A General Framework for Designing Approximation Schemes for Combinatorial Optimization Problems with Many Objectives Combined into One

A General Framework for Designing Approximation Schemes for Combinatorial Optimization Problems with Many Objectives Combined Into One

Shashi Mittal* Andreas S. Schulz**

Abstract

In this paper, 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 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. The main idea behind our approximation schemes is the construction of an approximate Pareto-optimal front of the functions which constitute the given objective. Using this idea, we give the first fully polynomial time approximation schemes for the max-min resource allocation problem with a fixed number of agents and for combinatorial optimization problems in which the objective function is the sum of a fixed number of ratios of linear functions, or the product of a fixed number of linear functions.

*Operations Research Scientist, Amazon.com, 331 Boren Avenue N, Seattle, WA 98107. This work was done when the author was a graduate student in the Operations Research Center at Massachusetts Institute of Technology. Email: mshashi@alum.mit.edu.

**Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, E62-569, Cambridge MA 02139, USA. Email: schulz@mit.edu.

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 paper 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 a fully polynomial time approximation scheme (FPTAS) for such problems. An FPTAS for a minimization (resp. maximization) problem is a family of algorithms parametrized by ϵ such that for all $\epsilon > 0$, there is an algorithm A_ϵ which for any instance of the problem returns a solution whose objective function value is no more then $(1 + \epsilon)$ (resp. no less than $(1 - \epsilon)$) times that of the optimal solution, and whose running time is polynomial in the input size of the problem as well as in $1/\epsilon$. For an NP-hard problem, the existence of an FPTAS is typically the best possible approximation result one can obtain. 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.

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:

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

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 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 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 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 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[2] for the problem when the number of resources on each machine is fixed. Moreover,

[2]A Polynomial Time Approximation Scheme (PTAS) for minimization problems is a family of algorithms parametrized by ϵ such that for all

they only consider the case where each job has the same requirement for a particular resource on all the machines. In Section 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 5.3 of this paper, 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:

$$\text{minimize } g(x) = \max_{w \in W} w^T f(x), \qquad x \in X \subseteq \{0,1\}^d. \tag{2}$$

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 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.

$\epsilon > 0$, there is a $(1+\epsilon)$ approximation algorithm, but the running time of the algorithm is not necessarily polynomial in $1/\epsilon$.

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.

Several methods also exist for designing approximation schemes for combinatorial optimization problems with multiple objectives. Safer and Orlin (1995a) give necessary and sufficient conditions for the existence of fully polynomial time approximation schemes in multi-criteria combinatorial optimization, and then use it for designing fast approximation schemes for multi-criteria flow, knapsack and scheduling problems (Safer and Orlin 1995b; Safer, Orlin, and Dror 2004). Papadimitriou and Yannakakis (2000) propose an efficient procedure to construct an approximate Pareto-optimal frontier for discrete multi-objective optimization problems, and we use their procedure in designing approximation schemes for the above problems, as explained in the next section.

1.3 Overview of Our Framework

We present a general framework which we use to design FPTASes for the problems given in Section 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 2, we introduce the concepts Pareto-optimal front and approximate Pareto-optimal front and also revisit previous results on constructing an approximate Pareto-optimal front for multi-objective optimization problems, which we use in the later sections for designing our approximation schemes. 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 paper in Section 3. Subsequently, we derive FPTASes for the problems mentioned in Section 1.1 as corollaries to the main theorem.

2 Preliminaries on Multi-Objective Optimization

An instance π of a multi-objective optimization problem Π is given by a set of m functions $f_1, \ldots, f_m$. Each $f_i : X \to \mathbb{R}_+$ is defined over the same set of feasible solutions, X. Let $|\pi|$ denote the binary-encoding size of the instance π. Assume that each f_i takes values in the range $[2^{-p(|\pi|)}, 2^{p(|\pi|)}]$ for some polynomial p. We first define the Pareto-optimal frontier for multi-objective minimization problems.

Definition 2.1 *Let π be an instance of a multi-objective minimization problem. A* Pareto-optimal frontier*, denoted by $P(\pi)$, is a set of solutions $x \in X$, such that 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 undominated solutions. In many cases, it may not be tractable to compute $P(\pi)$ (e.g., determining whether a point belongs to the Pareto-optimal frontier for the two-objective shortest path problem is NP-hard), or the number of undominated solutions can be exponential in $|\pi|$ (e.g., for the two-objective shortest path problem (Hansen 1979)). One way of getting around this problem is to look at an approximate Pareto-optimal frontier, which is defined below.

Definition 2.2 *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.*

In the rest of the paper, whenever we refer to an (approximate) Pareto-optimal frontier, we mutually refer to both its set of solutions and their vectors of objective function values.

Even though $P(\pi)$ may contain exponentially many points, there is always an approximate Pareto-optimal frontier having a polynomial number of points. The following theorem gives one possible way to construct such an approximate Pareto-optimal frontier in polynomial time.

Theorem 2.3 (Papadimitriou and Yannakakis (2000)) *Let m 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 an m-vector of values $(v_1, \ldots, v_m)$, either

(i) *return a solution $x \in X$ such that $f_i(x) \leq v_i$ for all $i = 1, \ldots, m$, or*

(ii) *assert that there is no $x \in X$ such that $f_i(x) \leq (1 - \epsilon')v_i$ for all $i = 1, \ldots, m$.*

We sketch the proof because our approximation schemes are based on it.

Proof. Suppose we can solve the gap problem in polynomial time. An approximate Pareto-optimal frontier can then be constructed as follows. Consider the box in $\mathbb{R}^m$ of possible function values given by $\{(v_1, \ldots, v_m) : 2^{-p(|\pi|)} \leq v_i \leq 2^{p(|\pi|)} \text{ for all } i\}$. We divide this box into smaller boxes, 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 boxes, we call 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 $P_\epsilon(\pi)$. Since there are $\mathrm{O}((p(|\pi|)/\epsilon)^m)$ many smaller boxes, this can be done in polynomial time.

Conversely, suppose we can construct $P_\epsilon(\pi)$ in polynomial time. To solve the gap problem for a given m-vector $(v_1, \ldots, v_m)$, if there is a solution point $(f_1(x), \ldots, f_m(x))$ in $P_\epsilon(\pi)$ such that $f_i(x) \leq v_i$ for all i, then we return x. Otherwise we assert that there is no $x \in X$ such that $f_i(x) \leq (1 - \epsilon')v_i$ for all $i = 1, \ldots, m$. □

Thus, 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 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 A.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 2.4 *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 $\mathrm{O}((d^2/\epsilon)^{m+1})$. 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 2.3. □

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. We then show how this general framework can be adapted to obtain FPTASes for the problems given in Section 1.1.

Let $f_1, \ldots, f_m$, for m fixed, be functions which satisfy the conditions given in the beginning of Section 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:

$$\text{minimize } g(x) = h(f(x)), \qquad x \in X. \tag{3}$$

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 3.1 *There is at least one optimal solution x^* to* (3) *such that $x^* \in P(\pi)$.*

Proof. Let $\hat{x}$ be an optimal solution of (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 3.2 *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)*; that is, $g(\hat{x})$ is at most $(1+\epsilon)$ times the value of an optimal solution to* (3)*.*

Proof. Let x^* be an optimal solution of (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 2.4, we get the main theorem of this paper regarding the existence of an FPTAS for solving (3).

Theorem 3.3 *Suppose the exact problem corresponding to the functions given in* (3) *can be solved in pseudo-polynomial time. Then there is an FPTAS for solving the general optimization problem* (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 3.4 *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. □

4 FPTAS for Scheduling and Resource Allocation Problems

Using the framework presented in Section 3, we give FPTASes for the max-min resource allocation problem, the $Rm||C_{\max}$ problem and the vector scheduling problem.

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).

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

$$\sum_{i=1}^{m} x_{ik} = 1 \qquad \text{for } k = 1, \ldots n, \tag{4a}$$

$$x_{ik} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, m, \quad k = 1, \ldots, n. \tag{4b}$$

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 (4a) and (4b)? The following lemma establishes that the exact problem can be solve in pseudo-polynomial time.

Lemma 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 $1 \leq i \leq n$ 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 $0 \leq i \leq n - 1$. 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 $j + c \leq C$. 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 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.

Corollary 4.2 *There is an FPTAS for the max-min resource allocation problem with a fixed number 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.*

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 $i = 1, \ldots m$ and $j = 1, \ldots, d$. 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 (4a)-(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.

Corollary 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.*

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.

5.1 Formulation of the FPTAS

Consider the following optimization problem.

$$\text{minimize } g(x) = f_1(x) \cdot f_2(x), \qquad x \in X, \tag{5}$$

where $f_i : X \to \mathbb{Z}_+$ are linear functions and $X \subseteq \{0,1\}^d$. In our general formulation given by (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.

Corollary 5.1 *There is an FPTAS for the problem given by* (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 $\sum_{i=1}^{d} c_i x_i = K$?*

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 2.3 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.

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 (5).

$$\text{minimize } g(x) = f_1(x) \cdot f_2(x) \cdot \ldots \cdot f_m(x), \qquad x \in X, \tag{6}$$

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), with $c = m$. Thus our technique yields an FPTAS for the problem given by (6) as well. We have therefore established the following corollary to Theorem 3.3.

Corollary 5.2 *There is an FPTAS for the problem given by* (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 $\sum_{i=1}^{d} c_i x_i = K$?*

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):

$$\text{minimize } g(x) = \max_{w \in W} w^T f(x), \qquad x \in X \subseteq \{0,1\}^d. \tag{7}$$

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.

$$\text{minimize } h(x) = \max_{c \in \{c_1, \ldots, c_m\}} c^T x, \qquad x \in X \subseteq \{0,1\}^d. \tag{8}$$

The connection between the problems given by (7) and (8) is established in the following lemma.

Lemma 6.1 *The problem given by* (8) *is equivalent to the problem given by* (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 (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 (7), $\text{conv}(\{c_1, \ldots, c_m\})$ 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 (7) with the simplex weight set and the optimization problem given by (8). □

Using this observation, we establish the NP-hardness of the optimization problem given by (7).

Lemma 6.2 *The optimization problem given by* (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):

$$\text{minimize } h(x) = \max_{c \in \{c_1, c_2\}} c^T x, \qquad x \in X \subseteq \{0,1\}^d. \tag{9}$$

Problem (9) is equivalent to the form given by (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 (7) is also NP-hard in general. □

Next, we establish that when m, the number of objectives, is fixed, the optimization problem given by (7) admits an FPTAS.

Lemma 6.3 *There is an optimal solution to* (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 (7). Suppose x^* is not on the Pareto-optimal front. By definition, there exists $\hat{x} \in P(\pi)$ such that $f_i(\hat{x}) \leq f_i(x^*)$ for $i = 1, \ldots, m$. Let $\hat{w} \in W$ be the weight vector which maximizes $w^T f(\hat{x})$. Then,

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

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

Lemma 6.4 *There is a solution $\hat{x}$ on $P_\epsilon(\pi)$ that is a $(1+\epsilon)$-approximate solution of the optimization problem* (7)*.*

Proof. Let x^* be the optimal solution to the problem given by (7). By definition of $P_\epsilon(\pi)$, there exists $\hat{x} \in P_\epsilon(\pi)$ such that $f_i(\hat{x}) \leq (1+\epsilon) f_i(x^*)$ for $i = 1, \ldots, m$. Let $\hat{w} \in W$ be the weight which maximizes $w^T f(\hat{x})$. Therefore,

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

Therefore $\hat{x}$ is a $(1+\epsilon)$ approximate solution to the problem given by (7). □

Together with the above two lemmata, we get the following corollary to Theorem 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 6.5 *There is an FPTAS for the problem given by* (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 $\sum_{i=1}^d c_i x_i = K$?*

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:

$$\begin{aligned} \text{minimize } g(x) &= \frac{f_1(x)}{f_2(x)} = \frac{a_0 + a_1 x_1 + \ldots + a_d x_d}{b_0 + b_1 x_1 + \ldots + b_d x_d} \\ \text{s.t.} \quad & x \in X \subseteq \{0,1\}^d. \end{aligned} \tag{10}$$

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 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 7.1 *Consider the problem given by* (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 7.2 *For the problem given by* (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 2.3, so we omit it.

Theorem 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 3.1 holds in this case, with the modified definition of the Pareto-optimal front. The analog of Lemma 3.2 is given below.

Lemma 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* (10)*.*

Proof. Let x^* be an optimal solution of (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 $P_\epsilon(\pi)$, $g(\hat{x}) \leq g(x') \leq (1+\epsilon)^2 g(x^*)$. □

The following theorem is an analog of Theorem 3.3 for this case.

Theorem 7.5 *There is an FPTAS for the problem given by* (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 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 $r = \lceil d/\epsilon' \rceil$. 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 $a_j' = \lceil a_j r / v_1 \rceil$. In the new objective function, the maximum value of a coefficient is now r. For $x \in X$, the following two statements hold by Lemma A.1.

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

For f_2, we do the following. Let $f_2'(x) = \sum_{j \in V} b_j' x_j$, where $b_j' = \min(r, \lfloor b_j r / v_2 \rfloor)$. So in f_2', all the coefficients are no more than r. The following two statements hold by Lemma A.2.

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

Therefore, to solve the gap problem, it suffices to find an $x \in X$ such that $f_1'(x) \leq r$ and $f_2'(x) \geq r$, or assert that no such x exists. There are $r + 1$ ways in which $f_1'(x) \leq r$ can be satisfied, and there are at most rd ways in which $f_2'(x) \geq r$ can be satisfied. Hence there are $O(r^2 d)$ 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 $M = dr + 1$ 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. □

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.

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 2.3.

Corollary 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

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

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 2.3). 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 7.5.

Corollary 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.*

8 Conclusion

The main contribution of this paper 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.

Acknowledgments

The authors thank Vineet Goyal and Ola Svensson for interesting discussions, and Claudio Telha for his feedback on a preliminary version of this paper.

References

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.

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.

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.

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

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.

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.

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.

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.

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.

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

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.

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

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.

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.

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 optimization. Technical report, Operations Research Center, Massachusetts Institute of Technology.

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

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.

A Appendix

Lemma A.1 *Suppose* $f(x) = \sum_{j=1}^{d} a_j x_j$, $0 \le a_j \le v$, $x_j \in \{0,1\}$ *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* $f'(x) \le r$, *then* $f(x) \le v$.
2. *If* $f(x) \le v(1-\epsilon)$, *then* $f'(x) \le r$.

Proof.

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

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

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

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

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

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

□

Lemma A.2 *Suppose* $f(x) = \sum_{j=1}^{d} b_j x_j$, $0 \le b_j \le v$, $x_j \in \{0,1\}$ *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* $f'(x) \ge r$, *then* $f(x) \ge v$.
2. *If* $f(x) \ge (1+\epsilon)v$, *then* $f'(x) \ge r$.

Proof.

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

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

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

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

□