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

Shashi Mittal and Andreas S. Schulz

Operations Research Center and Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139
{mshashi,schulz}@mit.edu

Abstract. In this paper, we propose a general framework for designing fully polynomial time approximation schemes for combinatorial optimization problems, in which more than one objective function are combined into one using any norm. The main idea is to exploit the approximate Pareto-optimal frontier for multi-criteria optimization problems. Using this approach, we obtain an FPTAS for a novel resource allocation problem, for the problem of scheduling jobs on unrelated parallel machines, and for the Santa Claus problem, when the number of agents/machines is fixed, for any norm, including the l_∞-norm. Moreover, either FPTAS can be implemented in a manner so that the space requirements are polynomial in all input parameters. We also give approximation algorithms and hardness results for the resource allocation problem when the number of agents is not fixed.

1 Introduction

Consider the following resource allocation problem. There are m agents, and n resources, which are to be distributed among the agents. Each resource is assumed to be unsplittable; that is, a resource can be allocated to only one of the agents. However, agents may need to access resources assigned to other agents as well. The cost incurred by agent i, if it needs to access resource k from agent j, is c_{ij}^k. We assume that the c_{ij}^k are non-negative integers, and that $c_{ii}^k = 0$. The goal is to have a fair allocation of the resources among the agents; in other words, the maximum cost of an agent is to be minimized.

A practical setting where such a resource allocation problem can arise is page sharing in a distributed shared memory multiprocessor architecture [1]. In this architecture, the shared memory is distributed among different processors (also referred to as nodes), and each node contains a part of the shared memory locally. Typically, accessing the local memory is faster than accessing the remote memory. Every physical page in this architecture is allocated to a fixed node, which is referred to as the home node of the page. Also, there cannot be more

A. Goel et al. (Eds.): APPROX and RANDOM 2008, LNCS 5171, pp. 179–192, 2008.
© Springer-Verlag Berlin Heidelberg 2008

than one copy of a page in the system. Suppose each node knows in advance the number of accesses it will need to make to a page. The total delay, or latency, faced by a node is the sum of latencies over all the pages it needs to access. Suppose the latency between node i and j is t_{ij}, and the number of times node i needs to access page k is a_{ik}. If page k is stored in node j, then the cost of accessing page k for node i will be $c_{ij}^k = t_{ij}a_{ik}$. The performance of the system is governed by the node having maximum total latency. Thus, the objective is to allocate pages among the nodes in an offline fashion so that the maximum total latency over all the nodes is minimized.

We present an FPTAS for this resource allocation problem when the number of agents is fixed. There are many standard techniques for obtaining approximation schemes for combinatorial optimization problems. They include rounding of the input parameters (e.g. [2,3,4]), and shrinking the state space of dynamic programs [5]. We propose a novel framework for designing approximation schemes. The idea behind the new procedure is to treat the cost of each agent as a separate objective function, and to find an approximate Pareto-optimal frontier corresponding to this multi-objective optimization problem. Safer et al. [6][1] give necessary and sufficient conditions for the existence of fully polynomial time approximation schemes in multi-criteria combinatorial optimization. Papadimitriou and Yannakakis [9] propose an efficient procedure to construct an approximate Pareto-optimal frontier for discrete multi-objective optimization problems, and we use their procedure in constructing the approximation scheme for the resource allocation problem.

A closely related problem is the Santa Claus problem [10,11,12]. In this problem, each agent has a utility corresponding to each resource allocated to it, and the objective is to allocate the resources among the agents so that the minimum utility over all the agents is maximized. Our problem is different from the Santa Claus problem in that there is a cost associated with accessing each resource an agent does not get, instead of having a utility for each resource it gets. Using the above framework, we obtain the first FPTAS for the Santa Claus problem with a fixed number of agents.

Another closely related problem is scheduling jobs on unrelated parallel machines to minimize the makespan, also referred to in the literature as the $Rm||C_{\max}$ problem. There are m machines and n jobs, and each job is to be scheduled on one of the machines. The processing time of job k on machine i is p_{ik}. The objective is to minimize the makespan, that is the time at which the last job finishes its execution. Our procedure yields the first FPTAS for this problem that has space requirements that are polynomial in all the input parameters.

The resource allocation problem is NP-hard even when there are only two agents, and strongly NP-hard when the number of agents is variable (see the proof of NP-hardness in the Appendix). It remains strongly NP-hard for the special case of *uniform costs*, in which for each agent i and each resource k, $c_{ij}^k = c_i^k$ for all agents $j \neq i$. In this paper, we give a 2-approximation algorithm for

[1] This paper is a combined version of two earlier working papers by Safer and Orlin [7,8].

the uniform cost case. The algorithm makes use of the well-known technique of parametric linear programming and rounding, which has been successfully used in obtaining approximation algorithm for scheduling problems in the past [3]. Our rounding procedure, however, differs from the one given in [3]; it is more similar to the one used by Bezàkovà and Dani [12] for the Santa Claus problem.

Our results: The results in this paper can be summarized as follows.
1. *Approximation schemes.* We present a general framework for designing approximation schemes for problems with multiple objective functions combined into one using norms or other functions. We illustrate the versatility of this scheme by applying it to the resource allocation problem, the $Rm|\,|C_{\max}$ problem, and the Santa Claus problem. An interesting byproduct is that, by a careful implementation of the FPTAS, the space requirements can be made polynomial in all the input parameters. Previously, all FPTASes for the $Rm|\,|C_{\max}$ problem had space complexity exponential in the number of machines. This settles an open question raised by Lenstra et al. [3].
2. *A 2-approximation algorithm.* We propose a 2-approximation algorithm for the resource allocation problem with an arbitrary number of agents, for the special case of uniform costs, in which each agent incurs the same cost to access a resource from another agent, irrespective of the agent the resource is allocated to. This is achieved by solving a linear programming relaxation of the problem, and then rounding the fractional solution.
3. *Hardness of approximation.* We show that the general resource allocation problem cannot be approximated within a factor better than 3/2 in polynomial time, unless P=NP. We achieve this by giving an approximation preserving reduction from the $R|\,|C_{\max}$ problem to the resource allocation problem. In [3], it had been shown that the former problem cannot be approximated better than 3/2 in polynomial time, unless P=NP, hence a similar result holds for the resource allocation problem, too. This reduction also establishes a direct connection between the resource allocation problem and the $R|\,|C_{\max}$ scheduling problem.

Related work: Lenstra et al. [3] presented a 2-approximation algorithm for the $R|\,|C_{\max}$ problem, based on a linear programming relaxation and rounding. For the case of a fixed number of machines, Horowitz and Sahni [2] gave the first FPTAS, which, however, has exponential space requirements. Lenstra et al. [3] derived a PTAS for this problem, which has better space complexity. In their paper, the authors mentioned that, "An interesting open question is whether this result can be strengthened to give a *fully* polynomial approximation scheme for fixed values of m, where the space required is bounded by a polynomial in the input size, m, and $1/\epsilon$ (or, even better, $\log(1/\epsilon)$)." We settle this open question in the affirmative in this paper. Azar et al. [13] gave an FPTAS for this problem for fixed m for any l_p-norm, but they do not analyze the space complexity of their approximation scheme.

The Santa Claus problem was first studied by Lipton et al. [11]. Bezàkovà and Dani [12] proposed a linear factor approximation algorithm for this problem, which is based on a linear programming relaxation and rounding; our rounding procedure is similar to the rounding procedure used in their paper. Bansal and

Sviridenko [10] obtained a tighter approximation algorithm for the restricted assignment version of the problem, where each resource can be allocated to only a subset of the agents, and each such agent has the same utility for that resource. As of now, no FPTAS has been proposed for the Santa Claus problem with a fixed number of agents.

The focus of this paper will be mainly on the resource allocation problem, since this problem was our original motivation for taking up this study. We will refer to the $Rm||C_{\max}$ problem and the Santa Claus problem whenever our techniques for the resource allocation problem also apply to these two problems. We begin by giving an integer programming formulation of the resource allocation problem. Let x_{ik} be a variable which is 1 if the kth resource is given to agent i, otherwise it is 0. Then the total cost incurred by agent i is $\sum_{k=1}^{n}\sum_{j=1}^{m} c_{ij}^{k} x_{jk}$. An integer programming formulation of the resource allocation problem is given by

$$\begin{aligned}
& & \min \quad & S & & \\
\text{s.t.} \quad & \sum_{k=1}^{n}\sum_{j=1}^{m} c_{ij}^{k} x_{jk} \leq S & & & \text{for } i = 1,\dots,m, \\
& \sum_{i=1}^{m} x_{ik} = 1 & & & \text{for } k = 1,\dots n, \\
& x_{ik} \in \{0,1\} & & & \text{for } i = 1,\dots,m, \quad k = 1,\dots,n.
\end{aligned}$$

2 An FPTAS for a Fixed Number of Agents

In this section, we give an FPTAS for the resource allocation problem with a fixed number of agents. We first discuss a polynomial-time procedure to compute an approximate Pareto-optimal frontier for general multi-objective optimization problems. We then show that using the approximate Pareto-optimal frontier, we can get an approximate solution for the resource allocation problem. Subsequently, we use this technique for obtaining an FPTAS for the $Rm||C_{\max}$ problem and the Santa Claus problem as well, and then extend it to the case of general l_p-norms, other norms, and beyond.

2.1 Formulation of the FPTAS

An instance π of a multi-objective optimization problem Π is given by a set of m functions $f_1,\dots,f_m$. 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 optimization problems.

Definition 1. *Let π be an instance of a multi-objective optimization problem. A* Pareto-optimal frontier *(with respect to minimization), 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 [14]). One way of getting around this problem is to look at an approximate Pareto-optimal frontier, which is defined below.

Definition 2. *Let π be an instance of a multi-objective optimization 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.

Papadimitriou and Yannakakis [9] showed that whenever m is fixed, there is always an approximate Pareto-optimal frontier that has polynomially many elements.

Theorem 1 (Papadimitriou and Yannakakis [9]). *Let π be an instance of a multi-objective optimization problem. For any $\epsilon > 0$ and for fixed m, there is an ϵ-approximate Pareto-optimal frontier $P_\epsilon(\pi)$ whose cardinality is bounded by a polynomial in $|\pi|$ and $1/\epsilon$.*

Let us consider the following optimization problem:

$$\text{minimize } g(x) = \max_{i=1,\ldots,m} f_i(x), \qquad x \in X. \tag{1}$$

We show that if an approximate Pareto curve can be constructed in polynomial time, then there is an FPTAS to solve this min-max problem.

Lemma 1. *There is at least one optimal solution x^* to* (1) *such that $x^* \in P(\pi)$.*

Proof. Let $\hat{x}$ be an optimal solution of (1). Suppose $f_k(\hat{x})$ is the maximum among all function values for $\hat{x}$; that is, $f_k(\hat{x}) \geq f_i(\hat{x})$ for all $i = 1, \ldots, m$. Suppose $\hat{x} \notin P(\pi)$. Then there exists $x' \in P(\pi)$ such that $f_i(x') \leq f_i(\hat{x})$ for $i = 1, \ldots, m$. Therefore, $f_i(x') \leq f_k(\hat{x})$ for all i, that is $\max_{i=1,\ldots,m} f_i(x') \leq f_k(\hat{x})$, or $g(x') \leq g(\hat{x})$. Thus x' minimizes the function g and is in $P(\pi)$. □

Lemma 2. *Let $\hat{x}$ be a solution in $P_\epsilon(\pi)$ that minimizes $g(x)$ over all points $x \in P_\epsilon(\pi)$. Then $\hat{x}$ is a $(1+\epsilon)$-approximate solution of* (1)*; that is, $g(\hat{x})$ is at most $(1+\epsilon)$ times the value of an optimal solution to* (1).

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

From these two lemmas, we get the following theorem regarding the existence of an FPTAS for solving (1).

Theorem 2. *Suppose there is an algorithm that computes $P_\epsilon(\pi)$ in time polynomial in $|\pi|$ and $1/\epsilon$ for a fixed value of m. Then there is an FPTAS for solving the min-max optimization problem* (1).

Thus, the only thing we are left with is to find a polynomial-time algorithm for computing an approximate Pareto-optimal frontier. Papadimitriou and Yannakakis [9] give a necessary and sufficient condition under which such a polynomial-time algorithm exists.

Theorem 3 (Papadimitriou and Yannakakis [9]). *Let m be fixed, and let $\epsilon, \epsilon' > 0$ be such that $(1-\epsilon')(1+\epsilon) = 1$. One can determine a $P_\epsilon(\pi)$ in time polynomial in $|\pi|$ and $1/\epsilon$ if and only 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 $O((p(|\pi|)/\epsilon)^m)$ 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, we only need to solve the gap problem to get a $(1+\epsilon)$-approximate solution for the min-max problem. This is accomplished in a manner similar to that given in [9]. Our description here is with respect to minimization problems; a similar description for maximization problems can be found in [9].

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 $x \in X$, 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, 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 $\sum_{j=1}^{d} c_j x_j = C$?

Theorem 4. *Suppose we can solve the exact version of the problem in pseudo-polynomial time, then there is an FPTAS for solving* (1).

Proof. The gap problem can be solved by making at most $(r + 1)^m$ calls to the pseudo-polynomial time algorithm, 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 Theorems 2 and 3. □

Now we give a pseudo-polynomial time algorithm for solving the exact version of the resource allocation problem for a fixed number of agents. The exact version for resource allocation 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 that $\sum_{j=1}^{m} x_{jk} = 1$ for $k = 1, \ldots, n$, and $x_{jk} \in \{0, 1\}$? 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.

Thus, the above pseudo-polynomial algorithm implies the following theorem.

Theorem 5. *There is an FPTAS for the resource allocation problem with a fixed number of agents.*

2.2 Space Complexity of the FPTAS

A straightforward implementation of the above algorithm will have substantial storage requirements. The bottleneck for space requirements appears at two places: one is storing the approximate Pareto-optimal frontier, and the other is in solving the exact problem. However, by a careful implementation of the algorithm, the storage requirements can be reduced significantly. We give an outline below for a space-efficient implementation of the above algorithm.

1. We do not need to store all the corner points of the smaller boxes into which the region of possible objective function values has been divided. By simply iterating over the corner points using loops, we can cover all the corner points.

2. We also do not need to store the approximate Pareto-optimal frontier, as it is sufficient to store the current best solution obtained after solving each gap problem.

3. When solving the exact problem using the depth-first search algorithm, we do not need to generate the whole graph explicitly. The only data we need to store in the execution of the depth-first search algorithm are the stack corresponding to the path traversed in the graph so far (the path length is at most n), and the coefficients of the modified objective function. There are mn coefficients that need to be stored, and the maximum magnitude of each coefficient is $\mathrm{O}((m^2n^2/\epsilon)^{m+1})$, thus the space complexity of the FPTAS is $\mathrm{O}(m^2 n \log(mn/\epsilon))$.

Thus, we have the following theorem.

Theorem 6. *There is an FPTAS for the resource allocation problem whose space requirements are polynomial in m, n and $\log(1/\epsilon)$.*

2.3 An FPTAS for Scheduling on Unrelated Parallel Machines and the Santa Claus 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 m objective functions in this case are given by $f_i(x) = \sum_{k=1}^{n} p_{ik}x_{ik}$, and the set X is given by $\sum_{i=1}^{m} x_{ik} = 1$ for each $k = 1, \ldots, n$, and $x_{ik} \in \{0, 1\}$. The Santa Claus problem is similar to this scheduling problem, except that the objective here is to maximize the minimum execution time over all the machines.

The exact version of the $Rm||C_{\max}$ problem and the Santa Claus problem is the same as that for the resource allocation problem, and hence we get an FPTAS for either problem for fixed m. For the $Rm||C_{\max}$ problem, we obtain the first FPTAS that has space requirements which are polynomial in m, n

and $\log(1/\epsilon)$, whereas all the previously obtained FPTASes for this problem had space complexity exponential in m. For the Santa Claus problem, we give the first FPTAS for a fixed number of agents. We therefore have the following theorem.

Theorem 7. *There are FPTASes for the* $Rm||C_{\max}$ *problem and the Santa Claus problem with a fixed number of agents whose space requirements are polynomial in* n, m, *and* $\log(1/\epsilon)$.

2.4 FPTAS for Any Norm

The above technique for obtaining an FPTAS in fact can be extended to include any norm used for combining the different objective functions. More generally, let $h : \mathbb{R}_+^m \to \mathbb{R}_+$ be any function that satisfies
(i) $h(y) \leq h(y')$ for all $y, y' \in \mathbb{R}_+^m$ such that $y_i \leq y'_i$ for all $i = 1, \ldots, m$, and
(ii) $h(\lambda y) \leq \lambda h(y)$ for all $y \in \mathbb{R}_+^m$ and $\lambda > 1$.

Consider the following generalization of the optimization problem given by (1):

$$\text{minimize } g(x) = h(f(x)), \qquad x \in X. \tag{2}$$

Then Lemma 1 and 2 can be easily generalized as follows.

Lemma 3. *There is at least one optimal solution* x^* *to* (2) *such that* $x^* \in P(\pi)$.

Lemma 4. *Let* $\hat{x}$ *be a solution in* $P_\epsilon(\pi)$ *that minimizes* $g(x)$. *Then* $\hat{x}$ *is a* $(1+\epsilon)$*-approximate solution of* (2)*; that is,* $g(\hat{x})$ *is at most* $(1+\epsilon)$ *times the optimal value of* (2).

These two lemmata then imply that the technique given in this section can be used to obtain an FPTAS for (2). The only difference is in selecting the solution from the approximate Pareto-optimal frontier: we have to choose the solution which is the best according to the given h. Thus we have the following theorem.

Theorem 8. *There is an FPTAS for the resource allocation problem, the problem of scheduling jobs on unrelated parallel machines, and the Santa Claus problem with fixed* m *when the objectives for the different agents/machines are combined into one using a function* h *that satisfies* (i) *and* (ii). *Moreover, this algorithm can be made to run with space requirements that are polynomial in* m, n, *and* $\log(1/\epsilon)$.

3 A 2-Approximation Algorithm for the Uniform Cost Case

Recall that in the case of the resource allocation problem with uniform costs, for each agent i and each resource k, $c_{ij}^k = c_i^k$ for all $j \neq i$, and $c_{ii}^k = 0$. Let $A_k(s)$

denote the set of all agents such that if resource k is allocated to an agent in this set, the cost that any other agent will have to pay to access resource k is no more than s.

We will consider a parametric linear programming relaxation of the problem, in which we have the constraint that no agent has a cost of more than s in the relaxed solution. For each resource k, we consider only the agents in the set $A_k(s)$ as possible candidates for allocating that resource. We show that if this parametric linear program has a feasible solution, then an extreme point of the feasible set of the linear program can be rounded to an integer solution in which each agent has cost no more than $2s$.

Theorem 9. *For $s \in \mathbb{Z}_+$, consider the following set of linear inequalities, which we denote by $LP(s)$:*

$$\sum_{k=1}^{n} \sum_{j \in A_k(s)} c_{ij}^k x_{jk} \leq s \qquad \text{for } i = 1, \ldots, m, \tag{3a}$$

$$\sum_{i \in A_k(s)} x_{ik} = 1 \qquad \text{for } k = 1, \ldots, n, \tag{3b}$$

$$x_{ik} \geq 0 \qquad \text{for } k = 1, \ldots, n, \quad i \in A_k(s). \tag{3c}$$

Suppose $LP(s)$ has a feasible solution, then, for the case of uniform costs, one can find $x_{ik}^R \in \{0, 1\}$ in polynomial time such that

$$\sum_{k=1}^{n} \sum_{j \in A_k(s)} c_{ij}^k x_{jk}^R \leq 2s \qquad \text{for } i = 1, \ldots, m, \tag{4a}$$

$$\sum_{i \in A_k(s)} x_{ik}^R = 1 \qquad \text{for } k = 1, \ldots, n. \tag{4b}$$

Proof. Let x^{LP} be an extreme point of the non-empty polytope defined by the inequalities of $LP(s)$. Let v be the total number of variables defining the system $LP(s)$. There are $v+m+n$ inequalities in $LP(s)$. Since $LP(s)$ has v variables, at any extreme point of this polytope, at least v linearly independent inequalities will be satisfied with equality. Hence, at most $m + n$ inequalities will not be satisfied with equality. Therefore, it follows from (3c) that at most $m+n$ variables will have a non-zero value.

Consider the bipartite graph G in which one of the partitions has nodes corresponding to each agent, and the other partition has nodes corresponding to each resource. There is an edge between agent i and resource k in G if $x_{ik}^{LP} > 0$. In this graph, the number of edges is less than or equal to the number of nodes. For the $R||C_{\max}$ problem, which has a similar integer programming formulation, Lenstra et al. [3] showed that each connected component of G also has the property that the number of edges is less than or equal to the number of nodes. This result holds here as well.

We now construct an integral solution x^R by rounding the fractional solution. Let G' be a connected component of G. The rounding is performed in two stages. In the first stage, the following two operations are performed on G' repeatedly, in the given order.

1. For all resource nodes k such that in G', exactly one edge, say (i, k), is incident to it, we set $x^R_{ik} = 1$, and remove all such resource nodes and the edges incident to these nodes from G'.
2. For all agent nodes i such that there is exactly one edge, say (i, k), incident to it, we set $x^R_{ik} = 0$, and remove all such agent nodes and all the edges incident to these nodes from G'.

The first stage of rounding ends when the above two operations can no longer be performed. Let the resulting subgraph after the first stage of rounding be G''. Note that in the first stage, whenever we are deleting a node, we are also deleting at least one edge from the graph. Hence after the first stage, the number of edges is still less than or equal to the number of nodes in G''. For the second stage, there are three possibilities.

1. There are no nodes corresponding to resources in G''. This means that all resources in this subgraph have already been allocated to some agent. In this case we are done for G''.
2. There are some nodes corresponding to resources in G'', but there are no edges incident to these resource nodes. That is, some of the resources in G' have not yet been assigned to any of the agents. In this case, each such resource is assigned to one of the agents to which it was incident before the starting of the rounding procedure.
3. If both the above cases do not hold, then each node in G'' has at least two edges incident to it. Since the number of edges is less than or equal to the number of nodes, this component is actually a cycle, and the number of agent nodes is the same as the number of resource nodes. In this component, there is now a perfect matching between the agent nodes and the resource nodes. We find any perfect matching in this component, and for each matching edge (i, k) we set $x^R_{ik} = 1$. All the remaining variables corresponding to G' whose values have not been determined yet, are assigned the value zero.

This rounding procedure is performed on each connected component of G to get a 0/1-solution x^R. Note that x^R_{ik} satisfies the constraint (4b), since each resource is allocated to exactly one of the agents. Also, for each agent i, there is at most one resource, say $r(i)$, for which the LP solution was fractional, and in the integral solution that resource was not allocated to i, but was instead allocated to agent $i' \in A_{r(i)}(s)$. This is because in the first stage of rounding, an agent node is deleted only when there is just one resource node in the graph to which it remains incident to, and hence it does not get that resource. And in the second stage, in the third case, there will be exactly one resource to which an agent is incident to, but that resource is not allocated to the agent.

For an agent i, define a partition of the resources into $R^i_{=0}$ and $R^i_{>0}$ as follows: $R^i_{=0} = \{k : x^{LP}_{ik} = 0\}$, and $R^i_{>0} = \{k : x^{LP}_{ik} > 0\}$. For all $i \in \{1, \ldots, m\}$,

$$\begin{aligned}
\sum_{k=1}^{n} \sum_{j \in A_k(s)} c^k_{ij} x^R_{jk} &= \sum_{k \in R^i_{=0}} \sum_{j \in A_k(s)} c^k_i x^R_{jk} + \sum_{k \in R^i_{>0}} \sum_{j \in A_k(s)} c^k_{ij} x^R_{jk} \\
&= \sum_{k \in R^i_{=0}} \sum_{j \in A_k(s)} c^k_i x^{LP}_{jk} + \sum_{k \in R^i_{>0}} \sum_{j \in A_k(s)} c^k_{ij} x^R_{jk} \qquad (5a) \\
&\leq \sum_{k \in R^i_{=0}} \sum_{j \in A_k(s)} c^k_i x^{LP}_{jk} + \sum_{k \in R^i_{>0}} \sum_{j \in A_k(s)} c^k_{ij} x^{LP}_{jk} + c^{r(i)}_{ii'} \qquad (5b) \\
&= \sum_{k=1}^{n} \sum_{j \in A_k(s)} c^k_{ij} x^{LP}_{jk} + c^{r(i)}_{ii'} \\
&\leq s + s = 2s. \qquad (5c)
\end{aligned}$$

The equality in (5a) follows from the fact that for each resource k, $\sum_{j \in A_k(s)} x^{LP}_{jk} = \sum_{j \in A_k(s)} x^R_{jk} = 1$, and also because we are dealing with the case of uniform costs. The inequality in (5b) holds because for each agent i, there is at most one resource $r(i)$ such that $x^{LP}_{i,r(i)} > 0$, but $x^R_{i,r(i)} = 0$. And finally, the inequality in (5c) is true by the definition of the set $A_{r(i)}(s)$, $c^{r(i)}_i \leq s$, and (3a). □

To get a 2-approximation algorithm for the problem that runs in polynomial time, one starts by choosing a trivial lower and upper bound on the optimum value of the objective function. The lower bound can be $\min\{c^k_{ij}\}$, and the upper bound can be $mn \max\{c^k_{ij}\}$. Then, by adopting a binary search procedure, one can find the minimum integer value of s, say s^*, for which $LP(s)$ is feasible, and get a corresponding vertex x^{LP} of the non-empty polytope in polynomial time by using the ellipsoid algorithm [15]. Clearly, s^* is a lower bound on the optimal objective function value of the resource allocation problem. Using the above rounding procedure, one can obtain a rounded solution whose value is at most $2s^*$. We therefore obtain a 2-approximation algorithm for the resource allocation problem with uniform costs.

4 Hardness of Approximation

In this section, we give a hardness of approximation result for the resource allocation problem with general costs.

Theorem 10. *There is no polynomial-time algorithm that yields an approximation ratio smaller than* 3/2 *for the resource allocation problem, unless P=NP.*

Proof. We prove this by a reduction from the problem of scheduling jobs on unrelated parallel machines $(R||C_{\max})$, which cannot have a better than 3/2-approximation algorithm, unless P=NP [3].

Consider an instance of the $R||C_{\max}$ problem with m machines and n jobs, where the processing time of job k on machine i is p_{ik}. Let $p_{\max} = \max\{p_{ik}\}$. We construct a corresponding instance of the resource allocation problem as follows. There are $2m$ agents and n resources. For $i, j \in \{1, \ldots, m\}, i \neq j$, let $c_{ij}^k = np_{\max} + 1$, and $c_{i,m+i}^k = p_{ik}$. All other cost coefficients are zero. Then, in any optimal allocation of resources in the resource allocation problem, all the resources will be distributed among the agents $m+1, \ldots, 2m$. It is easy to see that if there is an optimal solution of the $R||C_{\max}$ instance in which job k is allocated to machine $m(k)$, there is a corresponding optimal solution for the resource allocation problem in which resource k is allocated to agent $m + m(k)$, and vice-versa. Also, the optimal objective function value of both instances will be the same.

Thus, if the resource allocation problem could be approximated better than $3/2$ in polynomial time, then so can the $R||C_{\max}$ problem, which is impossible, unless P=NP [3]. □

Acknowledgments

The first author thanks Mainak Chaudhuri and Maunendra Sankar Desarkar for fruitful discussions on framing the resource allocation problem. The authors also thank Retsef Levi for pointing out the work by Bansal and Sviridenko [10]. This research was supported by NSF awards #0426686 and #0700044, and by ONR grant N00014-08-1-0029.

References

1. Laudon, J., Lenoski, D.: The SGI origin: A ccNUMA highly scalable server. In: Proceedings of the 24th International Symposium on Computer Architecture, Boulder, CO, pp. 241–251 (1997)
2. Horowitz, E., Sahni, S.: Exact and approximate algorithms for scheduling nonidentical processors. Journal of the ACM 23, 317–327 (1976)
3. Lenstra, J.K., Shmoys, D.B., Tardos, É.: Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 46, 259–271 (1990)
4. Hochbaum, D.S., Shmoys, D.B.: Using dual approximation algorithms for scheduling problems: Theoretical and practical results. Journal of the ACM 34, 144–162 (1987)
5. Woeginger, G.J.: When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (FPTAS)? INFORMS Journal on Computing 12, 57–74 (2000)
6. Safer, H.M., Orlin, J.B., Dror, M.: Fully polynomial approximation in multi-criteria combinatorial optimization. Technical report, Operations Research Center, MIT (2004)
7. Safer, H.M., Orlin, J.B.: Fast approximation schemes for multi-criteria combinatorial optimzation. Technical report, Operations Research Center, MIT (1995)
8. Safer, H.M., Orlin, J.B.: Fast approximation schemes for multi-criteria flow, knapsack, and scheduling problems. Technical report, Operations Research Center, MIT (1995)

9. Papadimitriou, C.H., Yannakakis, M.: 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 (2000)
10. Bansal, N., Sviridenko, M.: The Santa Claus problem. In: Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, pp. 31–40 (2006)
11. Lipton, R.J., Markakis, E., Mossel, E., Saberi, A.: On approximately fair allocations of indivisible goods. In: Proceedings of the 5th ACM Conference on Electronic Commerce, New York, NY, pp. 125–131 (2004)
12. Bezàkovà, I., Dani, V.: Allocating indivisible goods. ACM SIGecom Exchanges 5, 11–18 (2005)
13. Azar, Y., Epstein, L., Richter, Y., Woeginger, G.J.: All-norm approximation algorithms. Journal of Algorithms 52, 120–133 (2004)
14. Hansen, P.: Bicriterion path problems. In: Fandel, G., Gál, T. (eds.) Multiple Criteria Decision Making: Theory and Application. Lecture Notes in Economics and Mathematical Systems, vol. 177, pp. 109–127. Springer, Heidelberg (1980)
15. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Berlin (1988)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. W. H. Freeman, New York (1979)

Appendix

Lemma 5. *The resource allocation problem with uniform costs is NP-hard for two agents, and strongly NP-hard in general.*

Proof. The proof of NP-hardness for the two-agents case is by reduction from PARTITION [16]. Consider an instance of PARTITION given by a set A of n elements, where element $a \in A$ has size $s(a) \in \mathbb{Z}_+$. We construct an instance of the resource allocation problem with two agents and n resources as follows: $c^a_{12} = c^a_{21} = s(a)$ for each $a \in A$, and $c^a_{ii} = 0$ for $i = 1, 2$. Then, A can be partitioned into two sets of equal size if and only if the optimal solution for the given resource allocation problem has cost $\sum_{a \in A} s(a)/2$.

The strong NP-hardness proof for the general case is by a reduction from 3-PARTITION [16]. Let an instance of this problem be given by the set $A = \{a_1, \ldots, a_{3m}\}$, with $\sum_{a \in A} s(a) = mB$. The corresponding instance of the resource allocation problem is constructed as follows: There are m agents, and $3m$ resources. For each agent i, $c^k_{ij} = s(a_k)$ for $k = 1, \ldots, 3m; j = 1, \ldots, m, i \neq j$, and $c^k_{ii} = 0$. Then the answer to the 3-PARTITION instance is "Yes" if and only if the optimal solution to the given resource allocation problem has cost $(m-1)B$. □