d/\epsilon'_1 \rceil$. We first define a "truncated" objective function. For all $j = 1, \ldots, d$, if for some j, $a_j > v_1$, we set $x_j = 0$, and drop the variable x_j from each of the objective functions. Let V be the index set of the remaining variables. Thus, the remaining coefficients in f_1 are now less than or equal to v_1. Next, we define a new objective function $f'_1(x) = \sum_{j \in V} a'_j x_j$, where $a'_j = \lceil a_j r_1/v_1 \rceil$. In the new objective function, the maximum value of a coefficient is now r_1. For $x \in X$, the following two statements hold by Lemma 3.8.1.

- If $f'_1(x) \leq r_1$, then $f_1(x) \leq v_1$.
- If $f_1(x) \leq v_1(1 - \epsilon'_1)$, then $f'_1(x) \leq r_1$.

For f_2, we do the following. Let $r_2 = \lfloor d/\epsilon'_2 \rfloor$. Let $f'_2(x) = \sum_{j \in V} b'_j x_j$, where $b'_j = \min(r_2, \lfloor b_j r_2 / v_2 \rfloor)$. So in f'_2, all the coefficients are no more than r_2. The following two statements hold by Lemma 3.8.2.

- If $f'_2(x) \geq r_2$ then $f_2(x) \geq v_2$.
- If $f_2(x) \geq (1 + \epsilon'_2) v_2$ then $f'_2(x) \geq r_2$.

Therefore, to solve the gap problem, it suffices to find an $x \in X$ such that $f'_1(x) \leq r_1$ and $f'_2(x) \geq r_2$, or assert that no such x exists. There are $r_1 + 1$ ways in which $f'_1(x) \leq r_1$ can be satisfied, and there are at most $r_2 d$ ways in which $f'_2(x) \geq r_2$ can be satisfied. Hence there are $O(r_1 r_2 d)$ ways overall in which both the inequalities can be simultaneously satisfied. Suppose we want to find if there is an $x \in X$ such that $f'_i(x) = b_i$ for $i = 1, 2$. This is equivalent to finding an x such that $f'_1(x) + M f'_2(x) = b_1 + M b_2$, where $M = d \cdot \max(r_1, r_2) + 1$ is a number greater than the maximum value that $f'_i(x)$ can take, for $i = 1, 2$. Hence, if we have a pseudo-polynomial time algorithm for solving the exact problem, we can solve the gap problem in polynomial time. $\square$

This theorem implies than we can use our framework to get an FPTAS, for example, for the knapsack problem with a fractional objective. In fact, it is not hard to see that the method can be extended to functions g having the form $f_1 f_2 / f_3 f_4$, or $f_1/f_2 + f_3/f_4$ as well. As long as the number of functions is fixed, we will get an FPTAS for the problem using our framework.

3.7.1 FPTAS for Assortment Optimization Problems

The problem of minimizing a sum-of-ratios form often arises in assortment optimization in the context of retail management. In this section, we obtain FPTASes for two models of the assortment optimization problem: the mixture of logits choice model and the nested logit choice model.

In the assortment optimization problem with the mixture of logits choice model, we have a set of n products indexed by $N = \{1, \ldots, n\}$ and m customer classes indexed by $C = \{1, \ldots, m\}$. The demand of a customer in a customer class $i \in C$ is modeled using multinomial logit choice model with parameters $(v_{i0}, v_{i1}, \ldots, v_{in}) \in \mathbb{R}_+^{n+1}$. v_{i0} denotes the preference of the customer for purchasing no item, and v_{ij} is the preference of the customer to purchase product $j \in N$. If an assortment $S \subseteq N$ is offered to a customer in the class $i \in C$, the probability that the customer purchases a product $j \in N$ is given by

$$p_{ij}(S) = \begin{cases} \dfrac{v_{ij}}{v_{i0} + \sum_{k \in S} v_{ik}} & i \in S, \\ 0 & \text{otherwise}. \end{cases}$$

The profit corresponding to the purchase of an item j is w_j. Therefore the total profit from customer class $i \in C$ when an assortment $S \subseteq N$ is offered to the customer is given by

$$f_i(S) = \sum_{j \in S} p_{ij}(S) w_j = \frac{\sum_{j \in S} w_j v_{ij}}{v_{i0} + \sum_{j \in S} v_{ij}}.$$

Let λ_i denote the fraction of the customers in class i, where $\sum_{i \in C} \lambda_i = 1$. The optimization problem is to find an assortment S that maximizes the objective function

$$g(S) = \sum_{i \in C} \lambda_i f_i(S).$$

Thus, in this case the objective function is a sum of m ratios. This problem is NP-hard even when there are only $m = 2$ customer classes, but admits a PTAS if m is fixed (Rusmevichientong, Shmoys, and Topaloglu 2010). Using our framework, we can get an FPTAS for the case when m is fixed as follows. Let x_j be the variable which is 1 if product $j \in N$ is offered in an assortment, 0 otherwise. The objective function is $g(x) = \sum_{i=1}^m f_{i1}(x)/f_{i2}(x)$, where

$$\begin{aligned} f_{i1}(x) &= \lambda_i \sum_{j=1}^n w_j v_{ij} x_j, \\ f_{i2}(x) &= v_{i0} + \sum_{j=1}^n v_{ij} x_j. \end{aligned}$$

There are no constraints in this problem. The exact problem in this case is, given a vector $c \in \mathbb{Z}_+^n$ and a non-negative integer C, does there exist $x \in \{0,1\}^n$ such that $\sum_{j=1}^n c_j x_j = C$? This is the subset-sum problem which can be solved in pseudo-polynomial time by dynamic programming. Hence we get an FPTAS for the assortment optimization problem with the mixture of logits choice model. We therefore have the following corollary to Theorem 2.2.5.

Corollary 3.7.6 *The assortment optimization problem with the mixture of logits choice model admits an FPTAS when the number of customer classes is fixed.*

In the mixture of logits choice model, the likelihood of choosing between two alternative products is independent of the assortment offered to the customer. This may not necessarily be true in practice. An alternative model which takes care of this anomaly is the nested logit choice model. In this model, there are G partitions of the product set N given by $H_1, \ldots, H_G$, where G is fixed. Assuming that there is only one class of customers, the probability that a customer purchases a product $j \in N$ when offered an assortment $S \subseteq N$ is given by

$$p_j(S) = \begin{cases} \dfrac{v_j}{\left(\displaystyle\sum_{l \in H_g \cap S} v_l\right)^{\alpha_g}} \cdot \dfrac{1}{1 + \displaystyle\sum_{k=1}^{G} \left(\displaystyle\sum_{l \in H_k \cap S} v_l\right)^{1-\alpha_k}} & \text{if } j \in H_g \cap S \text{ for some } g, \\ 0 & \text{otherwise.} \end{cases}$$

Here, $0 \leq \alpha_g \leq 1$ for all $g = 1, \ldots, G$ and $v_l \in \mathbb{Z}_{\geq 0}$ for all $l \in N$. In this model, the likelihood of choosing between two products is independent of the assortment offered if they are in the same partition, but depends on the assortment if they are in different partitions. The probability of not purchasing any product is $p_0(S) = 1/(1 + \sum_{k=1}^{G}(\sum_{l \in H_k \cap S} v_l)^{1-\alpha_k})$.

In the capacitated version of this problem, we also have a constraint $\sum_{i \in S} c_i \leq C$, where $c_i \in \mathbb{Z}_{\geq 0}$ corresponds to the capacity taken up by the product i and $C \in \mathbb{Z}_+$ corresponds to the total capacity available. The objective is to find an assortment S that maximizes $\sum_{j \in S} p_j(S) w_j$ subject to the capacity constraint. Rusmevichientong, Shen, and Shmoys (2009) show that this problem is NP-hard, but admits a PTAS when G is fixed. They also prove that to get an approximate solution of this problem, it suffices to find an approximate solution of the following sum-of-ratios problem:

$$\begin{aligned} \text{maximize} \quad & g(S_1, \ldots, S_G) = \sum_{i=1}^{G} \frac{\sum_{l \in S_i} u_l}{(\sum_{l \in S_i} v_l)^{\alpha_i}} \\ \text{s.t.} \quad & \sum_{i=1}^{G} \sum_{l \in S_i} c_l \leq C, \\ & S_i \subseteq H_i, \quad \text{for all } i = 1, \ldots, G. \end{aligned}$$

Here, $u_l \in \mathbb{Z}_{\geq 0}$ for all $l \in N$. Let x_l be the indicator variable which is 1 if an item $l \in N$ is selected, 0 otherwise. The objective function is $g(x) = \sum_{i=1}^{G} f_{i1}(x)/(f_{i2}(x))^{\alpha_i}$, where

$$\begin{aligned} f_{i1}(x) &= \sum_{l \in H_i} u_l x_l, \\ f_{i2}(x) &= \sum_{l \in H_i} v_l x_l. \end{aligned}$$

Moreover, if $S_i = \emptyset$, then we count 0 for the term $f_{i1}(x)/(f_{i2}(x))^{\alpha_i}$ in the objective function. Note that the denominator in each of the ratios in the this problem is non-linear. However, because each exponent α_i is upper-bounded by 1, we can still use our framework to get an FPTAS for this problem. First, we choose some k of the G sets $S_1, \ldots, S_G$ to be non-empty and the rest of the sets to be empty. Since there are G groups, we will need to do this 2^G times to cover all the possible cases. This does not affect the polynomial running time of our algorithm as G is fixed. Once we choose the k sets, say $S_{i_1}, \ldots, S_{i_k}$ to be non-empty, we construct the Pareto-optimal frontier corresponding to maximizing the k linear functions $f_{i_1 1}, \ldots, f_{i_k 1}$ and minimizing the k linear functions $f_{i_1 2}, \ldots, f_{i_k 2}$. To ensure that each of the k sets $S_{i_1}, \ldots, S_{i_k}$ is non-empty, we set the lower bound for the numerator function corresponding to these groups to be 1 when solving the gap problem (see the proof of Theorem 2.2.5). The underlying set of constraints is given by

$$\begin{aligned} &\sum_{l \in N} c_l x_l \leq C, \\ &x_l \in \{0,1\}, \quad l \in N. \end{aligned}$$

This is the knapsack constraint, and the corresponding exact problem can be solved in pseudo-polynomial time by dynamic programming. Hence we get an FPTAS for the assortment optimization problem in the nested logit choice model with capacity constraints. We therefore have the following corollary to Theorem 3.7.5.

Corollary 3.7.7 *The capacitated assortment optimization problem with nested logit choice model admits an FPTAS when the number of partitions G of the set of products N is fixed.*

3.8 Conclusion

The main contribution of this chapter is a novel framework for designing approximation schemes for combinatorial optimization problems in which several functions are combined into one objective. Using this framework, we design FPTASes for problems arising in scheduling and resource allocation, combinatorial optimization problems with a rational or a product objective function and robust weighted multi-objective optimization problems. Given the versatility of our technique, we believe that it will be applicable to many other combinatorial optimization problems as well.

Appendix

Lemma 3.8.1 *Suppose* $f(x) = \sum_{j=1}^{d} a_j x_j$, $0 \leq a_j \leq v$, $x_j \in \{0,1\}$ *and* $r = \lceil d/\epsilon \rceil$. *Let* $f'(x) = \sum_{j=1}^{d} a'_j x_j$, *where* $a'_j = \lceil a_j r/v \rceil$. *Then,*

1. *If* $f'(x) \leq r$, *then* $f(x) \leq v$.

2. *If* $f(x) \leq v(1-\epsilon)$, *then* $f'(x) \leq r$.

Proof.

1. Given $f'(x) \leq r$,

$$f(x) = \sum_{j=1}^{d} a_j x_j = \frac{v}{r} \sum_{j=1}^{d} \frac{a_j r}{v} x_j \leq \frac{v}{r} \sum_{j=1}^{d} \left\lceil \frac{a_j r}{v} \right\rceil x_j = \frac{v}{r} f'(x) \leq v.$$

2. Since $f(x) \leq v(1-\epsilon)$,

$$\sum_{j=1}^{d} \frac{a_j r}{v} x_j \quad \leq \quad r(1-\epsilon).$$

Rounding up each of the d numbers on the left hand side, we get

$$\begin{aligned} \sum_{j=1}^{d} \left\lceil \frac{a_j r}{v} \right\rceil x_j &\leq r(1-\epsilon) + d \\ \Rightarrow \quad f'(x) &\leq r - \left\lceil \frac{d}{\epsilon} \right\rceil \epsilon + d \\ &\leq r. \end{aligned}$$

□

Lemma 3.8.2 *Suppose* $f(x) = \sum_{j=1}^{d} b_j x_j$, $0 \leq b_j \leq v$, $x_j \in \{0,1\}$ *and* $r = \lceil d/\epsilon \rceil$. *Let* $f'(x) = \sum b'_j x_j$, *where* $b'_j = \min(r, \lfloor b_j r/v \rfloor)$. *Then,*

1. *If* $f'(x) \geq r$, *then* $f(x) \geq v$.

2. *If* $f(x) \geq (1+\epsilon)v$*, then* $f'(x) \geq r$*.*

Proof.

1. Given $f'(x) \geq r$,

$$f(x) = \sum_{j=1}^{d} b_j x_j = \frac{v}{r}\sum_{j=1}^{d} \frac{b_j r}{v} x_j \geq \frac{v}{r}\sum_{j=1}^{r} \left\lfloor \frac{b_j r}{v} \right\rfloor x_j = \frac{v}{r}\sum_{j=1}^{r} b'_j x_j \geq \frac{v}{r} f'(x) \geq v.$$

2. Let V be the index of all the variables x_j such that $x_j = 1$. Suppose $j \in V$ and $b'_j = r$. Then clearly $f'(x) \geq r$. Now assume that for all $j \in V$, $b'_j = \lfloor b_j r/v \rfloor$. Then,

$$\sum_{j \in V} \frac{b_j r}{v} x_j \quad \geq \quad (1+\epsilon)r.$$

Rounding down each of the numbers on the left hand side and together with the assumption that $b'_j = \lfloor b_j r/v \rfloor$, we get

$$\begin{aligned} \sum_{j \in V} \left\lfloor \frac{b_j r}{v} \right\rfloor x_j \quad &\geq \quad (1+\epsilon)r - d \\ \Rightarrow \quad f'(x) \quad &\geq \quad r + \epsilon \left\lceil \frac{d}{\epsilon} \right\rceil - d \\ &\geq \quad r. \end{aligned}$$

□

Chapter 4

Approximation Schemes for Optimizing a Class of Low-Rank Functions Over a Polytope

4.1 Introduction

Non-convex optimization problems are an important class of optimization problems that arise in many practical situations (see e.g. Horst and Pardalos (1995) for a survey). However, unlike their convex counterpart for which efficient polynomial time algorithms are known (see e.g. Nesterov and Nemirovskii (1961)), non-convex optimization problems have proved to be much more intractable. A major impediment to efficiently solving non-convex optimization problems is the existence of multiple local optima in such problems; thus any algorithm which seeks to find a globally optimal solution (or a solution close to a global optimum) must avoid getting stuck in local optima.

In this chapter, we focus on optimizing a special class of non-convex functions, called low-rank functions, over a polytope. Informally speaking, a function has low rank if it depends only on a few linear combinations of the input variables. We present FPTASes for optimizing a very general class of low-rank functions over a polytope. Recall from Section 2.1 that an FPTAS for a minimization (resp. maximization) problem is a family of algorithms such that for all $\epsilon > 0$ there is a $(1+\epsilon)$-approximation (resp. $(1-\epsilon)$-approximation) algorithm for the problem, and the running time of the algorithm is polynomial in the input size of the problem, as well as in $1/\epsilon$.

Throughout this chapter, we use the following definition of a low-rank non-linear function, given by Kelner and Nikolova (2007).

Definition 4.1.1 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be of* rank k*, if there exist k linearly independent vectors $a_1, \ldots, a_k \in \mathbb{R}^n$ and a function $g : \mathbb{R}^k \to \mathbb{R}$ such that $f(x) = g(a_1^T x, \ldots, a_k^T x)$.*

The optimization problem we are attempting to solve is

$$\begin{array}{ll} \min & f(x) = g(a_1^T x, \ldots, a_k^T x) \\ \text{s.t.} & x \in P. \end{array}$$

Here, P is a polytope, and g is a continuous function (this guarantees that a minimum exists). We assume that the optimal value of this program is strictly positive; this is necessary for the notion of approximation considered here to be valid. Recent work on optimization problems of this kind has focused on the special case when g is quasi-concave (see e.g. Porembski (2004), Kelner and Nikolova (2007), Goyal and Ravi (2009)); all of these works exploit the fact that the minimum of a quasi-concave function over a polytope is always attained at an extreme point of the polytope (see e.g. Bertsekas, Nedić, and Ozdaglar (2003)). In contrast, our approximation scheme does not require the assumption of quasi-concavity.

In general, non-linear programming problems of this form are known to be NP-hard. Pardalos and Vavasis (1991) proved that minimizing a quadratic function $f(x) = c^T x + \frac{1}{2} x^T Q x$, where the Hessian Q has just one non-zero eigenvalue which is negative (and hence $f(x)$ is a function of rank two), over a polytope is NP-hard. Subsequently, Matsui (1996) proved that minimizing the product of two strictly positive linear functions over a polytope is NP-hard. Both these hardness results imply that minimizing a rank two function over a polytope is NP-hard. In fact, as we show in Section 4.5, the optimum value of the problem stated above cannot be approximated to within any factor unless P = NP. Therefore, we will need some extra assumptions on the properties of the function g to obtain an approximation scheme for the optimization problem (see Section 4.2.1).

We mention a few classes of non-convex optimization problems that we tackle in this chapter.

1. **Multiplicative programming problems:** In this case, g has the form $g(y_1, \ldots, y_k) = \prod_{i=1}^k y_i$. It is known that such a function g is quasi-concave (Konno and Kuno 1992), and therefore its minimum is attained at an extreme point of the polytope. Multiplicative objective functions also arise in combinatorial optimization problems. For example, consider the shortest path problem on a graph $G = (V, E)$ with two edge weights $a : E \to \mathbb{Z}_+$ and

$b : E \to \mathbb{Z}_+$. In the context of navigation systems, Kuno (1999) discusses the shortest path problem with the objective function $a(P) \cdot b(P)$ (where P is the chosen path), where a corresponds to the edge lengths, and b corresponds to the number of intersections at each edge in the graph. A similar problem is considered by Kern and Woeginger (2007) as well.

2. **Low rank bi-linear forms:** Bi-linear functions have the form $g(x_1, \ldots, x_k, y_1, \ldots, y_k) = \sum_{i=1}^{k} x_i \cdot y_i$. Such functions do not even possess the generalized convexity properties, such as quasi-concavity or quasi-convexity (Al-Khayyal and Falk 1983). Bi-linear programming problems are of two kinds: *separable*, in which x and y are disjunctively constrained, and *non-separable*, in which x and y appear together in a constraint. A separable bi-linear function has the neat property that its optimum over a polytope is attained at an extreme point of the polytope, and this fact has been exploited for solving such problems (see e.g. Konno (1976)). The non-separable case is harder, and it requires considerably more effort for solving the optimization problem (Sherali and Alameddine 1992). In this chapter, we investigate the particular case when the number of bi-linear terms, k, is fixed.

3. **Sum-of-ratios optimization:** Sum-of-ratios functions have the form $g(x_1, \ldots, x_k, y_1, \ldots, y_k) = \sum_{i=1}^{k} x_i / y_i$. Even for the case of the sum of a linear term and a ratio of two linear terms, the function can have many local optima (Schaible 1977). Further, Matsui (1996) has shown that optimizing functions of this form over a polytope is an NP-hard problem. Problems of this form arise, for example, in multi-stage stochastic shipping problems where the objective is to maximize the profit earned per unit time (Falk and Palocsay 1992). For more applications, see the survey paper by Schaible and Shi (2003) and the references therein.

There are other functions which do not fall into any of the categories above, but for which our framework is applicable; an example is aggregate utility functions (Eisenberg 1961).

Before proceeding further, we state the computational model we are assuming for our algorithmic results to hold:

- The vectors $a_1, \ldots, a_k$ are known to us (i.e. they are part of the input).
- We are given a polynomial time oracle to compute the function g.
- For the polytope P, we have a polynomial time separation oracle.

Our results: The main contributions of this chapter are as follows.

1. **FPTAS for minimizing low rank functions over a polytope:** We give an FPTAS for minimizing a low-rank function f over a polytope under very general conditions (Section 4.2.1). Even though we present our results only for the case of minimization, the method has a straightforward extension for maximization problems as well. The running time of our approximation scheme is exponential in k, but polynomial in $1/\epsilon$ and all other input parameters. Our algorithm relies on deciding feasibility of a polynomial number of linear programs. We emphasize here that this FPTAS does not require quasi-concavity of the function f. To the best of our knowledge, this is the first FPTAS for general non-quasi-concave minimization/non-quasi-convex maximization problems. We then derive approximation schemes for three categories of non-linear programming problems: multiplicative programming (Section 4.3.1), low-rank bi-linear programming (Section 4.3.2) and sum-of-ratios optimization (Section 4.3.3).

2. **Minimizing quasi-concave functions:** For the specific case of quasi-concave minimization, we give an alternative algorithm which returns an approximate solution which is also an extreme point of the polytope P (Section 4.4). Again, this algorithm relies on solving a polynomial number of linear programs, and it can be extended to the case of quasi-convex maximization over a polytope. As an application of our technique, we show that we can get an FPTAS for combinatorial optimization problems in which the objective is a product of a fixed number of linear functions, provided a complete description of the convex hull of the feasible points in terms of linear inequalities is known. For example, this technique can be used to get an FPTAS for the product version and the mean-risk minimization version of the spanning tree problem and the shortest path problem.

3. **Hardness of approximation result:** We show that unless P = NP, it is not possible to approximate the minimum of a positive valued concave function over a polytope to within any factor, even if the polytope is the unit hypercube (Section 4.5). This improves upon the $\Omega(\log n)$ inapproximability result given by Kelner and Nikolova (2007). We first show a similar result for unconstrained minimization of a supermodular set function. Then by using an approximation preserving reduction from supermodular function minimization to minimization of its continuous extension over a unit hypercube, we get the desired result. The hardness result for supermodular function minimization is in contrast with the related problem of submodular function maximization which admits a constant factor approximation algorithm (Feige, Mir-

rokni, and Vondrák 2007). We also give a stronger hardness of approximation result, namely that it is not possible to approximate the minimum of a concave quadratic function (even with just one negative eigenvalue in the Hessian) over a polytope to within any factor, unless P = NP.

The philosophy behind the approximation scheme is to view g as an objective function that combines several objectives ($a_1^T x, \dots, a_k^T x$ in this case) into one. Therefore the idea is to consider the original single-objective optimization problem as a multiple-objective optimization problem. We first construct an *approximate* Pareto-optimal front corresponding to the k linear functions $a_1^T x, \dots, a_k^T x$, and then choose the best solution from this approximate Pareto set corresponding to our objective function as the approximate solution. Constructing the exact Pareto-optimal front for linear functions, in general, is NP-hard, but an approximate Pareto-optimal front can be computed in polynomial time provided k is fixed (Section 2.2). Once we construct an approximate Pareto set, it is possible to compute an approximate solution for a large class of functions g (see Section 4.2 for more details).

Related work: An exhaustive reference on algorithms for non-linear programming can be found in Horst and Pardalos (1995). The case of optimizing low-rank non-linear functions is discussed extensively by Konno, Thach, and Tuy (1996). Konno, Gao, and Saitoh (1998) give cutting plane and tabu search algorithms for minimizing low-rank concave quadratic functions. A more recent work by Porembski (2004) deals with minimizing low-rank quasi-concave functions using cutting plane methods. The methods employed in both papers are heuristic, with no theoretical analysis of the running time of the algorithms, or performance guarantee of the solutions obtained. Vavasis (1992) gives an approximation scheme for low-rank quadratic optimization problems (i.e. the case where the Hessian has only a few non-zero eigenvalues.) However, Vavasis uses a different notion of approximation algorithm than the one we use in this chapter.

A more theoretical investigation of low-rank quasi-concave minimization was done by Kelner and Nikolova (2007), who give an expected polynomial-time smoothed algorithm for this class of functions over integral polytopes with polynomially many facets. They also give a randomized fully-polynomial time approximation scheme for minimizing a low-rank quasi-concave function over a polynomially bounded polytope (i.e. one in which the l_1-norm of every point contained in the polytope is bounded by a polynomial in n, the dimension of the input space), provided a lower bound on the minimum of the quasi-concave function is known a-priori, and the objective function satisfies a Lipschitz condition. Further, they show that it is NP-hard to approximate the general quasi-concave minimization problem by a ratio better than $\Omega(\log n)$ unless P = NP. More recently, Goyal and Ravi (2009) give an FPTAS for minimizing a class of low-rank quasi-concave functions over convex sets. The particular class of low-rank quasi-concave functions which can be optimized using this technique is similar to the one which we deal with in our chapter. Approximation algorithms for minimizing a non-linear function over a polytope without the quasi-concavity assumption have not been studied in the literature so far.

Konno and Kuno (1992) propose a parametric simplex algorithm for minimizing the product of two linear functions over a polytope. Benson and Boger (1997) give a heuristic algorithm for solving the more general linear multiplicative programming problem, in which the objective function can be a product of more than two linear functions. Survey articles for solving multiplicative programming problems can be found in the books by Horst and Pardalos (1995) and Konno, Thach, and Tuy (1996). For the case of combinatorial optimization problems with a product of two linear functions, Kern and Woeginger (2007) and Goyal, Genc-Kaya, and Ravi (2011) give an FPTAS when the description of the convex hull of the feasible solutions in terms of linear inequalities is known. However, the results in both the papers do not generalize to the case when the objective function is a product of more than two linear functions. In contrast, our results easily generalize to this case as well.

For separable bi-linear programming problems, Konno (1976) gives a cutting plane algorithm that returns an approximate locally optimal solution. Al-Khayyal and Falk (1983) handle the non-separable case using branch-and-bound, and they showed that their algorithm is guaranteed to converge to a globally optimal solution of the optimization problem. Another method for solving the non-separable case is the reformulation-linearization technique due to Sherali and Alameddine (1992). This technique is similar to the lift-and-project method for solving mixed integer programs: The algorithm first generates valid quadratic constraints by taking pairwise products of the constraints, then linearizes both the valid quadratic constraints and the bi-linear term to obtain a lower bounding linear program, and finally uses branch-and-bound to solve the resulting reformulation. Minimizing bi-linear functions of low-rank using a parametric simplex algorithm is discussed in the book by Konno, Thach, and Tuy (1996), however their algorithm works for the separable case only. From a theoretical point of view, an advantage of our technique, as compared to most of the existing algorithms in the literature, is that it works equally well for both separable as well as non-separable bi-linear programming problems.

A good reference for algorithms for solving the sum-of-ratios optimization problem is the survey paper by Schaible and Shi (2003). Almost all the existing algorithms for optimizing the sum of ratios of linear functions are heuristic, with no provable bounds on the running time of the algorithm, nor on the quality of the solution obtained. A common approach for solving these problems is to linearize the objective function by introducing a parameter for each ratio in the objective (see e.g. Falk and Palocsay (1992)). In contrast, our algorithm does not need to parametrize the objective function. We give the first FPTAS for this problem, when the number of ratios is fixed. Our algorithm is especially suited for the case where the number of ratios is small, but each ratio depends on several variables.

4.2 The Approximation Scheme

Recall the optimization problem given in Section 4.1.

$$\begin{aligned} \min \quad & f(x) = g(a_1^T x, \ldots, a_k^T x) \\ \text{s.t.} \quad & x \in P. \end{aligned} \tag{4.1}$$

We further assume that the following conditions are satisfied:

1. $g(y) \leq g(y')$ for all $y, y' \in \mathbb{R}_+^k$ such that $y_i \leq y_i'$ for all $i = 1, \ldots, k$,
2. $g(\lambda y) \leq \lambda^c g(y)$ for all $y \in \mathbb{R}_+^k$, $\lambda > 1$ and some constant c, and
3. $a_i^T x > 0$ for $i = 1, \ldots, k$ over the given polytope.

There are a number of functions g which satisfy conditions 1 and 2, for example the l_p norms (with $c = 1$), bi-linear functions (with $c = 2$) and the product of a constant number (say p) of linear functions (with $c = p$). Armed with Theorem 2.2.5, we now present an approximation scheme for the problem given by (4.1) under these assumptions. We denote the term $a_i^T x$ by $f_i(x)$, for $i = 1, \ldots, k$. We first establish a connection between optimal (resp. approximate) solutions of (4.1) and the (resp. approximate) Pareto-optimal front $P(\pi)$ (resp. $P_\epsilon(\pi)$) of the multi-objective optimization problem π with objectives $f_1, \ldots, f_k$ over the same polytope.

Before proceeding, we emphasize that the above conditions are not absolutely essential to derive an FPTAS for the general problem given by (4.1). Condition 1 may appear to be restrictive, but it can be relaxed, provided that there is at least one optimal solution of (4.1) which lies on the Pareto-optimal front of the functions $a_1^T x, \dots, a_k^T x$. For example, the sum-of-ratios form does not satisfy this condition, but still we can get an FPTAS for problems of this form (see Section 4.3.3).

4.2.1 Formulation of the FPTAS

Lemma 4.2.1 *There is at least one optimal solution x^* to* (4.1) *such that $x^* \in P(\pi)$.*

Proof. Let $\hat{x}$ be an optimal solution of (4.1). Suppose $\hat{x} \notin P(\pi)$. Then there exists $x^* \in P(\pi)$ such that $f_i(x^*) \leq f_i(\hat{x})$ for $i = 1, \dots, k$. By Property 1 of g, $g(f_1(x^*), \dots, f_k(x^*)) \leq g(f_1(\hat{x}), \dots, f_k(\hat{x}))$. Thus x^* minimizes the function g and is in $P(\pi)$. □

Lemma 4.2.2 *Let $\hat{x}$ be a solution in $P_\epsilon(\pi)$ that minimizes $f(x)$ over all points $x \in P_\epsilon(\pi)$. Then $\hat{x}$ is a $(1+\epsilon)^c$-approximate solution of* (4.1)*; that is, $f(\hat{x})$ is at most $(1+\epsilon)^c$ times the value of an optimal solution to* (4.1).

Proof. Let x^* be an optimal solution of (4.1) that is in $P(\pi)$. By the definition of ϵ-approximate Pareto-optimal front, there exists $x' \in P_\epsilon(\pi)$ such that $f_i(x') \leq (1+\epsilon) f_i(x^*)$, for all $i = 1, \dots, k$. Therefore,

$$\begin{aligned} f(x') = g(f_1(x'), \dots, f_k(x')) &\leq g((1+\epsilon)f_1(x^*), \dots, (1+\epsilon)f_k(x^*)) \\ &\leq (1+\epsilon)^c g(f_1(x^*), \dots, f_k(x^*)) = (1+\epsilon)^c f(x^*), \end{aligned}$$

where the first inequality follows from Property 1 and the second inequality follows from Property 2 of g. Since $\hat{x}$ is a minimizer of $f(x)$ over all the solutions in $P_\epsilon(\pi)$, $f(\hat{x}) \leq f(x') \leq (1+\epsilon)^c f(x^*)$. □

Recall from Theorem 2.2.5 that it is possible to construct $P_\epsilon(\pi)$ in polynomial time if the gap problem corresponding to the k functions $f_1, \dots, f_k$ can be solved in polynomial time. When the functions f_i are all linear, the gap problem corresponds to checking the feasibility of a linear program, which can be solved in polynomial time. Hence we get an approximation scheme for solving the problem given by (4.1). This is captured in the following theorem.

Theorem 4.2.3 *The gap problem corresponding to the multi-objective version of the problem given by* (4.1) *can be solved in polynomial time. Therefore, there exists an FPTAS for solving* (4.1)*, assuming Conditions 1-3 are satisfied.*

Proof. Solving the gap problem corresponds to checking the feasibility of the following linear program:

$$a_i^T x \leq (1-\epsilon')v_i, \quad \text{for} \quad i = 1, \ldots, k, \tag{4.2a}$$

$$x \in P. \tag{4.2b}$$

If this linear program has a feasible solution, then any feasible solution to this LP gives us the required answer to question (i). Otherwise, we can answer question (ii) in the affirmative. The feasibility of the linear program can be checked in polynomial time under the assumption that we have a polynomial time separation oracle for the polytope P (Grötschel, Lovász, and Schrijver 1988). The existence of the FPTAS follows from Lemma 4.2.1 and Lemma 4.2.2. $\square$

4.2.2 Outline of the FPTAS

The FPTAS given above can be summarized as follows.

1. Sub-divide the space of objective function values $[m, M]^k$ into hypercubes, such that in each dimension, the ratio of two successive divisions is $1 + \epsilon''$, where $\epsilon'' = (1+\epsilon)^{1/2c} - 1$.

2. For each corner of the hypercubes, solve the gap problem as follows, and keep only the set of non-dominated solutions obtained from solving each of the gap problems.

   (a) Check the feasibility of the LP given by (4.2a)-(4.2b).

   (b) If this LP is infeasible, do nothing. If feasible, then include the feasible point of the LP in the set of possible candidates for points in the approximate Pareto-optimal front.

3. Among the non-dominated points computed in Step 2, pick the point which gives the least value of the function f, and return it as an approximate solution to the given optimization problem.

The running time of the algorithm is $O\big((\frac{\log(M/m)}{\epsilon})^k \cdot LP(n, |\pi|)\big)$, where $LP(n, |\pi|)$ is the time taken to check the feasibility of a linear program in n variables and input size of $|\pi|$ bits. This is polynomial in the input size of the problem provided k is fixed. Therefore when the rank of the input function is a constant, we get an FPTAS for the problem given by (4.1).

4.3 Applications of the Approximation Scheme

Using the general formulation given in Section 4.2.1, we now give approximation schemes for three categories of optimization problems: multiplicative programming, low-rank bi-linear programming and sum-of-ratios optimization.

4.3.1 Multiplicative Programming Problems

Consider the following multiplicative programming problem for a fixed k:

$$\begin{aligned} \min \quad & f(x) = (a_1^T x) \cdot (a_2^T x) \cdot \ldots \cdot (a_k^T x) \\ \text{s.t.} \quad & x \in P. \end{aligned} \tag{4.3}$$

We assume that $a_i^T x > 0$, for $i = 1, \ldots, k$, over the given polytope P. In our general formulation, this corresponds to $g(y_1, \ldots, y_k) = \prod_{i=1}^k y_i$ with $c = k$. $f(x)$ has rank at most k in this case. Thus, we get the following corollary to Theorem 4.2.3.

Corollary 4.3.1 *Consider the optimization problem given by* (4.3)*, and suppose that k is fixed. Then the problem admits an FPTAS if $a_i^T x > 0$ for $i = 1, \ldots, k$ over the given polytope P.*

It should be noted that the function f given above is quasi-concave, and so it is possible to get an FPTAS for the optimization problem given by (4.3) which always returns an extreme point of the polytope P as an approximate solution (see Section 4.4).

4.3.2 Low Rank Bi-Linear Programming Problems

Consider a bi-linear programming problem of the following form for a fixed k.

$$\begin{aligned} \min \quad & f(x, y) = c^T x + d^T y + \sum_{i=1}^{k} (a_i^T x) \cdot (b_i^T y) \\ \text{s.t.} \quad & Ax + By \leq h. \end{aligned} \tag{4.4}$$

where $c, a_i \in \mathbb{R}^m$, $d, b_i \in \mathbb{R}^n$, $A \in \mathbb{R}^{l \times m}$, $B \in \mathbb{R}^{l \times n}$ and $h \in \mathbb{R}^l$. $f(x, y)$ has rank at most $2k + 1$. We have the following corollary to Theorem 4.2.3.

Corollary 4.3.2 *Consider the optimization problem given by* (4.4)*, and suppose that k is fixed. Then the problem admits an FPTAS if $c^T x > 0$, $d^T y > 0$ and $a_i^T x > 0$, $b_i^T y > 0$ for $i = 1, \dots, k$ over the given polytope $Ax + By \leq h$.*

It should be noted that our method works both in the separable case (i.e. when x and y do not have a joint constraint) as well as in the non-separable case (i.e. when x and y appear together in a linear constraint). For the case of separable bi-linear programming problems, the optimum value of the minimization problem is attained at an extreme point of the polytope, just as in the case of quasi-concave minimization problems. For such problems, it is possible to obtain an approximate solution which is also an extreme point of the polytope, using the algorithm given in Section 4.4.

4.3.3 Sum-of-Ratios Optimization

Consider the optimization of the following rational function over a polytope.

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{k} \frac{f_i(x)}{g_i(x)} \\ \text{s.t.} \quad & x \in P. \end{aligned} \tag{4.5}$$

Here, $f_1, \dots, f_k$ and $g_1, \dots, g_k$ are linear functions whose values are positive over the polytope P, and k is a fixed number. This problem does not fall into the framework given in Section 4.1 (the function combining $f_1, \dots, f_k, g_1, \dots, g_k$ does not necessarily satisfy Property 1). However, it is still possible to use our framework to find an approximate solution to this optimization problem. Let $h_i(x) = f_i(x)/g_i(x)$ for $i = 1, \dots, k$. We first show that it is possible to construct an approximate Pareto-optimal front of the functions $h_i(x)$ in polynomial time.

Lemma 4.3.3 *It is possible to construct an approximate Pareto-optimal front $P_\epsilon(\pi)$ of the k functions $h_i(x) = f_i(x)/g_i(x)$ in time polynomial in $|\pi|$ and $1/\epsilon$, for all $\epsilon > 0$.*

Proof. From Theorem 2.2.5, it suffices to show that we can solve the gap problem corresponding to the k functions $h_i(x)$ in polynomial time. Solving the gap problem corresponds to checking the feasibility of the following system:

$$\begin{aligned} h_i(x) &\leq (1-\epsilon')v_i, \quad \text{for } i = 1, \dots, k, \\ x &\in P. \end{aligned}$$

Each constraint $h_i(x) \leq (1-\epsilon')v_i$ is equivalent to $f_i(x) \leq (1-\epsilon')v_i \cdot g_i(x)$, which is a linear constraint as $f_i(x)$ and $g_i(x)$ are linear functions. Hence solving the gap problem reduces to checking the feasibility of a linear program, which can be done in polynomial time under the assumption that we have a polynomial time separation oracle for the polytope P. $\square$

The corresponding versions of Lemma 3.3.1 and Lemma 3.3.2 for the sum-of-ratios minimization problem are given below.

Lemma 4.3.4 *There is at least one optimal solution x^* to* (4.5) *such that x^* is in $P(\pi)$, the Pareto-optimal front of the functions $h_1(x), \dots, h_k(x)$.*

Proof. Suppose $\hat{x}$ is an optimal solution of the problem and $\hat{x} \notin P(\pi)$. Then there exists $x^* \in P(\pi)$ such that $h_i(x^*) \leq h_i(\hat{x})$ for all $i = 1, \dots, k$. Then $f(x^*) = \sum_{i=1}^k h_i(x^*) \leq \sum_{i=1}^k h_i(\hat{x}) \leq f(\hat{x})$. Thus x^* minimizes the function f and is in $P(\pi)$. $\square$

Lemma 4.3.5 *Let $\hat{x}$ be a solution in $P_\epsilon(\pi)$ that minimizes $f(x)$ over all points $x \in P_\epsilon(\pi)$. Then $\hat{x}$ is a $(1+\epsilon)$-approximate solution of the problem* (4.5).

Proof. Let x^* be an optimal solution of (4.5) that is in $P(\pi)$. By definition, there exists $x' \in P_\epsilon(\pi)$ such that $h_i(x') \leq (1+\epsilon)h_i(x^*)$, for all $i = 1, \dots, k$. Therefore,

$$f(x') = \sum_{i=1}^k h_i(x') \leq \sum_{i=1}^k (1+\epsilon)h_i(x^*) \leq (1+\epsilon)f(x^*).$$

Since $\hat{x}$ is a minimizer of $f(x)$ over all the solutions in $P_\epsilon(x)$, $f(\hat{x}) \leq f(x') \leq (1+\epsilon)f(x^*)$. $\square$

The existence of an FPTAS for problem (4.5) now follows from Lemma 4.3.4 and Lemma 4.3.5. We therefore have the following corollary.

Corollary 4.3.6 *Consider the problem given by* (4.5)*, and suppose that k is fixed. Then the problem admits an FPTAS if $f_i(x) > 0$, $g_i(x) > 0$ over the given polytope P.*

4.4 The Special Case of Minimizing Quasi-Concave Functions

The algorithm given in Section 4.2 may not necessarily return an extreme point of the polytope P as an approximate solution of the optimization problem given by (4.1). However, in certain cases it is desirable that the approximate solution we obtain is also an extreme point of the polytope.

For example, suppose P describes the convex hull of all the feasible solutions of a combinatorial optimization problem, such as the spanning tree problem. Then an algorithm that returns an extreme point of P as an approximate solution can be used directly to get an approximate solution for the combinatorial optimization problem with a non-linear objective function as well. In this section, we demonstrate such an algorithm for the case when the objective function is a quasi-concave function, which we define below.

Definition 4.4.1 *A function $f : \mathbb{R}^n \to \mathbb{R}$ is quasi-concave if for all $\lambda \in \mathbb{R}$, the set $S_\lambda = \{x \in \mathbb{R}^n : f(x) \geq \lambda\}$ is convex.*

It is a well known result that the minimum of a quasi-concave function over a polytope is attained at an extreme point of the polytope (see e.g. Bertsekas, Nedić, and Ozdaglar (2003)). In fact, for this case, it is also possible to get an approximate solution of the problem which is an extreme point of the polytope, a result already given by Goyal and Ravi (2009). We can get a similar result using our framework, by employing a different algorithm that uses the concept of approximate convex Pareto set, instead of approximate Pareto-optimal front. Recall the definition of an approximate convex Pareto-optimal front from Section 2.2.

Definition 4.4.2 *Let π be an instance of a multi-objective minimization problem. For $\epsilon > 0$, an* ϵ-approximate convex Pareto-optimal set, *denoted by $CP_\epsilon(\pi)$, is a set of solutions, such that for all $x \in X$, there is $x' \in conv(CP_\epsilon(\pi))$ such that $f_i(x') \leq (1+\epsilon)f_i(x)$, for all i.*

Before giving an algorithm for computing a particular approximate convex Pareto-optimal set, we first give some intuition about the structure of the convex Pareto-optimal set. The Pareto-optimal front $P(\pi)$ corresponds to the solutions of the weighted linear program $\min \sum_{i=1}^{k} w_i f_i(x)$ over the polytope P, for all weight vectors $w \in \mathbb{R}^k_{\geq 0}$. The solution points in the convex Pareto-optimal set $CP(\pi)$ are the extreme point solutions of these linear programs. Thus one way to obtain a convex Pareto-optimal set would be to obtain the optimal extreme points of the weighted linear program for all non-negative weights w. The idea behind the algorithm for finding an approximate convex Pareto-optimal set $CP_\epsilon(\pi)$ is to choose a polynomial number of such weight vectors, and obtain the corresponding extreme point solutions for the weighted linear programs.

The algorithm for computing CP_ϵ is presented below. Without any loss of generality, for this section we assume that $m = 1/M$. For a positive integer N, let $[N]$ denote the set $\{1, \ldots, N\}$. In steps $2-3$, we compute the weight set $W(U)$, which is a union of k sets $W_j(U)$ for $j = 1, \ldots, k$. In each $W_j(U)$, the jth component is fixed at U, and the other components vary from 1 to U. In steps $4-7$ we compute the weight set $R(M)$, which again is a union of k sets $R_j(M)$ for $j = 1, \ldots, k$. In each $R_j(M)$, the jth component is fixed at 1, while the other components take values in the set $\{2^0, 2^1, \ldots, 2^{2\lceil \log_2 M \rceil}\}$. In steps $7-11$ of the algorithm, the k objective functions are combined together using the two weight sets, and CP_ϵ is then obtained by computing optimal extreme points for all such weighted objective functions over the polytope P.

```
1.  $U \leftarrow \left\lceil \frac{2(k-1)}{\epsilon} \right\rceil$.
2.  For $j = 1, \ldots, k$, $W_j(U) \leftarrow [U]^{j-1} \times \{U\} \times [U]^{k-j}$.
3.  $W(U) \leftarrow \cup_{j=1}^{k} W_j(U)$.
4.  $S(M) \leftarrow \{2^0, 2^1, \ldots, 2^{2\lceil log_2 M \rceil}\}$.
5.  For $j = 1, \ldots, k$, $R_j(M) \leftarrow (S(M))^{j-1} \times \{1\} \times (S(M))^{k-j}$.
6.  $R(M) \leftarrow \cup_{j=1}^{k} R_j(M)$.
7.  $CP_\epsilon \leftarrow \emptyset$.
8.  For each $r \in R(M)$ do
9.      For each $w \in W(U)$ do
10.         $q \leftarrow$ optimal basic feasible solution for $\{\min \sum_{i=1}^{k} r_i w_i (a_i^T x) : x \in P\}$.
11.         $CP_\epsilon \leftarrow CP_\epsilon \cup \{q\}$.
12. Return $CP_\epsilon$.
```

Theorem 4.4.3 (Diakonikolas and Yannakakis (2008)) *The above algorithm yields an approximate convex Pareto-optimal front CP_ϵ corresponding to the k linear functions $a_i^T x$, $i = 1, \ldots, k$, subject to the constraints $x \in P$.*

A sketch of the proof of this theorem is given below for the sake of completeness.

Proof. Let us call a positive valued vector $(v_1, \ldots, v_k)$ α-balanced if for any $i, j \in \{1, \ldots, k\}$, $v_i/v_j \leq \alpha$. A solution x is U-enabled, if it is the optimal solution of the linear program for the objective $\min \sum_{i=1}^{k} w_i a_i^T x$ over the polytope P, where $w \in W(U)$ (Recall from Section 4.4 that

$W(U) = \cup_{j=1}^{k} W_j(U)$, where $W_j(U) = [U]^{j-1} \times \{U\} \times [U]^{k-j}$). Let all the U-enabled solutions be $q^1, \ldots, q^l$, where l is the number of all such solutions.

Lemma 4.4.4 (Diakonikolas and Yannakakis (2008)) *Let $\epsilon > 0$. Suppose that s is on the Pareto-optimal front of the k objectives $a_1^T x, \ldots, a_k^T x$ and is 2-balanced, but not U-enabled. Then there is a convex combination of U-enabled solutions, say s', such that $s'_i \le (1+\epsilon)s_i$ for $i = 1, \ldots, k$.*

Proof. Suppose there is no convex combination of the U-enabled solutions that is within a factor of $1+\epsilon$ from s in all the components. This implies that the following linear program is infeasible.

$$\begin{aligned}
&\sum_{j=1}^{l} \lambda_j q^j \le (1+\epsilon)s,\\
&\sum_{j=1}^{l} \lambda_j = 1,\\
&\lambda_1, \ldots, \lambda_l \ge 0.
\end{aligned}$$

By Farkas' lemma, there exist $w_1, \ldots, w_k$ and v which satisfy the following inequalities.

$$\begin{aligned}
&w \cdot q^j + v \ge 0, \quad j = 1, \ldots, l,\\
&(1+\epsilon) w \cdot s + v < 0,\\
&w \in \mathbb{R}_+^k.
\end{aligned}$$

This can be simplified to the following set of inequalities.

$$\begin{aligned}
&w \cdot q^j > (1+\epsilon) w \cdot s \quad \text{for all } j = 1, \ldots, l,\\
&w \in \mathbb{R}_+^k.
\end{aligned}$$

Thus, in order to obtain a contradiction to our assumption that there is no convex combination of the U-enabled solutions that is within a factor $1+\epsilon$ from s in all the components, it will be sufficient to show that for *any* $w \in \mathbb{R}_+^k$, there is a j such that $w \cdot q^j \le (1+\epsilon) w \cdot s$, which is what we will do in the rest of this proof.

Let $w \in \mathbb{R}_+^k$ be an arbitrary weight vector. Without loss of generality, we can assume that the maximum value of a component of vector w is U (this can be achieved by suitably scaling the components of w). Let w^* be the weight vector given by $w_i^* = \lceil w_i \rceil$ for $i = 1, \ldots, k$. Clearly,

$w^* \in W(U)$. Let q^* be the optimal solution for the objective $\min \sum_{i=1}^{k} w_i^* a_i^T x$ over the polytope P, then q^* is U-enabled. We will show that $w \cdot q^* \leq (1+\epsilon) w \cdot s$, thus achieving the desired contradiction.

Let t be such that $w_t^* = U$. By our choice of w^*, each component of $w^* - w$ is at most 1. Therefore,

$$(w^* - w) \cdot s \leq \sum_{i \in [k] \setminus \{t\}} s_i \leq 2(k-1)s_t \leq \epsilon U s_t \leq \epsilon(w \cdot s),$$

where the second inequality follows from the fact that s is 2-balanced, the third inequality follows from our choice of $U = \lceil 2(k-1)/\epsilon \rceil$, and the last inequality follows from the fact that $s_t \leq \frac{1}{U}(w \cdot s)$ (as each component of w is at most U, by assumption). Therefore, from this chain of inequalities, we get

$$w^* \cdot s \leq (1+\epsilon) w \cdot s.$$

Also, q^* is the optimal solution for the objective $\min \sum_{i=1}^{k} w_i^* a_i^T x$, therefore

$$w^* \cdot q^* \leq w^* \cdot s.$$

Therefore, we get

$$w \cdot q^* \leq w^* \cdot q^* \leq w^* \cdot s \leq (1+\epsilon) w \cdot s.$$

This establishes the desired contradiction, and completes the proof of the lemma. $\square$

Using the above lemma, we can now prove the theorem. Consider any Pareto-optimal solution $s = (s_1, \ldots, s_k)$. The maximum ratio between any two components of s is at most M^2. Therefore, for some $r \in R(M)$, all the components in the vector $(r_1 s_1, \ldots, r_k s_k)$ are within a factor of 2 of each other. Note that $(r_1 s_1, \ldots, r_k s_k)$ is on the Pareto-optimal front of the weighted k objectives $r_1 a_1^T x, \ldots, r_k a_k^T x$. The algorithm of Section 4.4 computes U-enabled solutions for these weighted k objectives for all $r \in R(M)$. The above lemma implies that there is a convex combination of the U-enabled solutions for the weighted objective functions, say s' such that $r_i s_i' \leq (1+\epsilon) r_i s_i$, for $i = 1, \ldots, k$. Equivalently, $s_i' \leq (1+\epsilon) s_i$, implying that the solution s is indeed approximately dominated by some convex combination of the solutions returned by the algorithm. $\square$

For quasi-concave functions, it suffices to consider only the points in $CP_\epsilon(\pi)$ computed using this algorithm to solve the problem given by (4.1). It should be noted that the following theorem holds specifically for the $CP_\epsilon(\pi)$ computing using the above algorithm, and not for any arbitrary $CP_\epsilon(\pi)$.

Theorem 4.4.5 *Consider the optimization problem given by* (4.1)*. If f is a quasi-concave function and satisfies Conditions 1-3 given in Section 4.2, then the set CP_ϵ obtained using the above algorithm contains a $(1+\epsilon)^c$-approximate solution to the optimization problem.*

Proof. The lower envelope of the convex hull of CP_ϵ is an approximate Pareto-optimal front. By Lemma 4.2.2, the approximate Pareto-optimal front contains a solution that is $(1+\epsilon)^c$-approximate. Therefore, to find an approximate solution of the optimization problem, it suffices to find a minimum of the function g over $conv(CP_\epsilon)$. Since f is a quasi-concave function, g is a quasi-concave function as well. Therefore, the minimum of g over $conv(CP_\epsilon)$ is attained at an extreme point of $conv(CP_\epsilon)$, which is in CP_ϵ. Since any point in CP_ϵ is an extreme point of the polytope P (as all the points in CP_ϵ are obtained by solving a linear program over the polytope P as given in the above algorithm), the theorem follows. $\square$

The overall running time of the algorithm is $O\big(k^2(\frac{(k-1)\log M}{\epsilon})^k \cdot LP(n,|\pi|)\big)$, where $LP(n,|\pi|)$ is the time taken to find an optimal extreme point of a linear program in n variables and $|\pi|$ bit-size input. We now discuss a couple of applications of this algorithm for combinatorial optimization problems.

4.4.1 Multiplicative Programming Problems in Combinatorial Optimization

Since the above algorithm always returns an extreme point as an approximate solution, we can use the algorithm to design approximation algorithms for combinatorial optimization problems where a complete description of the convex hull of the feasible set in terms of linear inequalities or a separation oracle is known. For example, consider the following optimization problem.

$$\begin{aligned} \min \quad & f(x) = f_1(x) \cdot f_2(x) \cdot \ldots \cdot f_k(x) \\ \text{s.t.} \quad & x \in X \subseteq \{0,1\}^n. \end{aligned} \tag{4.6}$$

Since the product of k linear functions is a quasi-concave function (Konno and Kuno 1992; Benson and Boger 1997), we can use the above algorithm to get an approximate solution of this problem by minimizing the product function over the polytope $P = conv(X)$. The FPTAS always returns an extreme point of P as an approximate solution, which is guaranteed to be integral. We therefore have the following theorem.

Theorem 4.4.6 *Consider the optimization problem given by* (4.6)*, and assume that a complete description of $P = conv(X)$ (or the dominant of P) is known in terms of linear inequalities or a polynomial time separation oracle. Then if k is fixed, the problem admits an FPTAS.*

Our FPTAS is both simple in description as well as easily generalizable to the case where we have more than two terms in the product, in contrast to the existing results in the literature (Kern and Woeginger 2007; Goyal, Genc-Kaya, and Ravi 2011; Goyal and Ravi 2009).

4.4.2 Mean-risk Minimization in Combinatorial Optimization

Another category of problems for which this framework is applicable is mean-risk minimization problems that arise in stochastic combinatorial optimization (Atamtürk and Narayanan 2008; Nikolova 2010). Let $f(x) = c^T x$, $c \in \mathbb{R}^n$ be the objective function of a combinatorial optimization problem, where as usual $x \in X \subseteq \{0,1\}^n$. Suppose that the coefficients c are mutually independent random variables. Let the vector $\mu \in \mathbb{R}^n_+$ denote the mean of the random variables, and $\tau \in \mathbb{R}^n_+$ the vector of variance of the random variables. For a given solution vector x, the average cost of the solution is $\mu^T x$ and the variance is $\tau^T x$. One way to achieve a trade-off between the mean and the variance of the solution is to consider the following optimization problem.

$$\begin{aligned} \min \quad & f(x) = \mu^T x + c\sqrt{\tau^T x} \\ \text{s.t.} \quad & x \in X \subseteq \{0,1\}^n. \end{aligned} \tag{4.7}$$

Here, $c \geq 0$ is a parameter that captures the trade-off between the mean and the variance of the solution. In this case, $f(x)$ is a concave function of rank two. If we have a concise description of $P = conv(X)$, then we can use the above algorithm to get an FPTAS for the problem. This is captured in the following theorem.

Theorem 4.4.7 *Consider the optimization problem given by* (4.7)*, and assume that a complete description of $P = conv(X)$ (or the dominant of P) is known in terms of linear inequalities or a polynomial time separation oracle. Then the problem admits an FPTAS.*

Again, although an FPTAS for this problem is known (Nikolova 2010), our FPTAS has the advantage of being conceptually simpler than the existing methods.

4.5 Inapproximability of Minimizing a Concave Function over a Polytope

In this section, we show that it is not possible to approximate the minimum of a concave function over a unit hypercube to within any factor, unless P = NP. First, we establish the inapproximability of supermodular function minimization.

Definition 4.5.1 *Given a finite set S, a function $f : 2^S \to \mathbb{R}$ is said to be* supermodular *if it satisfies the following condition:*

$$f(X \cup Y) + f(X \cap Y) \geq f(X) + f(Y), \quad \textit{for all } X, Y \subseteq S.$$

Definition 4.5.2 *A set function $f : 2^S \to \mathbb{R}$ is* submodular *if $-f$ is supermodular.*

In some sense, supermodularity is the discrete analog of concavity, which is illustrated by the continuous extension of a set function given by Lovász (1983). Suppose f is a set function defined on the subsets of S, where $|S| = n$. Then the continuous extension $\hat{f} : \mathbb{R}^n_+ \to \mathbb{R}$ of f is given as follows:

1. $\hat{f}(x) = f(X)$, where x is the $0/1$ incidence vector of $X \subseteq S$.
2. For any other x, there exists a unique representation of x of the form $x = \sum_{i=1}^k \lambda_i a_i$, where $\lambda_i > 0$, and a_i are $0/1$ vectors satisfying $a_1 \leq a_2 \leq \ldots \leq a_k$. Then $\hat{f}(x)$ is given by $\hat{f}(x) = \sum_{i=1}^k \lambda_i f(A_i)$, where a_i is the incidence vector of $A_i \subseteq S$.

The following theorem establishes a direct connection between f and $\hat{f}$.

Theorem 4.5.3 (Lovász (1983)) *f is a supermodular (resp. submodular) function if and only if its continuous extension $\hat{f}$ is concave (resp. convex).*

We first give a hardness result for supermodular function minimization.

Theorem 4.5.4 *Let $f : 2^S \to \mathbb{Z}_+$ be a supermodular function defined over the subsets of S. Then it is not possible to approximate the minimum of f to within any factor, unless P = NP.*

Proof. The proof is by reduction from the E4-Set splitting problem (Håstad 2001). The E4-Set splitting problem is this: given a ground set V, and a collection C of subsets $S_i \subset V$ of size exactly 4, find a partition V_1 and V_2 of V so as to maximize the number of subsets S_i such that both $S_i \cap V_1$ and $S_i \cap V_2$ are non-empty. Let $g : 2^V \to \mathbb{Z}$ be the function such that $g(V')$ is equal to the number of subsets S_i satisfying $V' \cap S_i \neq \emptyset$ and $(V \setminus V') \cap S_i \neq \emptyset$. Then g is a submodular function (g is just the extension of the cut function to hypergraphs), and therefore the function f defined by $f(V') = |C| - g(V') + \epsilon$ is supermodular, where $\epsilon > 0$. Clearly, f is a positive valued function.

Håstad (2001) has shown that it is NP-hard to distinguish between the following two instances of E4-Set splitting:

1. There is a set V' which splits all the subsets S_i, and
2. No subset of V splits more than a fraction $(7/8 + \eta)$ of the sets S_i, for any $\eta > 0$.

For the first case, the minimum value of f is ϵ, whereas for the second case, the minimum is at least $(\frac{1}{8} - \eta)|C|$. Therefore, if we had an α-approximation algorithm for supermodular function minimization, the algorithm would return a set for the first case with value at most $\epsilon\alpha$. Since ϵ is arbitrary, we can always choose ϵ so that $\epsilon\alpha < (\frac{1}{8} - \eta)|C|$, and hence it will be possible to distinguish between the two instances. We get a contradiction, therefore the hardness result follows. □

Using this result, we now establish the hardness of minimizing a concave function over a $0/1$ polytope.

Theorem 4.5.5 *It is not possible to approximate the minimum of a positive valued concave function f over a polytope to within any factor, even if the polytope is the unit hypercube, unless P = NP.*

Proof. Kelner and Nikolova (2007) have given an approximation preserving reduction from minimization of a supermodular function f to minimization of its continuous extension $\hat{f}$ over the $0/1$-hypercube. Thus any γ-approximation algorithm for the latter will imply a γ-approximation algorithm for the former as well. This implies that minimizing a positive valued concave function over a $0/1$-polytope cannot be approximated to within any factor, unless P = NP. □

In fact, a similar hardness of approximation result can be obtained for minimizing a concave quadratic function of rank 2 over a polytope. Pardalos and Vavasis (1991) show the NP-hardness of minimizing a rank 2 concave quadratic function over a polytope by reducing the independent set problem to the concave quadratic minimization problem. In their reduction, if a graph has an independent set of a given size k, then the minimum value of the quadratic function is 0, otherwise the minimum value is a large positive number. This gives the same hardness of approximation result for minimizing a rank 2 quadratic concave function over a polytope.

The two inapproximability results show that in order to get an FPTAS for minimizing a non-convex function over a polytope, we need not only the low-rank property of the objective function, but also additional conditions, such as Property 1 of the function g given in Section 4.2.1.

4.6 Open Problems

We have presented two different ways of obtaining an FPTAS for combinatorial optimization problems with the product objective function (Section 3.5 and Section 4.3.1). Interestingly, it is not known for several combinatorial optimization problems (for example, for the spanning tree problem and the shortest path problem) whether the minimization problem with a product objective function is NP-hard, even if there are only two functions in the product. Kern and Woeginger (2007) conjecture that this problem should be solvable in polynomial time, though no such algorithm has been proposed for this problem yet. On the other hand, the product version of the spanning tree problem and the shortest path problem where the objective is to maximize the product, are NP-hard. Resolving the computational complexity of the product minimization problem remains open.

Chapter 5

Robust Appointment Scheduling

5.1 Introduction

In this chapter, we study the problem of appointment scheduling in a robust optimization framework. The appointment scheduling problem arises in many service operations where customers are served sequentially in a facility, the service times of the customers are uncertain and one needs to assign time slots for serving the customers in advance. A practical setting where this problem arises is in health care services. Modern health care involves the usage of several high cost devices and facilities such as MRI installations, CT scanners and operation rooms. For these facilities, appointment scheduling is vital to ensure a high utilization of the resources as well as a high quality of service (Cayirli and Veral 2003). For example, consider the problem of scheduling surgeries for outpatients in an operation room at a hospital. The information about which surgeries are to be performed on a particular day is known in advance. However, the time needed to perform each surgery can vary. The hospital manager needs to decide in advance the time at which a particular surgery is scheduled to start, and how much duration to assign to that surgery. If the manager assigns a small time interval for a surgery, then it is likely that the realized time of the surgery will exceed its assigned duration, thus delaying the next surgery. The inconvenience and costs resulting from the delay of both the patients and the staff constitute the *overage* cost of that surgery. If on the other hand, the hospital manager assigns an excessively long interval for a surgery, then the surgery may end early and the operation room will be left idle till the next surgery commences. In that case, the hospital incurs *underage* cost, which corresponds to the under-utilization of the resources in the operation room. Therefore, an appointment schedule should achieve the right trade-off between the underage and the overage costs.

A few other service operations where the appointment scheduling problem arises are:

1. Sabria and Daganzo (1989) analyze the operations at a seaport where ships have to be served sequentially in a given order and the time slots for servicing the ships are computed in advance.

2. In certain serial production systems, the lead times at each of the production stages is stochastic. For each stage, we need to compute the planned lead time. There is an inventory holding cost at each stage corresponding to the job at that stage completing early, a tardiness cost corresponding to the job finishing late and a backlogging cast at the end stage for not meeting demand at the scheduled time. The objective is to minimize the average sum of these three costs (Elhafsi 2002).

3. In many project scheduling problems where the duration of each activity is stochastic, one needs to compute a gate for each activity, that is the time before which a project cannot start. There is a cost associated with a particular activity starting later than its scheduled gate, and also a cost if an activity finishes earlier than the gate of the successor activity (Bendavid and Golany 2009).

In the rest of this chapter, we will refer to any task or customer that needs to be scheduled as a job, and the service provider as a facility.

Existing models in the literature for the appointment scheduling problem include queueing models (Wang 1993; Wang 1999), continuous stochastic models (Denton and Gupta 2003; Robinson and Chen 2003; Kandoorp and Koole 2007) and discrete stochastic models (Begen and Queyranne 2011; Begen, Levi, and Queyranne 2008). In the stochastic models, the processing times of the jobs are assumed to be independent random variables, and the objective is to find an appointment schedule that minimizes the expected cost. In all these models, one assumes complete knowledge about the distribution of the processing times of the jobs. However, in many service settings the distributions may not be known accurately, limiting the utility of the stochastic models. There might not be sufficient historical data of the processing times of the jobs to get a reasonable estimate of the probability distributions. Furthermore, because the cost function in the stochastic model is the expectation of a non-linear function of several random variables, the computational cost of finding an optimal schedule is significantly high. As a consequence, the methods employed to solve the problem are usually based on heuristics with no provable bounds on the running time of the algorithm nor on the performance guarantee of the solutions. Many other methods require the use of advanced techniques such as Monte-Carlo simulations or submodular function minimization. Such techniques may not necessarily be practical in many situations.

The drawbacks of the stochastic models mentioned above are not limited to the appointment scheduling problem alone, but are encountered in many problems where stochastic programming is used. Robust optimization is an alternative framework to deal with the drawbacks arising in stochastic programming. In robust optimization, the uncertainty in the input parameters are handled using uncertainty intervals instead of random variables (see e.g. Ben-Tal and Nemirovski (2002), Bertsimas and Sim (2004)). Robust optimization models have been shown to be much more tractable as compared to the corresponding stochastic optimization models. For example, a closely related application of robust optimization is finding optimal policies for inventory management problems (Bertsimas and Thiele 2006).

Our Contributions: The contributions of this chapter can be summarized as follows.

1. **Robust formulation of the problem**: We propose to look at the appointment scheduling problem in a robust optimization framework. For each job we only need the following information: the minimum and the maximum possible time the job will take to complete, the underage cost if the job finishes early, and the overage cost if the job finishes late. The objective in the robust model is to find a schedule for which the cost in the worst case scenario of the realized processing times of the jobs is minimized (Section 5.2.1). We establish certain analytical properties of the robust model, which we use subsequently to find an optimal solution of the robust appointment scheduling problem.

2. **Analysis of global balancing heuristic**: We propose an intuitive heuristic for scheduling jobs called global balancing heuristic (Section 5.3). This heuristic aims to balance the maximum possible underage cost due to a job with the maximum possible overage cost due to that job. We show that this heuristic is in fact optimal when the underage cost for the jobs in the sequence are non-decreasing. The biggest advantage of this heuristic is that it gives a simple, easy to compute closed form solution for the optimal duration assigned to each job. Computational results show that for typical instances of the appointment scheduling problem, the average cost of a robust optimal schedule is within 20% of the average cost of a stochastic optimal solution (Section 5.3.3).

3. **Analysis of the worst case scenarios**: For the above mentioned special case of the appointment scheduling problem, we establish the worst case scenarios for the schedule given by the

global balancing heuristic. Even though the set of realized processing times of the jobs is infinite, we show that there are only a finite number of worst case scenarios that the optimal solution needs to balance between. The worst case scenarios provide further insight into the structure of the optimal solution (Section 5.3.2).

4. **Ordering problem**: Inspired from single machine scheduling problems, we present two non-linear programming formulations for the ordering problem. Using the insights obtained from the formulations, we present a simple heuristic which gives a near-optimal order of the jobs (Section 5.4).

Related Work: An overview of the appointment scheduling problem is given in the review paper by Cayirli and Veral (2003). The existing literature on appointment scheduling can be roughly divided into three categories: queueing models, stochastic optimization models and stochastic models which use notions of discrete convexity, for example, submodular functions over an integer lattice. We discuss the relevant literature for all the three models below.

Wang (1993) proposes a queueing model for the problem, in which the processing times of the jobs are assumed to be independent and identically distributed random variables with exponential distribution. Both static and dynamic problems (i.e. the case when all the information about the jobs is not known in advance) are considered in this model, and an optimal schedule is obtained by solving a set of non-linear equations. In Wang (1999), the model is generalized to the case where the jobs can have different mean processing times. For this model, he shows that the optimal sequence of the execution of the jobs is to process them in the increasing order of their mean processing times.

Denton and Gupta (2003) formulate the problem as a two-stage stochastic linear program, and then use a sequential bounding algorithm to solve the corresponding stochastic optimization problem. They also give general upper bounds on the cost of a schedule which does not depend on the particular distribution of the processing times or the cost parameters of the jobs. Robinson and Chen (2003) use a Monte Carlo integration technique to compute near-optimal solutions for the appointment scheduling problem. They show that an optimal schedule has a “dome shaped” structure. That is, the allowances for the assigned durations for the jobs first increase, and then decrease steadily for jobs in the end of the sequence. They also give heuristics which approximate this dome shaped structure of the optimal schedule. Green, Savin, and Wang (2006) consider the problem of outpatient appointment scheduling in which serving emergency patients is also permitted. They formulate the problem as a dynamic stochastic control problem and establish properties of an optimal policy for real-time scheduling and capacity allocation. Yet another way of computing an appointment schedule is using local search by Kandoorp and Koole (2007), who show that a local search algorithm converges to an optimal schedule. Gupta (2007) considers the problem of optimally sequencing two jobs, and establishes the optimality of an ordering when a stochastic dominance condition holds for the distribution of the processing durations of the two jobs.

In a recent paper, Begen and Queyranne (2011) show that when the processing times of the jobs are discrete random variables with finite integer support, then there is an optimal schedule which is integral (i.e. the assigned starting times of the jobs have integer values in the optimal solution). They also show that under very general conditions, the cost function with respect to an integer appointment schedule is submodular. An optimal solution can then be found using well known algorithms for submodular function minimization (e.g. Iwata (2008), Orlin (2009)). The running time of their algorithm is $O(n^9 p_{\max}^2 \log p_{\max})$, where n is the number of jobs and $p_{\max}$ is the largest integer in the support of the processing time distributions of the jobs. This idea has also been extended to a get a near-optimal schedule for a data driven model (Begen, Levi, and Queyranne 2008), where the processing time distributions of the jobs are not known in advance, but instead one uses the past data on the realized processing times of the jobs to approximate the distributions.

5.2 Model Description

There are n jobs indexed by $1, \dots, n$ which are to be scheduled in this order on a single facility. The processing time of job i is P_i; P_i can be a random variable or an uncertainty interval. An appointment schedule is given by an $n+1$ vector $A = (A_1, \dots, A_{n+1})$, where A_i is the scheduled start time of job i. Job 1 is always assumed to start at time $A_1 = 0$, and A_{n+1} denotes the scheduled end time of job n. Alternatively, an appointment schedule can also be given by an n vector $(a_1, \dots, a_n)$, where a_i is the assigned duration for job i. That is, $a_i = A_{i+1} - A_i$. In this chapter, whenever we refer to an appointment schedule, we mutually refer to both the vector of the scheduled start times of the jobs and the vector of the assigned durations for the jobs.

The jobs are processed as follows. Job i can be started only at time A_i or later. Let p_i be the realized processing time of job i, and C_i the completion time of job i under this realization. If $C_i \leq A_{i+1}$, then job $i+1$ is started at time A_{i+1}, otherwise job i is started at time C_i. Therefore C_i is given by

$$C_i = \max(A_i, C_{i-1}) + p_i. \tag{5.1}$$

If job i finishes before A_{i+1}, it incurs an underage cost of $u_i(A_{i+1} - C_i)$ and we say that job i is *underaged*. On the other hand, if it finishes after A_{i+1}, the job incurs an overage cost of $o_i(C_i - A_{i+1})$ and we say that job i is *overaged*. The total cost of the schedule for a realization $P = (p_1, \ldots, p_n)$ of the processing times of the jobs is given by

$$F(A,P) = \sum_{i=1}^{n} \max(u_i(A_{i+1} - C_i), o_i(C_i - A_{i+1})). \tag{5.2}$$

In the stochastic formulation of the problem, the processing duration of job i is assumed to be a random variable P_i whose distribution is known. The cost of a schedule is then $\mathbb{E}_P[F(A,P)]$, and the objective is to find an appointment schedule A that minimizes the expected cost.

5.2.1 The Robust Model

In the robust version of the problem, for each job i we are given its minimum possible processing time $\underline{p}_i$ and its maximum possible processing time $\overline{p}_i$. The realized processing time of the job is assumed to be in the interval $[\underline{p}_i, \overline{p}_i]$. We use Δ_i to denote $\overline{p}_i - \underline{p}_i$, the length of the uncertainty interval of job i. Let $P = (p_1, \ldots, p_n)$ be a vector of the realized processing time of the n given jobs, and $\mathcal{P}$ denote the set $\prod_{i=1}^{n}[\underline{p}_i, \overline{p}_i]$. Given an appointment schedule $A = (A_1, \ldots, A_n)$ for the n jobs, the cost of this schedule is given by

$$F(A) = \sup_{P \in \mathcal{P}} F(A,P). \tag{5.3}$$

In other words, the cost of a given schedule is the worst-case scenario cost among all the possible realizations of the processing time of the jobs. We first show that for a given appointment schedule, there is a worst case scenario for which the supremum in equation (5.3) is actually attained.

Lemma 5.2.1 *For a given appointment vector A, there exists $P \in \mathcal{P}$ such that $F(A,P) = F(A)$.*

Proof. The function $F(A,P)$ is a continuous function in P (Begen and Queyranne 2011), and $F(A)$ is the supremum of $F(A,P)$ with respect to P over the compact set $\mathcal{P}$. Therefore there must exist a $P \in \mathcal{P}$ such that $F(A,P) = F(A)$. □

The above lemma implies that the $\sup$ in the equation (5.3) can be replaced by $\max$. This helps in establishing that $F(A)$ is a continuous function with respect to A, as shown in the following lemma.

Lemma 5.2.2 *$F(A)$ is a continuous function in A.*

Proof. From the previous lemma,

$$F(A) = \max_{P \in \mathcal{P}} \sum_{i=1}^{n} \max(u_i(A_{i+1} - C_i), o_i(C_i - A_{i+1})). \tag{5.4}$$

All the terms in the summation are themselves bounded continuous piecewise linear functions in A, and hence taking the maximum of the summation with respect to P ensures that $F(A)$ is a continuous function with respect to A. $\square$

Next, we show that for a given problem instance, there always exists an optimal solution.

Lemma 5.2.3 *There exists an appointment vector A^* such that for any other appointment vector A, $F(A^*) \leq F(A)$.*

Proof. Consider the set $\mathcal{K} = [\underline{A}, \overline{A}] \subseteq \mathbb{R}^{n+1}$, where $\underline{A} = (\underline{A}_1, \dots, \underline{A}_{n+1})$ and $\overline{A} = (\overline{A}_1, \dots, \overline{A}_{n+1})$. $\underline{A}_1 = \overline{A}_1 = 0$, and for any $2 \leq i \leq n$, $\underline{A}_i = \sum_{j<i} \underline{p}_i$ and $\overline{A}_i = \sum_{j<i} \overline{p}_i$. If $A \notin \mathcal{K}$, then there exists $A' \in \mathcal{K}$ such that for *any* realization P of the processing times of the jobs, $F(A', P) \leq F(A, P)$, and therefore $F(A') \leq F(A)$ (Begen and Queyranne 2011). Thus, without loss of generality, we can restrict the set of appointment vector to the compact set $\mathcal{K}$. Since $F(A)$ is a continuous function in A by Lemma 5.2.2, therefore its minimum must be attained at some point A^* in the compact set $\mathcal{K}$. $\square$

5.3 The Global Balancing Heuristic

In the appointment scheduling problem, for each job there are two conflicting costs: one is the underage cost that the job can incur if it finishes early and the other is the overage cost that the job (and possibly the jobs following it) can incur if this job finishes late. The intuition behind the global balancing heuristic is to find an appointment schedule that balances between these two costs caused *due* to each job. The cost balancing idea is inspired from similar cost balancing policies used previously in inventory management problems (Levi, Pál, Roundy, and Shmoys 2007; Levi, Roundy, Shmoys, and Truong 2008).

Let us denote the schedule generated by this heuristic as A^G. Suppose job i is overaged, then the overage of job i causes overage cost to be incurred not only on job i, but possibly on all the subsequent jobs as well. The maximum possible overage due to job i on itself and all the subsequent jobs is $\overline{p}_i - a_i^G$. Thus the maximum overage cost *due* to job i is $(\sum_{k=i}^{n} o_k)(\overline{p}_i - a_i^G)$. The maximum possible underage cost due to job i is $u_i(a_i^G - \underline{p}_i)$. Equating the two costs, we get

$$a_i^G = \frac{u_i \underline{p}_i + o_{\geq i}\overline{p}_i}{u_i + o_{\geq i}}, \tag{5.5}$$

where $o_{\geq i} = \sum_{k=i}^{n} o_k$. The following lemma gives an upper bound on the cost of this schedule.

Lemma 5.3.1 *An upper bound on the cost of the schedule A^G is given by*

$$F(A^G) \leq \sum_{i=1}^{n} \frac{u_i o_{\geq i} \Delta_i}{u_i + o_{\geq i}}.$$

Proof. The maximum possible underage cost of job i is $u_i(a_i^G - \underline{p}_i)$, and the maximum possible overage cost due to job i on itself and the subsequent jobs is $o_{\geq i}(\overline{p}_i - a_i^G)$. Therefore the maximum possible cost due to job i has an upper bound

$$F_i(A^G) \leq \max(u_i(a_i^G - \underline{p}_i), o_{\geq i}(\overline{p}_i - a_i^G)) \leq \frac{u_i o_{\geq i} \Delta_i}{u_i + o_{\geq i}}.$$

The second inequality holds because from (5.5), it follows that $u_i(a_i^G - \underline{p}_i) = o_{\geq i}(\overline{p}_i - a_i^G) = u_i o_{\geq i} \Delta_i/(u_i + o_{\geq i})$.

Hence an upper bound on the cost of the schedule A^G is

$$F(A^G) \leq \sum_{i=1}^{n} \frac{u_i o_{\geq i} \Delta_i}{u_i + o_{\geq i}}.$$

$\square$

5.3.1 Analysis of the Global Balancing Heuristic

In this section, we analyze the special case when the underage cost of all the jobs is the same. For this case, we show that the global balancing heuristic gives an optimal schedule, and the result holds for the more general case when the underage costs of the jobs are non-decreasing. Before proceeding, we define some terminologies which we use in the rest of the chapter. For a given scenario of realized processing times of the jobs, we say that job i is of max-length if its realized processing time is $\overline{p}_i$, and it is of min-length if its realized processing time is $\underline{p}_i$.

We first give upper and lower bounds on the assigned duration of a job in an optimal schedule when the underage cost of all the jobs are equal. These bounds hold for an optimal appointment schedule for both the stochastic as well as the robust model.

Lemma 5.3.2 *If all the jobs have the same underage cost, then for an optimal appointment schedule A, $\underline{p}_i \leq a_i \leq \overline{p}_i$.*

Proof. The first part of the inequality was proved by Begen and Queyranne (2011). For the second part, assume on the contrary that in an optimal solution A, for job i the assigned duration is greater than $\overline{p}_i$. Let $\delta = A_{i+1} - A_i - \overline{p}_i$. By assumption, $\delta > 0$. We claim that for any $\delta' < \delta$, changing A_i to $A_i + \delta'$ does not increase the cost in any scenario. There are two cases to consider:

Case 1: Job $i-1$ is underaged. In this scenario, job i is underaged as well. If A_i is changed to $A_i + \delta'$, then the overage cost of job $i-1$ increases, but the underage cost of job i decreases by the same amount. This is true as long as $\delta' < \delta$.

Case 2: Job $i-1$ is overaged. Let C_{i-1} be the completion time of job $i-1$. Then $C_{i-1} > A_i$. If $C_{i-1} - A_i \geq \delta'$, then increasing A_i to $A_i + \delta'$ only decreases the overage cost of job $i-1$, and changes nothing else. If $C_{i-1} - A_i < \delta'$, then after increasing A_i to $A_i + \delta'$, job $i-1$ becomes underaged. However, job i remains underaged as well, therefore any increase in the underage cost of job $i-1$ is neutralized by the decrease in the underage cost of job i. The net effect is decrease in the overall cost, as the overage cost that job $i-1$ was incurring in the earlier schedule is no more there in the new schedule.

Thus, in either case, the cost in every realized scenario either remains the same, or decreases upon increasing A_i. This contradicts the assumption that the given schedule A is optimal. Hence the statement of the lemma holds. $\square$

In fact, the above lemma holds for the more general case when u_i is non-decreasing in i, that is $u_i \leq u_{i+1}$ for all $i = 1, \ldots, n-1$. However, it does not hold for the most general case. For example, consider the following instance of the appointment scheduling problem. There are two jobs, with $u_1 = 100, o_1 = 1, u_2 = 1, o_2 = 100, \underline{p}_1 = \underline{p}_2 = 8$ and $\overline{p}_1 = \overline{p}_2 = 10$. The optimal appointment schedule for this instance is $A_1 = 0, A_2 = 8, A_3 = 19.98$ with cost 3.96. Thus the assigned duration for job 2 in this example is $A_3 - A_2 = 11.98$, which is greater than $\overline{p}_2$.

Next, we show that the in an optimal robust appointment schedule, the upper bound for the optimal assigned duration is in fact stronger than the one given above. This bound will be used subsequently in establishing a lower bound on the cost of an appointment schedule. For the sake of simplicity, we prove our results for the special case when $u_i = u$ for all the jobs, however the results hold for the more general case of non-decreasing u_i's as well.

Lemma 5.3.3 *Suppose A is an optimum appointment schedule for a given instance of n jobs. Then for each job i,*

$$u(a_i - \underline{p}_i) \leq o_{\geq i}(\overline{p}_i - a_i),$$

for each $i = 1, \ldots, n$.

Proof. Suppose, on the contrary, the optimal solution satisfies $u(a_i - \underline{p}_i) > o_{\geq i}(\overline{p}_i - a_i)$ for some job i. Consider the solution with the appointment schedule A', where $A'_i = A_i + \epsilon$ and $A'_j = A_j$, for all other j. We choose a value of ϵ small enough so that for the new schedule A', the relation $u(a'_i - \underline{p}_i) > o_{\geq i}(\overline{p}_i - a'_i)$ is still satisfied, and $a_i - \epsilon > \underline{p}_i$. For a given realization P of the processing times of the jobs, there are two cases to consider:

Case 1: Job $i-1$ is overaged in schedule A'. Then job $i-1$ remains overaged in schedule A as well. Clearly, for this case $F(A', P) < F(A, P)$, as the overage cost of job $i-1$ is lower in schedule A', and all other costs remain the same.

Case 2: Job $i-1$ is underaged in schedule A'. This means that job i starts at time A'_i in A'. Consider the scenario P' in which $p'_i = \underline{p}_i$, and $p'_j = p_j$ for all other jobs j. Since we assumed that $u(a'_i - \underline{p}_i) > o_{\geq i}(\overline{p}_i - a'_i)$, therefore $F(A', P) < F(A', P')$. This is because if job i is overaged, the maximum possible increase in cost is $o_{\geq i}(\overline{p}_i - a'_i)$, hence having job i of min-length gives a higher cost. However, for the scenario P', $F(A, P') \geq F(A', P')$, as the increase in the underage cost of job $i-1$ in schedule A' is compensated by the decrease in the underage cost of job i. Therefore we get $F(A', P) < F(A', P') \leq F(A, P') \leq F(A)$, hence the cost of A' in scenario P does not exceed the cost of the schedule A.

Therefore schedule A' has a lower cost than the cost of A in any realized scenario, a contradiction. Hence, for the optimal solution, we must have $u(a_i - \underline{p}_i) \leq o_{\geq i}(\overline{p}_i - a_i)$. □

Using the above lemma, we now prove a lower bound on the cost of an optimal appointment schedule.

Lemma 5.3.4 *Let A be an optimal appointment schedule for a given instance of n jobs. If the underage cost of all the jobs is equal to u, then a lower bound on the cost of the schedule A is given by*

$$F(A) \geq \sum_{i=1}^{n} \frac{u o_{\geq i} \Delta_i}{u + o_{\geq i}}.$$

Proof. Consider a scenario P of the realized processing times of the jobs in which $p_i = \overline{p}_i$, that is each job is of max length in the scenario P. From Lemma 5.3.3, $a_i \leq (u\underline{p}_i + o_{\geq i}\overline{p}_i)/(u + o_{\geq i})$. The contribution of job i in the overage of itself and the subsequent jobs is $\overline{p}_i - a_i$, which in the scenario P is at least $u\Delta_i/(u + o_{\geq i})$. Therefore, the contribution of overage cost of job i in scenario P has a lower bound given by

$$F_i(A, P) \geq \frac{uo_{\geq i}\Delta_i}{u + o_{\geq i}}.$$

Hence a lower bound on the cost of the schedule A is

$$F(A) \geq F(A, P) \geq \sum_{i=1}^{n} \frac{uo_{\geq i}\Delta_i}{u + o_{\geq i}}.$$

□

Thus, we get the following main theorem of this chapter.

Theorem 5.3.5 *The global balancing heuristic gives an optimal solution of the robust appointment scheduling problem when the underage cost of the jobs are equal. The assigned duration to job i in the optimal schedule A^G is given by equation* (5.5)*, and the cost of this schedule is given by*

$$F(A^G) = \sum_{i=1}^{n} \frac{u_i o_{\geq i}\Delta_i}{u + o_{\geq i}}. \tag{5.6}$$

Proof. Follows from Lemma 5.3.1 and Lemma 5.3.4. □

5.3.2 Key Insights

Suppose that job i is the only job that needs to be scheduled. Then it is easy to see that there are two worst-case scenarios for job i: one in which job i is of min-length, and the other in which the job is of max-length. The optimal schedule must balance the cost between these two worst-case scenarios for job i. This yields the optimal assigned duration for job i as

$$a_i = \frac{u_i\underline{p}_i + o_i\overline{p}_i}{u_i + o_i}.$$

Now suppose that job i is followed by jobs $i + 1, \ldots, n$, and underage costs for all the jobs are equal. Then in the optimal solution,

$$a_i = \frac{u_i\underline{p}_i + o_{\geq i}\overline{p}_i}{u_i + o_{\geq i}}.$$

Thus $o_{\geq i}$ acts as the "effective overage cost" for job i in this case. Note also that the optimal assigned duration to job i depends only on the cost parameters of the jobs succeeding the job i, and not on any of the jobs preceding job i.

Next, we establish the worst case scenarios for the optimal solution.

Lemma 5.3.6 *For a given instance of the appointment scheduling problem with n jobs, let $P^1, \ldots, P^{n+1}$ be $n+1$ scenarios of realized processing times of the jobs such that in scenario P^j, the first $j-1$ jobs are all of min-length, and the rest of the jobs are of max-length. Then these are the worst-case scenarios for the optimal robust schedule A^G.*

Proof. Consider the scenario P^j. The first $j-1$ jobs in this scenario are all of min-length, therefore the underage cost due to these $j-1$ jobs is

$$\begin{aligned} F^1(A^G, P^j) &= \sum_{i=1}^{j-1} u_i(a_i^G - \underline{p}_i) \\ &= \sum_{i=1}^{j-1} \frac{u_i o_{\geq i} \Delta_i}{u_i + o_{\geq i}}. \end{aligned}$$

On the other hand, the jobs $j, \ldots, n$ are all of max-length. For $i = j, \ldots, n$, the overage of job i on itself and the subsequent jobs is $(\overline{p}_i - a_i^G)$, leading to an overage cost of $o_{\geq i}(\overline{p}_i - a_i^G)$. Hence the total overage cost due to these jobs is

$$\begin{aligned} F^2(A^G, P^j) &= \sum_{i=j}^{n} o_{\geq i}(\overline{p}_i - a_i^G) \\ &= \sum_{i=j}^{n} \frac{u_i o_{\geq i} \Delta_i}{u_i + o_{\geq i}}. \end{aligned}$$

Hence, the total cost of the schedule A^G in scenario P^j is

$$F(A^G, P^j) = F^1(A^G, P^i) + F^2(A^G, P^i) = F(A^G).$$

Therefore P^j is indeed a worst-case scenario for the optimal solution A^G, for $j = 1, \ldots, n+1$. □

Essentially, the optimal solution achieves a balance between these $n+1$ worst case scenarios. Alternatively, another way of deriving the assigned durations to the jobs given by equation (5.5) is to solve the system of linear equations obtained by equating the cost of the schedule in these $n+1$ worst case scenarios. Figure 5-1 shows the worst case scenarios for the optimal schedule for the case of 3 jobs.

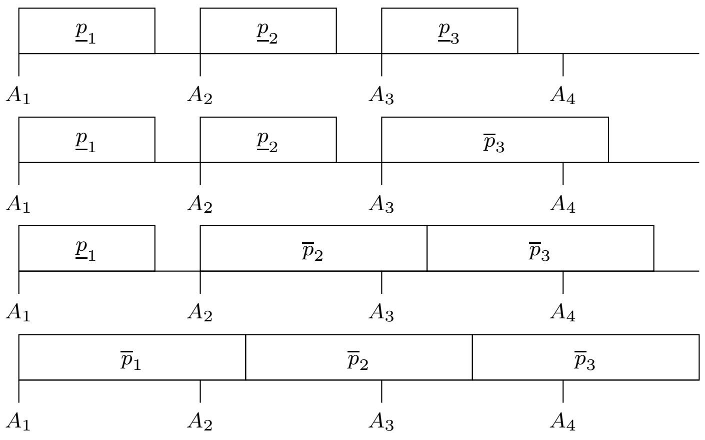

Figure 5-1: Worst-case scenarios for an optimal solution for the case of 3 jobs.

5.3.3 Computational Results

We perform computational experiments to compare the schedules obtained using our robust model and stochastic models. We use the discrete stochastic model of Begen and Queyranne (2011) for comparison. The aim of this section is to show that for typical instances of the appointment scheduling problem, the average cost of a robust optimal schedule is within a reasonable limit of the average cost of a stochastic optimal schedule.

Distribution of the processing times of jobs: We use the data for service times in an MRI facility given by Green, Savin, and Wang (2006). The observed mean duration for serving each patient in this case is 48 minutes, with a standard deviation of 26 minutes. The distribution of service times matches closely with that of a Weibull distribution when the parameters of the Weibull distribution are chosen appropriately. For our computational study, we only consider the case of serving outpatients, and assume that no inpatients or emergency arrivals are served in the facility. We further assume that the processing times of all the jobs are independent and identically distributed random variables.

We use three different distributions in our study: Weibull, Gaussian and Gamma. The mean and the standard deviation for all the distributions are 48 and 26, except for the case when we study the impact of the standard deviation on the performance of the robust model. For the Weibull distribution, the shape parameter is kept fixed at 1.54 for all the cases (same as that given by Green, Savin, and Wang (2006)). The displacement parameter is 43.59 and the scale parameter is 8.77, except for the case where we are studying the impact of the standard deviation. The discrete versions of the distributions are generated over 15 discrete supports ranging from 10 to 150 using the R software program. For the Gaussian distribution, the corresponding discrete version is truncated at the negative tail of the distribution to ensure that the discrete distribution has positive support values.

Cost parameters: For all the instances, the underage cost and the overage cost are assumed to be the same for all the jobs. We fix the overage cost to be 1. In a typical instance of the appointment scheduling problem in health care services, the underage cost is much higher than the overage cost. For our computational experiments, the underage cost of all the jobs is assumed to be 10, except for the case where we study the impact of the underage cost on the performance. In that case, the underage cost varies from 5 to 25.

Computing the stochastic optimal solution: An optimal solution for the discrete stochastic model is computed using a local search algorithm (see Appendix). The algorithm is guaranteed to return an optimal solution, although it may take exponential time to compute an optimal solution. However, for all the instances of the problem considered below, an implementation of the algorithm returned an optimal solution in a reasonable amount of time. Since the optimal schedule in the discrete stochastic model is integral, its cost can be computed efficiently (Begen and Queyranne 2011).

Uncertainty interval for the robust model: For the robust model, the uncertainty interval for each job is assumed to be $[\mu - \sigma, \mu + \sigma]$, where μ is the mean processing time of the job and σ is the standard deviation of the processing time. The robust model gives a schedule which is not necessarily integral, and its average cost is computed by considering all the possible realizations of the processing times of the jobs exhaustively. This turns out to be the main bottleneck in our computational study. Because of this, the maximum number of jobs for which the average cost can be computed in a reasonable amount of time is 9. Except for the case where we study the impact of the number of jobs, the number of jobs for all other cases is fixed at 7.

5.3.4 Comparison of Robust Optimal and Stochastic Optimal Schedules

Figure 5-2 shows the assigned durations for the jobs for the optimal robust schedule and the optimal stochastic schedule for the case of 8 jobs, when the processing times of the jobs have independent and identically distributed Weibull distributions. The stochastic optimal schedule follows roughly a dome like pattern: the assigned durations to the jobs first increase, and then decrease. Such a form of the stochastic optimal schedule has also been reported before in the literature (Robinson and Chen 2003). On the other hand, the assigned duration to the jobs in the robust optimal schedule is steadily decreasing. Such a schedule takes care of the scenario in which the starting jobs may take a longer time to complete: even if these jobs finish later than their assigned deadlines, the impact on the subsequent jobs is expected to be minimal.

5.3.5 Impact of Various Factors on the Average Cost of the Robust Schedule

For each particular case consider below, we plot the relative performance of the robust optimal schedule, which is given by $(ROB - OPT)/OPT \times 100$, where ROB is the average cost of the robust schedule and OPT is the average cost of the stochastic optimal schedule. We compute the relative performance for the following three cases:

1. **Impact of the number of jobs:** This is shown in Figure 5-3. For all the three distributions, the average cost of the robust optimal schedule is within 20% of the stochastic optimal schedule cost. The performance of the robust schedule deteriorates slightly with increase in the number of the jobs.

2. **Impact of the underage cost:** This is shown in Figure 5-4. Similar to the previous case, the

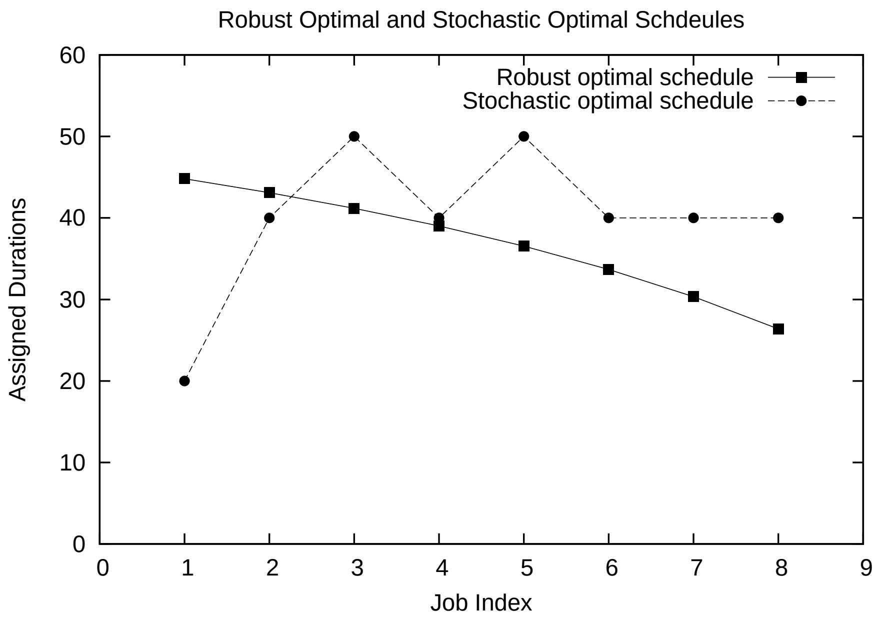

Figure 5-2: Comparing the schedules obtained using robust model and stochastic model.

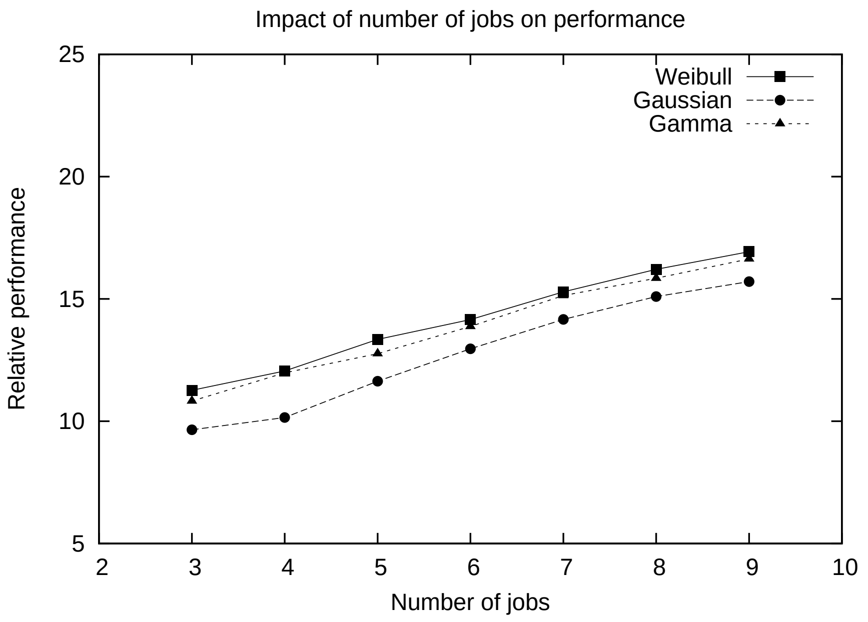

Figure 5-3: Impact of the number of jobs on the relative performance of the robust schedule.

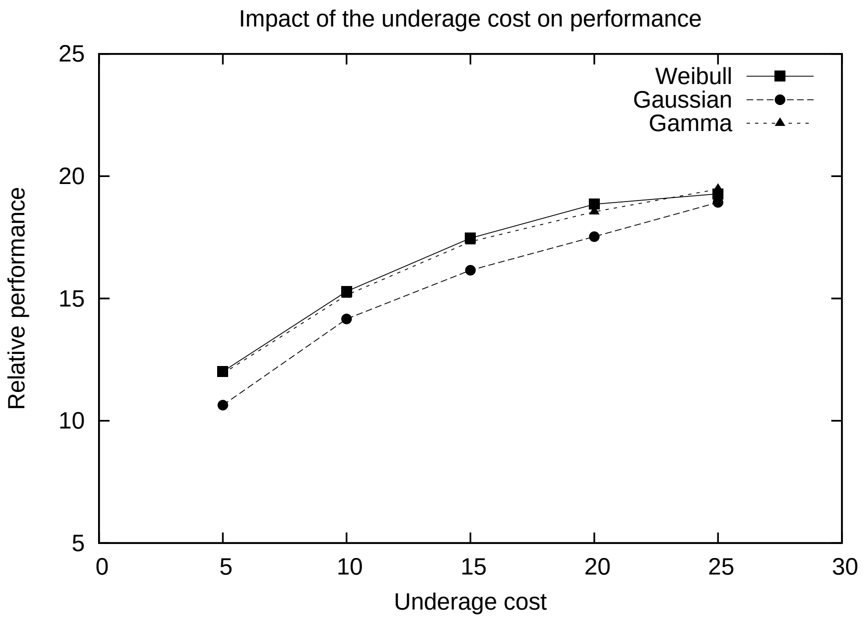

Figure 5-4: Impact of the underage cost on the relative performance of the robust schedule.

average cost of the robust optimal schedule is within 20% of the stochastic optimal schedule cost, and the performance of the robust schedule deteriorates slightly with increase in the underage cost of the jobs.

3. **Impact of the standard deviation:** This is shown in Figure 5-5. Unlike the previous two

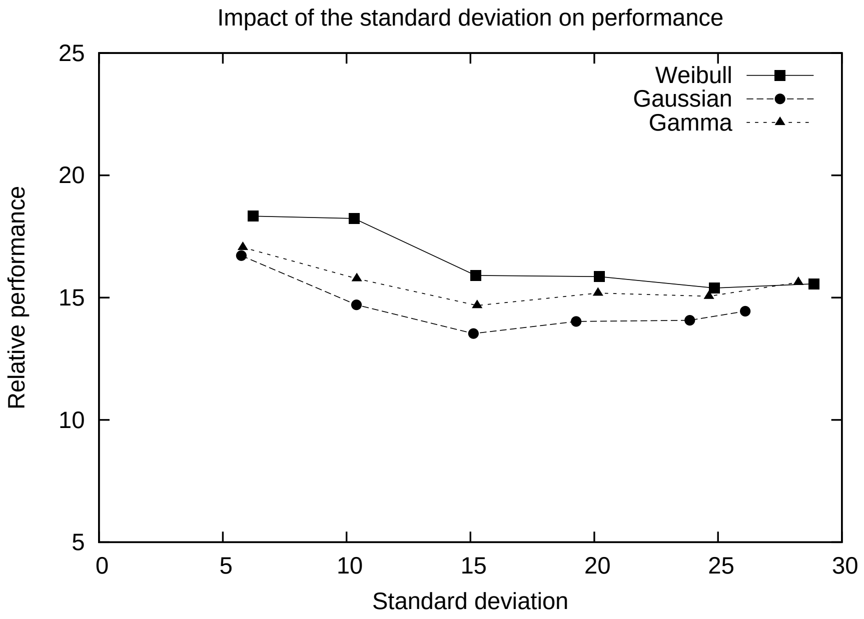

Figure 5-5: Impact of the standard deviation on the relative performance of the robust schedule.

cases, the performance of the robust schedule remains steady (and in fact it decreases slightly) with increase in the standard deviation of the processing times of the jobs.

In all the cases, the average cost of the robust optimal schedule is within 20% of the average cost of the stochastic optimal schedule. This shows that for typical instances of the appointment scheduling problem, the global balancing heuristic gives easy to compute near-optimal schedules.

5.4 The Ordering Problem

In this section, we look at the case where we also have the flexibility of changing the order in which the jobs are processed. Using the closed form for the cost of a schedule given by equation (5.6), we first give two non-linear programming formulations of the ordering problem in which the underlying set of constraints is a polyhedron. Later, we present a simple heuristic which gives us a near-optimal solution for the ordering problem. We only look at the case where all the jobs have the same underage cost, which we assume, without loss of generality, equal to 1.

5.4.1 A Linear Ordering Formulation for the Ordering Problem

For an ordering of the jobs, we can define a vector δ with δ_{ij} equal to 1 if job i precedes job j in a processing sequence, and 0 otherwise ($\delta_{ii} = 1$ for all the jobs i). With this definition of δ, the cost of an ordering can be written as follows.

$$F(\delta) = \sum_{i=1}^{n} \frac{\Delta_i \sum_{j=1}^{n} \delta_{ij} o_j}{1 + \sum_{j=1}^{n} \delta_{ij} o_j}. \tag{5.7}$$

Let $N = \{1, \ldots, n\}$ denote the set of all jobs. The following set of constraints characterize exactly those vectors δ that correspond to a total ordering of the jobs.

$$\begin{aligned} \delta_{ij} + \delta_{ji} &= 1, \quad \text{for all } i, j \in N, i \neq j, & \text{(5.8a)} \\ \delta_{ij} + \delta_{jk} + \delta_{ki} &\geq 1, \quad \text{for all } i, j, k \in N, i \neq j \neq k, & \text{(5.8b)} \\ \delta_{ii} &= 1, \quad \text{for all } i \in N, & \text{(5.8c)} \\ \delta_{ij} &\in \{0, 1\}, \quad \text{for all } i, j \in N. & \text{(5.8d)} \end{aligned}$$

We consider a relaxation of this integer optimization problem in which δ_{ij} can take any value between 0 and 1:

$$0 \leq \delta_{ij} \leq 1, \quad \text{for all } i, j \in N. \tag{5.9}$$

Given a fractional solution δ^* for the relaxation with the objective function (5.7) and constraints (5.8a)-(5.8c), (5.9), we propose a rounding scheme to get an ordering on the jobs. Let $\Theta_i^* = \sum_{j=1}^{n} \delta_{ij}^* o_j$. The jobs are re-numbered so that $\Theta_1^* \geq \Theta_2^* \geq \ldots \geq \Theta_n^*$, and they are executed in this particular order. For this order, let $\Theta_i = \sum_{j=i}^{n} o_j$. An upper bound on Θ_i is given in the following lemma.

Lemma 5.4.1 (Schulz (1996)) *For all* $i = 1, \ldots, n$, $\Theta_i \leq 2\Theta_i^*$.

Proof. For any subset S of the jobs, the following inequality can be derived, similar to that for the sum of completion times of jobs in single machine scheduling (Queyranne 1993).[1]

$$\sum_{j\in S} o_j\Theta_j^* = \frac{1}{2}\left(\sum_{j\in S} o_j\right)^2 + \frac{1}{2}\sum_{j\in S} o_j^2 \geq \frac{1}{2}\left(\sum_{j\in S} o_j\right)^2. \tag{5.10}$$

Using $S = \{i, \ldots, n\}$ in the above inequality, we get

$$\Theta_i = \sum_{j=i}^{n} o_j \leq 2\sum_{j=i}^{n} o_j\Theta_j^* / \sum_{j=i}^{n} o_j \leq 2\Theta_i^*, \tag{5.11}$$

where the second inequality follows from the fact that $\Theta_j^* \leq \Theta_i^*$ for $j \geq i$, by assumption. $\square$

Lemma 5.4.2 *The above algorithm gives an ordering on the jobs whose cost is at most* 2 *times the cost of the fractional solution* δ^*.

Proof. The rounding scheme ensures that for each job i,

$$\sum_{j=1}^{n} \delta_{ij}^* o_j \leq \sum_{j=i}^{n} o_j \leq 2\sum_{j=1}^{n} \delta_{ij}^* o_j.$$

Therefore, for each job i, we get

$$\frac{\sum_{j=i}^{n} o_j}{1+\sum_{j=i}^{n} o_j} \leq \frac{2\sum_{j=1}^{n} \delta_{ij}^* o_j}{1+\sum_{j=1}^{n} \delta_{ij}^* o_j}.$$

Hence the cost of the rounded solution is at most twice that of δ^*. $\square$

Thus, if the sum-of-fractions problem corresponding to the relaxation of the ordering problem can be solved exactly in polynomial time, then this heuristic will be a 2-approximation algorithm for the ordering problem. However, the number of fractions in this formulation is equal to the number of jobs. For a large number of jobs, solving the relaxation exactly will be computationally prohibitive. We present a more compact formulation of the ordering problem which uses Θ variables instead of the δ variables in the next section.

[1]The main difference between the two cases is that in the case of sum of completion times, for each job i we count the processing time of the jobs that are executed *before* job i, whereas in this case we count the overage costs of the jobs that are executed *after* job i. However, the inequality holds for the latter case as well.

5.4.2 An Exact Formulation of the Ordering Problem

The formulation given in this section is inspired from the completion time indexed formulation for the single machine scheduling problem with sum of completion time as the objective (Queyranne 1993). Instead of using the δ variables, we directly use the Θ variables in this formulation, which is given below.

$$\min \quad \sum_{i=1}^{n} \frac{\Delta_i \Theta_i}{1+\Theta_i} \tag{5.12a}$$

$$\text{s.t.} \quad \sum_{j \in S} o_j \Theta_j \geq \frac{1}{2}\left(\sum_{j \in S} o_j\right)^2 + \frac{1}{2}\sum_{j \in S} o_j^2 \quad \text{for all } S \subseteq N. \tag{5.12b}$$

Theorem 5.4.3 *The above formulation is an exact formulation of the ordering problem.*

Proof. The extreme point of the polyhedral set (5.12b) can be characterized as follows (Queyranne 1993). Let $\sigma : N \to N$ be a permutation of the jobs. That is, $\sigma(i)$ is the ith job in the execution sequence. Let $o_{\geq \sigma(i)} = \sum_{j \geq i} o_{\sigma(j)}$. Then $(o_{\geq 1}, \ldots, o_{\geq n})$ is an extreme point of this polyhedron, and all the extreme points of the polyhedron have this form. The objective function (5.12a) then corresponds to the cost of the schedule when the jobs are processed in the order given by the permutation σ. Thus each extreme point of the polyhedron (5.12b) corresponds to some ordering of the jobs.

Moreover, the objective function (5.12a) is a concave function in the variables Θ. Hence the optimal solution of the non-linear program (5.12a)-(5.12b) is attained at an extreme point of the polyhedron. This means that an optimal ordering of the jobs can be found by solving for an optimal extreme point of this formulation. If $(\Theta_1^*, \ldots, \Theta_n^*)$ is an optimal extreme point, then an optimal ordering of the jobs is given by a permutation σ which satisfies $\Theta^*_{\sigma(1)} \geq \ldots \geq \Theta^*_{\sigma(n)}$. $\square$

5.4.3 KKT Conditions for Local Optimality

For the above formulation, we look at the KKT conditions for obtaining a necessary condition for a sequence of jobs to be an optimal sequence. For all $S \subseteq N$, let $f(S)$ denote the function

$$f(S) = \frac{1}{2}\left(\sum_{j \in S} o_j\right)^2 + \frac{1}{2}\sum_{j \in S} o_j^2.$$

A necessary condition for a sequence of jobs to be an optimal sequence is captured in the following theorem.

Theorem 5.4.4 *For a given instance of the robust appointment scheduling problem, a necessary condition for the sequence* $1, \ldots, n$ *to be optimal is that the values*

$$\frac{\Delta_i / o_i}{(1 + o_{\geq i})^2}$$

must be in increasing order for $i = 1, \ldots, n$.

Proof. The Lagrangian formulation of the optimization problem (5.12a)-(5.12b) is given below.

$$L(\Theta, \mu) = \sum_{i=1}^{n} \frac{\Delta_i \Theta_i}{1 + \Theta_i} - \sum_{S \subseteq N} \mu_S \left(\sum_{j \in S} o_j \Theta_j - f(S) \right). \tag{5.13}$$

If $\Theta = (\Theta_1, \ldots, \Theta_n)$ is an optimal solution, then it must satisfy the following KKT conditions (Bertsekas, Nedić, and Ozdaglar 2003).

$$\text{Stationarity:} \quad \frac{\partial L}{\partial \Theta_i} = 0 \Rightarrow \frac{\Delta_i}{(1 + \Theta_i)^2} - o_i \sum_{S : i \in S} \mu_S = 0, i \in N. \tag{5.14a}$$

$$\text{Complementary slackness:} \quad \mu_S \left(\sum_{j \in S} o_j \Theta_j - f(S) \right) = 0, \quad \text{for all } S \subseteq N. \tag{5.14b}$$

$$\text{Primal feasibility:} \quad \sum_{j \in S} o_j \Theta_j \geq f(S), \quad \text{for all } S \subseteq N. \tag{5.14c}$$

$$\text{Dual feasibility:} \quad \mu_S \geq 0, \quad \text{for all } S \subseteq N. \tag{5.14d}$$

From Theorem 5.4.3, we know that there is a (local) optimal solution which is also an extreme point of the polyhedral set given by the constraints (5.12b). Therefore we restrict our analysis only to the extreme point solutions. The extreme point of the polyhedron corresponding to the job sequence $1, \ldots, n$ is $\Theta_i = o_{\geq i} = \sum_{j=i}^{n} o_j$ for $i = 1, \ldots, n$. Let $S_i = \{i, i+1, \ldots, n\}$ for $i = 1, \ldots, n$.

If the overage costs are all positive, then the constraints (5.14c) will be satisfied with equality for the sets $S = S_i$ for all $i = 1, \ldots, n$, and for all other sets S the inequality will be strict. Therefore, from the complementary slackness conditions (5.14b) we get $\mu_S = 0$ for all $S \neq S_i$ for any $i = 1, \dots, n$. The stationarity conditions (5.14a) then give

$$\sum_{j=1}^{i} \mu_{S_j} = \frac{\Delta_i / o_i}{(1 + o_{\geq i})^2}.$$

Therefore the value of the Lagrange multipliers we get are

$$\mu_S = \begin{cases} \dfrac{\Delta_1/o_1}{1 + o_{\geq 1}} & \text{for } S = S_1, \\ \dfrac{\Delta_i/o_i}{(1 + o_{\geq i})^2} - \dfrac{\Delta_{i-1}/o_{i-1}}{(1 + o_{\geq i-1})^2} & \text{for } S = S_i, i = 2, \dots, n, \\ 0 & \text{otherwise}. \end{cases} \tag{5.15}$$

The dual feasibility conditions (5.14d) and (5.15) together imply that a necessary condition for the sequence $1, \dots, n$ of jobs to be an optimal sequence is that the values

$$\frac{\Delta_i / o_i}{(1 + o_{\geq i})^2}$$

must be in increasing order. □

Note that the KKT conditions give only a necessary condition for local optimality. Thus the condition that we obtain above characterizes not only the globally optimal solutions, but also the local optima (with respect to optimizing the objective function (5.12a) over the polyhedral set (5.12b)). This means that a sequence of jobs satisfying the above condition may not necessarily be a globally optimal sequence. This analysis, however, does give us the intuition that scheduling the jobs in increasing order of Δ_i / o_i ratios may be a reasonable strategy, especially if the overage costs of the jobs are much smaller as compared to the underage cost. We analyze this particular heuristic in the next section.

5.4.4 An Approximation Algorithm for the Ordering Problem

In this section, we look at heuristic methods which give us near-optimal solution for the ordering problem in appointment scheduling. We first show that when the overage costs of the jobs are high as compared to the underage cost, then any ordering has a cost which is reasonably close to the cost of the optimal ordering.

Let σ^* denote an optimal ordering of the jobs for a given instance of the appointment scheduling problem. For an ordering σ of the jobs, $F(\sigma)$ denotes the cost of the corresponding optimal robust appointment schedule given by (5.6). By $o_{\min}$, we denote the minimum overage cost among the overage costs of all the jobs, and $o_{\geq 1}$ denotes the sum of the overage costs of all the jobs.

Lemma 5.4.5 *Let σ^* be an optimal ordering of the jobs for a given instance of the appointment scheduling problem. For any ordering σ,*

$$F(\sigma) \leq \frac{1+o_{\min}}{o_{\min}} \cdot \frac{o_{\geq 1}}{1+o_{\geq 1}} F(\sigma^*).$$

Proof. An upper bound on the cost of the ordering σ is given by

$$\begin{aligned} F(\sigma) &= \sum_{i=1}^{n} \frac{o_{\geq \sigma(i)} \Delta_i}{1+o_{\geq \sigma(i)}} \\ &\leq \sum_{i=1}^{n} \frac{o_{\geq 1} \Delta_i}{1+o_{\geq 1}} \\ &= \frac{o_{\geq 1}}{1+o_{\geq 1}} \sum_{i=1}^{n} \Delta_i. \end{aligned}$$

A lower bound on the cost of the optimal ordering σ^* is

$$\begin{aligned} F(\sigma^*) &= \sum_{i=1}^{n} \frac{o_{\geq \sigma^*(i)} \Delta_i}{1+o_{\geq \sigma(i)}} \\ &\geq \sum_{i=1}^{n} \frac{o_{\min} \Delta_i}{1+o_{\min}} \\ &= \frac{o_{\min}}{1+o_{\min}} \sum_{i=1}^{n} \Delta_i. \end{aligned}$$

Taking the ratio of the upper bound to the lower bound we get the statement of the lemma. □

The above lemma shows that when the overage costs are significantly higher compared to the underage costs, then any ordering has a cost that is quite close to the cost of the optimal ordering. For example, if $o_{\min} = 10$, then any ordering is a factor of 1.1 within that of the optimal ordering.

We propose a simple heuristic for the ordering problem, which is similar to the Smith's rule for finding an optimal schedule for scheduling jobs on a single machine to minimize weighted completion time.

Smith's ordering heuristic: Schedule the jobs in the non-decreasing ratio of Δ_i/o_i ratios.

Let the schedule obtained by Smith's rule be σ^S. The following lemma holds for this schedule (Smith 1956).

Lemma 5.4.6 *If σ^S is the ordering given by Smith's ordering heuristic, then for any ordering σ of the jobs,*

$$\sum_{i=1}^{n} o_{\geq \sigma^S(i)} \Delta_i \leq \sum_{i=1}^{n} o_{\geq \sigma(i)} \Delta_i.$$

Proof. Follows from (Smith 1956). □

The idea behind the heuristic is simple: If there is a job with a large uncertainty, then it is better to execute it in the end, so that if the job is overaged it does not delay the subsequent jobs. Similarly, if a job has a higher overage cost, then it is better to execute it in the start of the schedule otherwise the delay due to the lateness of the preceding jobs may lead to a higher overage cost of this job. The next lemma gives a bound on the performance of this ordering heuristic.

Lemma 5.4.7 *For Smith's ordering heuristic,*

$$F(\sigma^S) \leq \frac{1 + o_{\geq 1}}{1 + o_{\min}} F(\sigma^*).$$

Proof. The cost of the schedule σ^S is

$$\begin{aligned}
F(\sigma^S) &= \sum_{i=1}^{n} \frac{o_{\geq \sigma^S(i)} \Delta_i}{1 + o_{\geq \sigma^S(i)}} \\
&\leq \frac{1}{1 + o_{\min}} \sum_{i=1}^{n} o_{\geq \sigma^S(i)} \Delta_i \\
&\leq \frac{1}{1 + o_{\min}} \sum_{i=1}^{n} o_{\geq \sigma^*(i)} \Delta_i \\
&\leq \frac{1 + o_{\geq 1}}{1 + o_{\min}} \sum_{i=1}^{n} \frac{o_{\geq \sigma^*(i)} \Delta_i}{1 + o_{\geq \sigma^*(i)}} \\
&= \frac{1 + o_{\geq 1}}{1 + o_{\min}} F(\sigma^*),
\end{aligned}$$

where the second inequality follows from Lemma 5.4.6. □

The two lemmata together imply the following theorem.

Theorem 5.4.8 *The Smith's ordering heuristic gives a* $\min\left(\frac{1+o_{\geq 1}}{1+o_{\min}}, \frac{o_{\geq 1}}{1+o_{\geq 1}} \cdot \frac{1+o_{\min}}{o_{\min}}\right)$*-approximate solution to the ordering problem.*

Proof. Follows from Lemma 5.4.5 and Lemma 5.4.7. □

Thus, for the case where $o_{\geq 1}$ is not too large compared to $o_{\min}$, Smith's ordering heuristic gives a near-optimal ordering of the jobs.

5.5 Conclusion and Future Directions

We have presented a robust optimization framework for the appointment scheduling problem arising in service operations. The main contribution of this chapter is in demonstrating that there is a neat closed form for the optimal schedule in the robust model. We also give insights into the structure of the optimal solution. Furthermore, we propose simple heuristics to get a near-optimal ordering of the jobs, something which has eluded the stochastic models so far.

We present two problems below related to appointment scheduling that are worth pursuing:

1. **Hardness of the appointment scheduling problem**: It is well known that the appointment scheduling problem is quite intractable for the stochastic model. Surprisingly, no one has given a proof of NP-hardness of this problem for for the most general case for any of the models mentioned in Section 5.1. We believe that at the very least, the problem of finding an optimal order of the execution of the jobs is NP-hard, both for the stochastic model as well as for the robust model.

2. **Scheduling multiple facilities**: So far, in our model we have assumed that the jobs are to be scheduled on a single facility. In general, there might be more than one facility where the jobs can be processed, with possibly different underage/overage cost for different facilities. In the most general form, the problem will have three components: assigning which jobs are to processed on which machines, ordering the jobs assigned to each machine in an optimal sequence, and then computing the optimal duration that should be assigned to each job.

Appendix

A Local Search Algorithm for the Discrete Stochastic Model

The discrete stochastic model of Begen and Queyranne (2011) makes the assumptions that the processing duration P_i of each job i is a discrete random variable with integer support. The probability distributions of the random variables are assumed to be mutually independent. The probability mass functions of the distributions are known, and is a part of the input to the problem. Under this assumption, they prove the following results:

1. There is an integer appointment schedule (i.e. a schedule in which the assigned starting time of each job is an integer) that is optimal.
2. The cost function $F(A) = \mathbb{E}_P[F(A, P)]$ is an L-convex function over the lattice of integer points. (For more details on L-convex functions, see Murota (2003).)
3. An integer schedule A minimizing the cost function can be computed in polynomial time by using a polynomial time algorithm for minimizing a submodular set function.

The algorithm used for minimizing the cost function over an integer lattice is the steepest descent algorithm (Murota 2003), whose outline is given below. In the rest of this section, $\mathbb{I}_X$ denotes the indicator vector of $X \subseteq \{1, \ldots, n\}$.

```
1. Let $A$ be any integer appointment schedule for the given instance.
2. Find $\epsilon \in \{-1, 1\}$ and $X \subseteq \{1, \ldots, n\}$ that minimizes $F(A + \epsilon \mathbb{I}_X)$.
3. If $F(A + \epsilon \mathbb{I}_X) < F(A)$
4.        $A \leftarrow A + \epsilon \mathbb{I}_X$,
5.        Go to step 2.
6. Return $A$ as the optimal solution.
```

The key step in the above algorithm is the local search in step 2. The function $g(X) = F(A+\mathbb{I}_X) - F(A)$ is a submodular function in X, and hence the function can be minimized in polynomial time by using an efficient algorithm for submodular function minimization (see e.g. Iwata (2008), Orlin (2009)).

In our implementation, instead of computing the solution that gives the best improvement for the objective function in each step, we compute *any* solution that gives an improvement over the current solution. This modified algorithm is guaranteed to return an optimal solution, however it is not guaranteed to run in polynomial time. In our computational study, the modified local search algorithm returned an optimal solution in a reasonable time for up to 14 jobs in an instance.

Bibliography

Aissi, H., C. Bazgan, and D. Vanderpooten (2007). Approximation of min-max and min-max regret versions of some combinatorial optimization problems. *European Journal of Operational Research 179*, 281–290.

Al-Khayyal, F. A. and J. E. Falk (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research 8*, 273–286.

Asadpour, A. and A. Saberi (2007). An approximation algorithm for max-min fair allocation of indivisible goods. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing*, San Diego, CA, pp. 114–121.

Atamtürk, A. and V. Narayanan (2008). Polymatroids and mean0risk minimization in discrete optimization. *Operations Research Letters 36*, 618–622.

Azar, Y., L. Epstein, Y. Richter, and G. J. Woeginger (2004). All-norm approximation algorithms. *Journal of Algorithms 52*, 120–133.

Barahona, F. and W. Pulleyblank (1987). Exact arborescences, matchings and cycles. *Discrete Applied Mathematics 16*, 91–99.

Begen, M. A., R. Levi, and M. Queyranne (2008). A sampling-based approach to appointment scheduling. Working Paper, Sauder School of Business, University of British Columbia.

Begen, M. A. and M. Queyranne (2011). Appointment scheduling with discrete random durations. *Mathematics of Operations Research 36*, 240–257.

Ben-Tal, A. and A. Nemirovski (2002). Robust optimization - methodology and applications. *Mathematical Programming 92*, 453–480.

Bendavid, I. and B. Golany (2009). Setting gates for activities in the stochastic project scheduling problem through the cross entropy methodology. *Annals of Operations Research 172*, 259–276.

Benson, H. P. and G. M. Boger (1997). Multiplicative programming problems: Analysis and efficient point search heuristic. *Journal of Optimization Theory and Applications 94*, 487–510.

Bertsekas, D., A. Nedić, and A. Ozdaglar (2003). *Convex Analysis and Optimization*. Belmont, MA: Athena Scientific.

Bertsimas, D. and M. Sim (2004). The price of robustness. *Operations Research 52*, 35–53.

Bertsimas, D. and A. Thiele (2006). A robust optimization approach to inventory theory. *Operations Research 54*, 150–168.

Billionnet, A. (2002). Approximation algorithms for fractional knapsack problems. *Operations Research Letters 30*, 336–342.

Cayirli, T. and E. Veral (2003). Outpatient scheduling in healthcare: a review of literature. *Production and Operations Management 12*, 519–549.

Chekuri, C. and S. Khanna (2004). On multidimensional packing problems. *SIAM Journal on Computing 33*, 837–851.

Correa, J. R., C. G. Fernandes, and Y. Wakabayashi (2010). Approximating a class of combinatorial problems with rational objective function. *Mathematical Programming B 124*, 255–269.

Denton, B. and D. Gupta (2003). A sequential bounding approach for optimal appointment scheduling. *IIE Transactions 35*, 1003–1016.

Diakonikolas, I. and M. Yannakakis (2008). Succinct approximate convex pareto curves. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, pp. 74–83.

Eisenberg, E. (1961). Aggregation of utility functions. *Management Science*, 337–350.

Elhafsi, M. (2002). Optimal leadtimes planning in serial production systems with earliness and tardiness costs. *IIE Transactions 34*, 233–243.

Falk, J. E. and S. W. Palocsay (1992). Optimizing the sum of linear fractional functions. In C. A. Floudas and P. M. Pardalos (Eds.), *Recent Advances in Global Optimization*, pp. 228–258. Dordrecht: Kluwer Academic Publishers.

Feige, U., V. S. Mirrokni, and J. Vondrák (2007). Maximizing non-monotone submodular functions. In *48th Annual IEEE Symposium on Foundations of Computer Science*, Providence, RI, pp. 461–471.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York, NY: W. H. Freeman.

Garofalakis, M. N. and Y. E. Ioannidis (1996). Multi-dimensional resource scheduling for parallel queries. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, Montreal, Quebec, Canada, pp. 365–376.

Goyal, V., L. Genc-Kaya, and R. Ravi (2011). An FPTAS for minimizing the product of two non-negative linear cost functions. *Mathematical Programming 126*, 401–405.

Goyal, V. and R. Ravi (2009). An FPTAS for minimizing a class of quasi-concave functions over a convex domain. Technical report, Tepper School of Business, Carnegie Mellon University.

Green, L. V., S. Savin, and B. Wang (2006). Managing patient service in a diagnostic medical facility. *Operations Research 54*, 11–25.

Grötschel, M., L. Lovász, and A. Schrijver (1988). *Geometric Algorithms and Combinatorial Optimization*. Berlin: Springer.

Gupta, D. (2007). Surgical suites' operations management. *Productions and Operations Management 16*, 689–700.

Halman, N., D. Klabjan, M. Mostagir, J. B. Orlin, and D. Simchi-Levi (2009). A fully polynomial-time approximation scheme for single-item stochastic inventory control with discrete demand. *Mathematics of Operations Research 34*, 674–685.

Hansen, P. (1979). Bicriterion path problems. *Proceedings of the 3rd Conference on Multiple Criteria Decision Making Theory and Application*, 109–127.

Hashizume, S., M. Fukushima, N. Katoh, and T. Ibaraki (1987). Approximation algorithms for combinatorial fractional programming problems. *Mathematical Programming 37*, 255–267.

Håstad, J. (2001). Some optimal inapproximability results. *Journal of the ACM 48*, 798–859.

Hochbaum, D. S. and D. B. Shmoys (1987). Using dual approximation algorithms for scheduling problems: Theoretical and practical results. *Journal of the ACM 34*, 144–162.

Horowitz, E. and S. Sahni (1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM 23*, 317–327.

Horst, R. and P. M. Pardalos (Eds.) (1995). *Handbook of Global Optimization*, Volume 1. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Hu, J. and S. Mehrotra (2010). Robust and stochastically weighted multi-objective optimization models and reformulations. Technical report, Department of Industrial and Management Sciences, Northwestern University.

Ibarra, O. H. and C. E. Kim (1975). Fast approximation algorithms for the knapsack and sum of subset problems. *Journal of the ACM 22*, 463–468.

Iwata, S. (2008). Submodular function minimization. *Mathematical Programming 112*, 45–64.

Kandoorp, G. C. and G. Koole (2007). Optimal outpatient appointment scheduling. *Health Care Management Science 10*, 217–229.

Kelner, J. A. and E. Nikolova (2007). On the hardness and smoothed complexity of quasi-concave minimization. In *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, Providence, RI, pp. 472–482.

Kern, W. and G. J. Woeginger (2007). Quadratic programming and combinatorial minimum weight product problems. *Mathematical Programming 110*, 641–649.

Konno, H. (1976). A cutting plane algorithm for solving bilinear programs. *Mathematical Programming 11*, 14–27.

Konno, H., C. Gao, and I. Saitoh (1998). Cutting plane/tabu search algorithms for low rank concave quadratic programming problems. *Journal of Global Optimization 13*, 225–240.

Konno, H. and T. Kuno (1992). Linear multiplicative programming. *Mathematical Programming 56*, 51–64.

Konno, H., T. Thach, and H. Tuy (1996). *Optimization on Low Rank Nonconvex Structures*. Dordrecht, The Netherlands: Kluwer Academic Publishers.

Kouvelis, P. and G. Yu (1997). *Robust discrete optimization and its applications*. Boston: Kluwer Academic Publishers.

Kuno, T. (1999). Polynomial algorithms for a class of minimum rank-two cost path problems. *Journal of Global Optimization 15*, 405–417.

Lenstra, J. K., D. B. Shmoys, and É. Tardos (1990). Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming 46*, 259–271.

Levi, R., M. Pál, R. Roundy, and D. B. Shmoys (2007). Approximation algorithms for stochastic inventory control models. *Mathematics of Operations Research 32*, 284–302.

Levi, R., R. Roundy, D. B. Shmoys, and V. A. Truong (2008). Approximation algorithms for capacitated stochastic inventory control models. *Operations Research 56*, 1184–1199.

Lovász, L. (1983). Submodular functions and convexity. In A. Bachem, M. Grötschel, and B. Korte (Eds.), *Mathematical Programming - The State of the Art*, pp. 235–257. Springer-Verlag.

Matsui, T. (1996). NP-hardness of linear multiplicative programming and related problems. *Journal of Global Optimization 9*, 113–119.

Megiddo, N. (1979). Combinatorial optimization with rational objective functions. *Mathematics of Operations Research 4*, 414–424.

Mittal, S. and A. S. Schulz (2008). A general framework for designing approximation schemes for combinatorial optimization problems with many objectives combined into one. In *Proceedings of the APPROX-RANDOM*, Volume 5171 of *Lecture Notes in Computer Science*, Cambridge, MA, pp. 179–192.

Mittal, S. and A. S. Schulz (2010). An FPTAS for optimizing a class of low-rank functions over a polytope. Technical report, Operations Research Center, Massachusetts Institute of Technology.

Mittal, S. and S. Stiller (2011). Robust appointment scheduling. In *Proceedings of the MSOM Annual Conference*, Ann Arbor, MI.

Murota, K. (2003). *Discrete Convex Analysis*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Nesterov, Y. and A. Nemirovskii (1961). *Interior Point Polynomial Methods in Convex Programming*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Nikolova, E. (2010). Approximation algorithms for reliable stochastic combinatorial optimization. In *Proceedings of the APPROX-RANDOM*, Volume 6302 of *Lecture Notes in Computer Science*, Barcelona, Spain, pp. 338–351.

Orlin, J. B. (2009). A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming 118*, 237–251.

Papadimitriou, C. H. and M. Yannakakis (2000). On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science*, Redondo Beach, CA, pp. 86–92.

Pardalos, P. M. and S. A. Vavasis (1991). Quadratic programming with one negative eigenvalue is NP-hard. *Journal of Global Optimization 1*, 15–22.

Porembski, M. (2004). Cutting planes for low-rank like concave minimization problems. *Operations Research 52*, 942–953.

Queyranne, M. (1993). Structure of a simple scheduling polyhderon. *Mathematical Programming 58*, 263–285.

Robinson, L. W. and R. R. Chen (2003). Scheduling doctor's appointments: Optimal and empirically-based heuristic policies. *IIE Transactions 35*, 295–307.

Rusmevichientong, P., Z.-J. M. Shen, and D. B. Shmoys (2009). A PTAS for capacitated sum-of-ratios optimization. *Operations Research Letters 37*, 230–238.

Rusmevichientong, P., D. B. Shmoys, and H. Topaloglu (2010). Assortment optimization with mixtures of logits. Technical report, School of Operations Research and Information Engineering, Cornell University.

Sabria, F. and C. F. Daganzo (1989). Approximate expressions for queueing systems with scheduled arrivals and established service orders. *Transportation Science 23*, 159–165.

Safer, H. M. and J. B. Orlin (1995a). Fast approximation schemes for multi-criteria combinatorial optimization. Technical report, Operations Research Center, Massachusetts Institute of Technology.

Safer, H. M. and J. B. Orlin (1995b). Fast approximation schemes for multi-criteria flow, knapsack, and scheduling problems. Technical report, Operations Research Center, Massachusetts Institute of Technology.

Safer, H. M., J. B. Orlin, and M. Dror (2004). Fully polynomial approximation in multi-criteria combinatorial. Technical report, Operations Research Center, Massachusetts Institute of Technology.

Sahni, S. (1976). Algorithms for scheduling independent tasks. *Journal of the ACM 23*, 116–127.

Schaible, S. (1977). A note on the sum of a linear and linear-fractional function. *Naval Research Logistics 24*, 691–693.

Schaible, S. and J. Shi (2003). Fractional programming: The sum-of-ratios case. *Optimization Methods and Software 18*, 219–229.

Schulz, A. S. (1996). Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In *Proceedings of the 5th International Integer Programming and Combinatorial Optimizaton Conference*, Vancouver, British Columbia, Canada, pp. 301–315.

Sherali, H. D. and A. Alameddine (1992). A new reformulation-linearization technique for bilinear programming problems. *Journal of Global Optimization 2*, 379–410.

Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly 3*, 59–66.

Vavasis, S. A. (1992). Approximation algorithm for indefinite quadratic programming. *Mathematical Programming 57*, 279–311.

Wang, P. P. (1993). Static and dynamic scheduling of customer arrivals to a single-server system. *Naval Research Logistics 40*, 345–360.

Wang, P. P. (1999). Sequencing and scheduling n customers for a stochastic server. *European Journal of Operational Research 119*, 729–738.

Williamson, D. P. and D. B. Shmoys (2011). *The Design of Approximation Algorithms*. Cambridge University Press.

Woeginger, G. J. (2000). When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? *INFORMS Journal on Computing 12*, 57–74.