Approximation Techniques for Utilitarian Mechanism Design

Patrick Briest[*] Piotr Krysta[†] Berthold Vöcking[‡]

ABSTRACT

This paper deals with the design of efficiently computable incentive compatible, or truthful, mechanisms for combinatorial optimization problems with multi-parameter agents. We focus on approximation algorithms for NP-hard mechanism design problems. These algorithms need to satisfy certain monotonicity properties to ensure truthfulness. Since most of the known approximation techniques do not fulfill these properties, we study alternative techniques.

Our first contribution is a quite general method to transform a pseudopolynomial algorithm into a monotone FPTAS. This can be applied to various problems like, e.g., *knapsack, constrained shortest path*, or *job scheduling with deadlines*. For example, the monotone FPTAS for the *knapsack problem* gives a very efficient, truthful mechanism for *single-minded multi-unit auctions*. The best previous result for such auctions was a 2-approximation. In addition, we present a monotone PTAS for the *generalized assignment problem* with any bounded number of parameters per agent.

The most efficient way to solve packing integer programs (PIPs) is LP-based randomized rounding, which also is in general not monotone. We show that primal-dual greedy algorithms achieve almost the same approximation ratios for PIPs as randomized rounding. The advantage is that these algorithms are inherently monotone. This way, we can significantly improve the approximation ratios of truthful mechanisms for various fundamental mechanism design problems like *single-minded combinatorial auctions (CAs), unsplittable flow routing* and *multicast routing*. Our approximation algorithms can also be used for the winner determination in *CAs with general bidders* specifying their bids through an oracle.

[*]Dept. of Computer Science, Dortmund University, Germany. patrick.briest@cs.uni-dortmund.de. Supported by DFG grant Kr 2332/1-1 within Emmy Noether program.

[†]Dept. of Computer Science, Dortmund University, Germany. piotr.krysta@cs.uni-dortmund.de. Supported by DFG grant Kr 2332/1-1 within the Emmy Noether program.

[‡]Dept. of Computer Science, RWTH Aachen, Germany. voecking@cs.rwth-aachen.de. Partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS) and by DFG grant Vo 889/1-2.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Algorithms, Economics, Theory

Keywords

Mechanism design, approximation algorithms, combinatorial and multi-unit auctions, primal-dual method, enumeration techniques

1. INTRODUCTION

Mechanism design deals with algorithmic problems in a game theoretic setting in which the input data is not directly available to the algorithm. Instead there are several agents each of which knows only a part of the input. As such agents are capable of manipulating the algorithm by lying about their parts of the input data, the algorithm should work in such a way that reporting the true input data is a dominant strategy for all agents. Such algorithms are called incentive compatible or truthful mechanisms. The study of efficiently computable truthful mechanism for combinatorial optimization problems was initiated by Nisan and Ronen [24] investigating problems like shortest paths, minimum spanning trees, and makespan scheduling. Such mechanisms have since been widely studied for many problems, e.g., [1, 2, 3, 6, 21, 23].

In this paper, we consider the design of efficiently computable truthful mechanisms for combinatorial optimization problems with multi-parameter agents. The objective of these mechanisms is to optimize the social benefit, i.e., to maximize either the sum of the valuations or to minimize the sum of the costs over all agents. As many relevant problems in this area are NP-hard, our focus lies on approximation algorithms. Unfortunately, most of the known techniques for devising approximation algorithms cannot be used for mechanism design as they do not satisfy the required monotonicity properties. Even the most basic techniques like the transformation of a pseudopolynomial time algorithm into a fully polynomial time approximation scheme (FPTAS), or the distinction of elements according to the size of their parameters as often used in the design of a polynomial time approximation scheme (PTAS) are not monotone. Also more advanced techniques like randomized rounding for packing integer programs (PIPs) cannot directly be used for mechanism design as also they are not monotone.

We study modifications and alternatives to these techniques that behave in a monotone way with respect to the parameters of the agents and, hence, are suitable for mechanism design. Before we present these techniques and their applications, let us introduce the concepts underlying utilitarian mechanism design in a formal way.

1.1 Technical background

A famous example of problems studied in the context of mechanism design is given by combinatorial auctions (CAs), in which a single auctioneer wants to sell some collection of goods to a set of potential buyers. To have some unified notation for a wider class of problems, we will consider a *utilitarian mechanism design (maximization) problem* Π defined by a finite set of *objects* $\mathcal{A}$, a set of feasible outputs $O_\Pi \subseteq \mathcal{A}^n$ and a set of n agents. Each agent declares a set of objects $S_i \subseteq \mathcal{A}$ and a valuation function $v_i : \mathcal{P}(\mathcal{A}) \times \mathcal{A}^n \to \mathbb{R}$ by which she values all possible outputs. Given a vector $S = (S_1, \ldots, S_n)$ of declarations we are interested in output $o^* \in O_\Pi$ maximizing the objective function value, i.e., $o^* \in \operatorname{argmax}_{o \in O_\Pi} \sum_{i=1}^n v_i(S_i, o)$. This objective is usually referred to as *social welfare*. In combinatorial auctions, an object a corresponds to a subset of goods to be sold. Each agent declares all the subsets she is interested in and the prices she would be willing to pay. An output specifies the sets to be allocated to the agents.

We are going to deal with a limited type of agents called *single–minded* that were introduced by Lehmann et al. [21]. Let $R_\preceq \subseteq \mathcal{A}^2$ be a reflexive and transitive relation on $\mathcal{A}$, such that there exists a special object $\varnothing \in \mathcal{A}$ with $\varnothing \preceq a$ for any $a \in \mathcal{A}$ to model the situation in which some agent does not contribute to the solution at all. For $a, b \in \mathcal{A}$ we will denote $(a, b) \in R_\preceq$ by $a \preceq b$. (For some intuition on $\preceq$ see an example about CAs below.) The single–minded agent i declares a single object a_i rather than a set of objects and is fully defined by her *type* (a_i, v_i), with $a_i \in \mathcal{A}$ and $v_i > 0$. The valuation function introduced earlier reduces to

$$v_i(a_i, o) = \begin{cases} v_i, & \text{if } a_i \preceq o_i \\ 0, & \text{else.} \end{cases}$$

Agent i is called *known* if object a_i is known to the mechanism [23]. We will cover a more general situation and assume that this is not the case and, thus, agents are unknown. Intuitively, each a_i corresponds to an object agent i offers to contribute to the solution, v_i describes her valuation of any output o that indeed selects a_i. In CAs, relation $R_\preceq$ is set inclusion. An agent interested in set S will obviously be satisfied by any S' with $S \subseteq S'$. Note, that our agents are not 1–parameter–agents as in [2], but are more general, since any number of parameters can be hidden within an element $a \in \mathcal{A}$, e.g., for a networking problem each a_i might describe an edge including its incident vertices, its length and possibly further parameters. For ease of notation we let $(a, v) = ((a_1, v_1), \ldots, (a_n, v_n))$, $(a_{-i}, v_{-i}) = ((a_1, v_1), \ldots, (a_{i-1}, v_{i-1}), (a_{i+1}, v_{i+1}), \ldots, (a_n, v_n))$ and finally $((a_i, v_i), (a_{-i}, v_{-i})) = (a, v)$.

A *mechanism* $M = (A, p)$ consists of an algorithm A computing a solution $A(a, v) \in O_\Pi$ and an n–tuple $p(a, v) = (p_1(a, v), \ldots, p_n(a, v)) \in \mathbb{R}^n_+$ of payments collected from the agents. If $a_i \preceq A(a, v)_i$ we say that agent i is selected. By $S(A(a, v)) = \{i \mid a_i \preceq A(a, v)_i\}$ we refer to the set of selected agents.

Agent i's type is considered as her private knowledge. Consequently, the types declared by agents may not necessarily match their true types. To reflect this difference, let (a_i^*, v_i^*) refer to agent i's true type and (a_i, v_i) be the type she declares. But why do agents lie about their types? Given an output $o \in O_\Pi$, we say that agent i has *utility* $u_i(a, v) = v_i(a_i^*, o) - p_i(a, v)$. Each agent's goal is to maximize her utility. To achieve this, she will try to manipulate the mechanism by declaring a false type if a gain in utility can be expected. A mechanism is called *truthful*, or *incentive compatible*, if no agent i can gain by lying about her type, i.e., given declarations (a_{-i}, v_{-i}) we have that $u_i((a_i^*, v_i^*), (a_{-i}, v_{-i})) \geq u_i((a_i, v_i), (a_{-i}, v_{-i}))$ for any $(a_i, v_i) \neq (a_i^*, v_i^*)$.

A sufficient condition for truthfulness of approximate mechanisms for single-minded CAs was first given by Lehmann et al. [21]. Their results can easily be adopted for our more general scenario. We say that an algorithm A is *monotone* with respect to $R_\preceq$ if

$$i \in S(A((a_i, v_i), (a_{-i}, v_{-i}))) \Rightarrow i \in S(A((a_i', v_i'), (a_{-i}, v_{-i})))$$

for any $a_i' \preceq a_i$ and $v_i' \geq v_i$. Intuitively, we require that a winning declaration (a_i, v_i) remains winning if an object a_i', smaller according to $R_\preceq$, and a higher valuation v_i' are declared. If we fix declarations (a_{-i}, v_{-i}) and object a_i declared by i, we observe that algorithm A defines a *critical value* θ_i^A, i.e., the minimum valuation v_i that makes (a_i, v_i) winning, i.e., $i \in S(A((a_i, v_i), (a_{-i}, v_{-i})))$ for any $v_i > \theta_i^A$ and $i \notin S(A((a_i, v_i), (a_{-i}, v_{-i})))$ for any $v_i < \theta_i^A$. The *critical value payment scheme* p^A associated with algorithm A is then defined by $p_i^A(a, v) = \theta_i^A$, if $i \in S(A(a, v))$, and $p_i^A(a, v) = 0$, otherwise. The critical value for any fixed agent i can always be computed, e.g., by performing binary search on interval $[0, v_i]$ and repeatedly running algorithm A to check whether i is selected. Note, that mechanism $M_A = (A, p^A)$ is *normalized*, i.e., agents that are not selected pay 0. We say that algorithm A is *exact*, if for declarations (a, v) we have $A(a, v)_i = a_i$ or $A(a, v)_i = \varnothing$ for all i. In analogy to [21] we obtain the following result.

PROPOSITION 1. *Let A be a monotone and exact algorithm for some utilitarian problem Π and single–minded agents. Then mechanism $M_A = (A, p^A)$ is truthful.*

1.2 Our contributions

We study algorithmic techniques that help to design monotone approximation algorithms for utilitarian mechanism design problems. In order to demonstrate the applicability and strength of these techniques, we apply them to various well-known optimization problems and show that they lead to significant improvements on the previously known approximation ratios. For formal definitions of the considered optimization problems see Sections 2–7.

Enumerative methods for designing approximation schemes. The standard way to obtain FPTASs for optimization problems is to round the coefficients of the objective function in such a way that the modified optimization problem can be solved efficiently by a pseudopolynomial algorithm. Unfortunately, this technique is non-monotone as the degree of accuracy chosen by the rounding procedure depends on the outcome of the coefficients. Incentive compatible mechanisms cannot be based on such a method as agents could have an incentive to manipulate the rounding procedure in their favor by lying about their parameters [23].

We present an alternative rounding scheme that leads to monotone FPTASs. In the description of this mechanism, we explicitly distinguish between the specification of the mechanism and its efficient implementation. In the specification, we assume that the mechanism enumerates over "all possible" degrees of accuracy and, for each of these degrees, computes a solution that is optimal when all coefficients are rounded with respect to this degree of accuracy. This exhaustive enumeration method ensures that at least one of the generated solutions satisfies the approximation ratio. The mechanism selects one of these "good" solutions in a monotone way. Let us remark that it cannot simply choose the best among the enumerated solutions as this would not be monotone. An important property of the specification of the mechanism is that it is oblivious to the actual outcome of the coefficients. This way, we can ensure monotonicity. An efficient implementation of the mechanism, however, has to take into account the values of the coefficients. Depending on the size of the coefficients, it computes solutions only with respect to those degrees of accuracy that can potentially become the solution described by the specification.

Based on this enumerative method, we can present a transformation of a pseudopolynomial time algorithm into a monotone FP-

TAS. For example, we can derive a monotone FPTAS for the *knapsack problem*. This way, we obtain the first incentive compatible FPTAS for *multi–unit auctions*. The best previous result for this kind of auctions was a 2–approximation based on the greedy method for the *knapsack problem* [23]. Other interesting applications are incentive compatible mechanisms for *reverse auctions*, the *constrained shortest path problem* and the *problem of job scheduling with deadlines*. We show that these mechanisms do not only approximate the overall welfare but they are also frugal in the sense that the overpayment in comparison to the corresponding VCG mechanism can be made arbitrarily small. This shows that our FPTAS based mechanisms truly approximate the VCG mechanism both with respect to welfare and payment. More details about these results can be found in Section 2 and 3.

One of the most fundamental techniques in devising PTASs for assignment problems like the *multiple knapsack problem* or the *general assignment problem* is to partition the set elements into different groups according to the size of their parameters and to apply different methods to the elements in different groups. This explicit grouping of elements according to the size of their parameters is inherently non–monotone. We propose an alternative way to group elements based on another enumerative method. Using this approach we can guarantee that the most valuable among the elements contained in the optimal solution are also contained in one of the enumerated solutions. This is the key argument which enables us to achieve approximation factors arbitrarily close to one, i.e., we get a monotone PTAS for the *general assignment problem* and, hence, also for the less general *multiple knapsack problem*, provided that the number of resources is bounded by a constant. Thus, we obtain very efficient truthful mechanism for optimization problems in which agents come with any bounded number of secret parameters. More details about these results can be found in Section 4.

Primal-dual algorithms for routing problems and combinatorial auctions. The *unsplittable flow problem* is a widely accepted model for routing in networks. This problem falls into the class of packing integer programs (PIPs)[1]. The most general approach to approximately solve this kind of problems is randomized rounding [27, 28, 31]. Unfortunately, also this method is not monotone. We show that primal-dual algorithms can achieve essentially the same approximation ratios for these problems as randomized rounding. Our primal–dual algorithms, however, have two advantages: they are significantly faster and they are monotone. Based on the primal–dual method, we can devise the first combinatorial approximation algorithms that solve the unsplittable flow problem in polynomial time within constant factors, provided that the capacities are sufficiently large. As this algorithm is monotone with respect to valuations and demands, it also yields the first truthful mechanism for routing in general networks maximizing the network utilization up to constant factors. The best previously known mechanisms could only guarantee approximation ratios logarithmic in the size of the network [3, 6]. We can obtain this result not only for pairwise communication, corresponding to the allocation of paths, but also for communication in groups, corresponding to the allocation of multicast trees. More details appear in Section 5.

The primal–dual method enables us also to improve on the best results known for *multi–unit combinatorial auctions (CAs)*. We first use our full primal–dual machinery to design a primal–dual approximation algorithm for the winner determination problem in multi–unit CAs with general bidders given by oracles. This algorithm has best possible approximation ratio which improves on the best known ratio of combinatorial methods for the problem – see Section 6 for details. We show that this general algorithm is truthful for the case of multi–unit CAs with unknown single–minded bidders. Our approximation factor is again constant if the supply of the goods is high enough, whereas the previous known bounds are logarithmic [3, 6]. A constant–approximation mechanism based on LP randomized rounding is known in this setting, but it only works for known bidders and is truthful only in a probabilistic sense [1]. The primal–dual method has been used in context of mechanism design to obtain cost-sharing mechanisms for some covering problems, e.g., [16, 17]. Our results on truthful primal–dual mechanisms for CAs appear in Section 7.

2. MAKING AN FPTAS MONOTONE

A well known technique to construct FPTASs for a variety of weakly NP-hard utilitarian optimization problems is based upon the idea of scaling the numbers included in the input and then applying some optimal algorithm with pseudopolynomial running time. This approach, however, leads to approximation algorithms that are not monotone and, thus, not suitable for mechanism design. We will present a way to construct FPTASs that avoids this problem and leads to monotone algorithms. We will assume maximization problems, but our argumentation also applies to minimization.

We start by giving a modified definition of the notion of *bitonicity* due to Mu'alem and Nisan [23]. Given a function $f : \mathcal{A} \to \mathbb{R}$, a monotone algorithm A is called *bitonic w.r.t. f* if for any agent i:

1. If $i \in S(A(a, v))$ then $f(A((a'_i, v'_i), (a_{-i}, v_{-i}))) \geq f(A(a, v))$ for any $a'_i \preceq a_i$ and $v_i \leq v'_i$.
2. If $i \notin S(A(a, v))$ then $f(A((a'_i, v'_i), (a_{-i}, v_{-i}))) \geq f(A(a, v))$ for any $a_i \preceq a'_i$ and $v'_i \leq v_i$.

Now let Π be some utilitarian (maximization) problem as defined in Section 1.1 and let A_Π be an optimal algorithm for Π with pseudopolynomial running time $poly(n, V)$, where $V = \max_i v_i$ denotes the largest declared valuation. Assume that A_Π is monotone with respect to $R_\preceq$ and bitonic with respect to the social welfare $w(A(a, v)) = \sum_{i \in S(A(a,v))} v_i$. (For any optimal algorithm both assumptions are w.l.o.g. if we can assure that possible ties are broken according to some fixed ordering based on the sets of selected agents.) Given declarations (a, v), let $Opt(a, v)$ denote an optimal solution to Π on this instance and $w(Opt(a, v))$ the corresponding social welfare. Assume that V is a lower bound on $w(Opt(a, v))$. We use A_Π to define algorithm A^k_Π in Fig. 1. A^k_Π works by scaling agents' valuations and then applying A_Π to compute an exact solution on the modified declarations. We define $v'''_i = v''_i/\alpha_k$ to invert A^k_Π's scaling procedure without respect to the applied rounding and let $w_k(A^k_\Pi(a, v)) = \sum_{i \in S(A^k_\Pi(a,v))} v'''_i$ be the computed solution's value based on these modified valuations.

LEMMA 1. *Let $\varepsilon > 0$, $k \in \mathbb{N}$, (a, v) be a vector of declarations and $V = \max_i v_i$. Then algorithm A^k_Π has running time $poly(n, \varepsilon^{-1})$. If $2^k \leq V \leq 2^{k+1}$ then A^k_Π computes a $(1 - \varepsilon)$–approximation to $w(Opt(a, v))$. Furthermore, A is monotone and bitonic with respect to function w_k.*

PROOF. After line 3, no declaration with value higher than 2^{k+1} exists. Hence, all declarations are scaled to range $\{0, \ldots, \lfloor \frac{2n}{\varepsilon} \rfloor\}$

[1]A PIP, cf. [31], is an integer linear program of a form $\max\{c \cdot x : Ax \leq b, x \in \{0, 1\}^q\}$, where $p, q \in \mathbb{N}$, $c \in \mathbb{R}^q_{\geq 0}$, $A \in [0, 1]^{p \times q}$, $b \in [1, \infty)^p$. We could also assume that $x \in \mathbb{Z}^q_{\geq 0}$. A (0,1)-PIP is a PIP where $A \in \{0, 1\}^{p \times q}$. A column-restricted PIP, cf. [19], is a PIP in which each column of A may only assume two values, 0 and some other value in $(0, 1]$.

```
**Algorithm** $A_\Pi^k$
1   $\alpha_k := \frac{n}{\varepsilon \cdot 2^k}$;
2   **for** $i = 1, \dots, n$ **do**
3       $v_i' := \min\{v_i, 2^{k+1}\}$;
4       $v_i'' := \lfloor \alpha_k \cdot v_i' \rfloor$;
5   return $A_\Pi(a, v'')$;
```

Figure 1: The building block for monotone FPTAS.

in line 4 and running time follows from the fact that A_Π is pseudopolynomial algorithm for Π. The approximation ratio can be shown as for the well known FPTAS for the Knapsack Problem. Let S denote the solution returned by A_Π^k, Opt an optimal solution. In line 3 no declarations are changed and, defining v_i''' as above, it can be observed that $v_i - v_i''' \leq \alpha_k^{-1}$. By $2^k \leq w(Opt)$ we have $w(Opt) - w_k(Opt) \leq n\alpha_k^{-1} \leq \varepsilon \cdot w(Opt)$, and finally

$$w(S) \geq w_k(S) \geq w_k(Opt) \geq (1 - \varepsilon) \cdot w(Opt)$$

as claimed. Monotonicity and bitonicity with respect to w_k are a direct consequence of the fact that A_Π is monotone and bitonic with respect to $\alpha_k w_k$ and that rounding of the declared valuations can be seen as an application of a monotone function. $\square$

The main step towards a monotone FPTAS consists of combining the algorithms we just defined by a generalization of the toolkit presented by Mu'alem and Nisan in [23]. Given an indexed set $S = \{A_i \mid 0 \leq i < \ell\}$ of algorithms for problem Π and an indexed set $F = \{f_i \mid 0 \leq i < \ell\}$ of functions with $f_i : \mathcal{A}^n \to \mathbb{R}$ and $\ell \in \mathbb{N} \cup \{\infty\}$, the MAX–operator is defined by $MAX(S, F)(a, v) = \operatorname{argmax}\{f_i(A_i(a, v)) \mid A_i \in S \text{ and } f_i \in F\}$, where possible ties are broken according to index i. We assume that for given declarations (a, v) the maximum is indeed taken, i.e., there exists index i, such that $f_i(A_i(a, v)) = \max_j f_j(A_j(a, v))$. This ensures that the MAX–operator is well defined for infinite sets of algorithms.

LEMMA 2. *Let S and F be defined as above. If each algorithm A_i is monotone and bitonic with respect to function f_i, then algorithm $MAX(S, F)$ is monotone.*

PROOF. Assume $MAX(S, F)$ is not monotone. Then there exists an agent i and declarations (a_i, v_i), (a_i', v_i') with $a_i' \preceq a_i$ and $v_i \leq v_i'$, such that (a_i, v_i) results in being selected and (a_i', v_i') does not. For any k let $A_k(a_i, v_i)$ denote the solution returned by A_k. With agent i declaring (a_i, v_i), let A_j be the algorithm computing the solution with maximum value $f_j(A_j(a_i, v_i))$, formally $f_j(A_j(a_i, v_i)) \geq f_k(A_k(a_i, v_i))$ for $0 \leq k < \ell$. From the bitonicity of A_j it follows that $f_j(A_j(a_i', v_i')) \geq f_j(A_j(a_i, v_i))$. With agent i declaring (a_i', v_i'), let A_l be the algorithm resulting in a solution with maximum value. As before we observe that $f_l(A_l(a_i', v_i')) \geq f_k(A_k(a_i', v_i'))$ for $0 \leq k < \ell$, $f_l(A_l(a_i, v_i)) \geq f_l(A_l(a_i', v_i'))$ and, thus,

$$\begin{aligned} f_j(A_j(a_i, v_i)) &= f_j(A_j(a_i', v_i')) \\ &= f_l(A_l(a_i, v_i)) = f_l(A_l(a_i', v_i')). \end{aligned}$$

If $j = l$, then A_j is not monotone, thus $j \neq l$. We now look at our tie breaking rule. Since A_j is given preference when agent i declares (a_i, v_i), we have $j < l$. Analogously, declaring (a_i', v_i') implies $l < j$, a contradiction. $\square$

There are two main differences between the above lemma and the corresponding result in [23]. First, we cover our generalized definition of bitonicity and allow application of the MAX–operator to arbitrary functions rather than just the obtained social welfare. Actually, we do not even require that the same function is evaluated for every algorithm. Second, we explicitly allow the application to infinite sets of algorithms, which will be required for our FPTAS. We now let $S_\Pi = \{A_\Pi^k \mid 0 \leq k < \infty\}$ and $F_\Pi = \{w_k \mid 0 \leq k < \infty\}$.

LEMMA 3. *For any $\varepsilon > 0$, $MAX(S_\Pi, F_\Pi)$ is monotone and has approximation ratio $(1 - \varepsilon)$.*

PROOF. Monotonicity follows from Lemmas 1 and 2. For the approximation, consider declarations (a, v) and let $V = \max_i v_i$. Define l, such that $2^l \leq V < 2^{l+1}$. From the proof of Lemma 1 we know that $w_l(A_\Pi^l(a, v)) \geq (1-\varepsilon) \cdot w(Opt(a, v))$. Since for any k, $w(A_\Pi^k(a, v)) \geq w_k(A_\Pi^k(a, v))$, we get $MAX(S_\Pi, F_\Pi)(a, v) \geq w_l(A_\Pi^l(a, v)) \geq (1 - \varepsilon) \cdot w(Opt(a, v))$. $\square$

```
**Algorithm** $A_\Pi^{FPTAS}$
1   $V := \max_i v_i$, $Best := (\varnothing, \dots, \varnothing)$, $best := 0$;
2   **for** $j = 0, \dots, \log((1 - \varepsilon)^{-1} n) + 1$ **do**
3       $k := \lceil \log(V) \rceil - j$;
4       **if** $w_k(A_\Pi^k(a, v)) > best$ **then**
5           $Best := A_\Pi^k(a, v)$;
6           $best := w_k(A_\Pi^k(a, v))$;
7   return $Best$;
```

Figure 2: Our monotone FPTAS.

$MAX(S_\Pi, F_\Pi)$ fully specifies the output of our monotone approximation scheme. Apart from this specification, we now need to describe some implementation that guarantees polynomial running time. The design of $MAX(S_\Pi, F_\Pi)$ was focused on the requirement of maintaining monotonicity. It was crucial to ensure that the applied rounding steps do not depend on the actual input. For the remainder of this section, however, these issues can be neglected. The behavior of any final polynomial time algorithm may well depend on the input in a (seemingly) non–monotone way, as long as it implements the defined specification. To obtain polynomial running time we have to limit the number of algorithms A_Π^k that can potentially maximize $f_k(A_\Pi^k(a, v))$ by taking a closer look at the declarations (a, v). Theorem 1 shows that A_Π^{FPTAS} in Figure 2 is the desired polynomial time implementation.

THEOREM 1. *Let Π be a utilitarian mechanism design problem among single–minded agents, A_Π monotone pseudopolynomial algorithm for Π with running time $poly(n, V)$, where $V = \max_i v_i$, and assume that $V \leq w(Opt(a, v))$ for declaration (a, v). Then A_Π^{FPTAS} is a monotone FPTAS for Π.*

PROOF. We show that $A_\Pi^{FPTAS} = MAX(S_\Pi, F_\Pi)$. Let $l = \lceil \log(V) \rceil$ and consider algorithm A_Π^k for some $k > l$. By S_k we refer to the set of agents selected by A_Π^k. It follows that

$$\begin{aligned} w_k(A_\Pi^k(a, v)) &= \sum_{i \in S_k} \left\lfloor \frac{n \cdot v_i}{\varepsilon \cdot 2^k} \right\rfloor \leq \sum_{i \in S_k} \left\lfloor \frac{n \cdot v_i}{\varepsilon \cdot 2^l} \right\rfloor \\ &\leq \sum_{i \in S_l} \left\lfloor \frac{n \cdot v_i}{\varepsilon \cdot 2^l} \right\rfloor = w_l(A_\Pi^l(a, v)), \end{aligned}$$

since A_Π^l computes optimal solutions w.r.t. w_l. In case of equality our tie breaking rule ensures that the solution computed by the algorithm with smallest index is given preference. Thus, all algorithms A_Π^k, $k > l$, can be ignored. Now assume that $k < \log(V) - \log((1-\varepsilon)^{-1} n) - 1$. As a consequence, $w_k(A_\Pi^k(a, v)) \leq n \cdot 2^{k+1} < (1 - \varepsilon) \cdot V \leq (1 - \varepsilon) \cdot w(Opt(a, v))$ and, thus, A_Π^k cannot return the solution with maximum value. $\square$

We mention that, with a few small modifications, the described techniques can also be applied to minimization problems. Let us now give a list of problems that our technique can be applied to and briefly compare to previous results about monotone algorithms. Afterwards, we discuss the consequences for incentive compatible mechanisms.

Forward multi–unit auctions. A single auctioneer wants to sell m identical items to a set of n possible buyers (or bidders). Each single–minded bidder specifies the number of items she is interested in and a price that she is willing to pay. Elements in our general notation correspond to the requested and allocated numbers of items. Relation $R_{\preceq}$ describes that bidder i requesting q_i items will be satisfied also by any larger number of items. Previously the best monotone algorithm ([23]) was 2–approximate. We obtain the first monotone FPTAS for multi–unit auctions among unknown single–minded bidders with running time $O(n^3\varepsilon^{-1}\log((1-\varepsilon)^{-1}n))$.

Reverse multi–unit auctions. Here a single buyer wants to buy m identical items from a set of n possible suppliers. Non–monotone approximation algorithms for these auctions and more complex agents have been presented by Kothari et al. [20]. Their algorithm is only approximately truthful, and in particular, a possible deviation of a single bidder might be very high. We obtain an FPTAS in analogy to forward multi–unit auctions with identical running time.

Job scheduling with deadlines (JSD). Each agent i presents a job with running time t_i, deadline d_i and a price v_i she is willing to pay if her job is processed by deadline d_i. Element a_i is defined as $a_i = (t_i, d_i)$. Output for agent i can be seen as a time slot that is reserved for processing i's job. For two elements $a_i = (t_i, d_i)$ and $a'_i = (t'_i, d'_i)$ we have that $a_i \preceq a'_i$ if $t_i \le t'_i$ and $d_i \ge d'_i$. A pseudopolynomial algorithm can be obtained by dynamic programming based on ideas of Sahni [30]. The resulting FPTAS has running time $O(n^3\varepsilon^{-1}\log((1-\varepsilon)^{-1}n)))$. As in the case of multi–unit auctions a minimization variant to which our technique also applies is known in the literature as *scheduling to minimize tardiness*.

Constrained shortest path (CSP). Given a graph $G = (V, E)$, $u, w \in V$, each edge having length $l(e)$ and cost $v(e)$, and a number $L > 0$, we want to find a minimum cost path from u to w of length at most L. An element consists of an edge and its length. For two elements $a_1 = (e_1, l_1)$ and $a_2 = (e_2, l_2)$ we have that $a_1 \preceq a_2$ if $e_1 = e_2$ and $l_1 \le l_2$. FPTASs for this problem, which cannot be shown to be monotone, were given by Hassin [15] and Phillips [26]. The approach of Phillips can be modified to obtain a pseudopolynomial algorithm we need to apply our technique. We obtain an FPTAS with running time $O(mn^2\varepsilon^{-1}\log(n^2\varepsilon^{-1})\log((1-\varepsilon)^{-1}n))$, where $|V| = n$ and $|E| = m$.

Constrained minimum spanning tree (CMST). Given a graph $G = (V, E)$, each edge of length $l(e)$ and cost $v(e)$, and a number $L > 0$, we want to find a minimum cost spanning tree for G of length at most L. Again, an element consists of an edge and its length. Relation $R_{\preceq}$ is defined as before. Marathe et al. [22] give a pseudopolynomial algorithm for the case of treewidth-bounded graphs and a constant number of terminals, with running time depending on L. Running this algorithm for all values $\{0, \dots, |E| \cdot v_{max}\}$ of our objective (cost) while minimizing the other objective (length) gives the algorithm we need to apply our technique.

Proposition 1 says that algorithms to be embedded in truthful mechanisms need to be monotone and exact. We have only argued about the monotonicity so far. We will now consider the exactness. Exactness is not an issue with forward and reverse multi-unit auctions, CSP, and CMST problems. In these cases exactness of our FPTAS follows from the obvious exactness of the pseudopolynomial algorithms used. Hence, our FPTASs can be embedded in truthful mechanisms for the respective problems and unknown single–minded agents. In the case of JSD, however, the solution returned by the dynamic programming approach is exact only with respect to the running times t_i but not with respect to deadlines d_i, that is, a job might be finished before its deadline. Thus, the resulting FPTAS ensures truthfulness only if we assume that deadlines are known to the mechanism. In many application contexts the following workaround can solve the problem. For each job, the mechanism defines an additional release time. A job is returned to an agent at its release instead of its completion time. By setting the release time of each served job to its declared deadline, we obtain truthfulness also with respect to the deadlines.

3. FRUGALITY CONSIDERATIONS

We will now compare the payments of our approximate mechanisms from Section 2 to those of a normalized VCG mechanism for the same problems, which is a well established way of measuring approximate mechanisms' payment behavior [1, 7, 32]. We assume minimization problems, but our arguments apply also to maximization. Since a VCG mechanism is based on an optimal (monotone) algorithm Opt, we note that VCG payments are in fact the critical value payment scheme p^{Opt} defined by Opt. We compare the VCG mechanism $M_{Opt} = (Opt, p^{Opt})$ and a mechanism $M_A = (A, p^A)$ based on approximation algorithm A. We fix an agent i and start by comparing her critical values θ_i^{Opt} and θ_i^A in algorithms Opt and A. $Opt(a, v \mid \neg i)$ denotes an optimal solution assuming agent i must not be selected.

LEMMA 4. *Let Opt be an exact algorithm for* Π *and let* A *be monotone approximation algorithm for* Π *with approximation ratio* $(1+\varepsilon)$. *Then the following inequality holds for all declarations* a_i *and* (a_{-i}, v_{-i})*:*

$$\theta_i^{Opt} - \frac{\varepsilon}{1+\varepsilon} \cdot w(Opt(a, v \mid \neg i)) \le \theta_i^A \le \theta_i^{Opt} + \varepsilon \cdot w(Opt(a, v \mid \neg i)).$$

The idea of the proof is illustrated in Figure 3. Outside the depicted interval around θ_i^{Opt} agent i must be treated in algorithm A as in Opt in order to obtain the required approximation ratio. Using the above lemma we are able to derive a bound on the total payment made by our approximate mechanism, denoted by $p^A(S(A))$, in comparison to $p^{Opt}(S(Opt))$, i.e., the total payment of the VCG mechanism.

THEOREM 2. *Let* M_{Opt} *and* M_A *be the mechanisms defined above. Fix some arbitrary declaration vector* (a, v) *and let* $Opt = Opt(a, v)$ *and* $A = A(a, v)$ *denote the computed outputs. Then the following holds:*

$$\frac{1}{1+\varepsilon}\, p^{Opt}(S(Opt)) - \frac{\varepsilon}{1+\varepsilon}|S(Opt)|\, w(Opt) \le p^A(S(A)) \le (1+\varepsilon)\, p^{Opt}(S(Opt)) + \varepsilon\,(|S(A)|+1)\, w(Opt)\,.$$

We note that, since the number of selected agents is obviously bounded by the total number of agents n and $p^{Opt}(Opt) \ge w(Opt)$, we can derive something similar to an approximation ratio for the payments made by mechanism M_A.

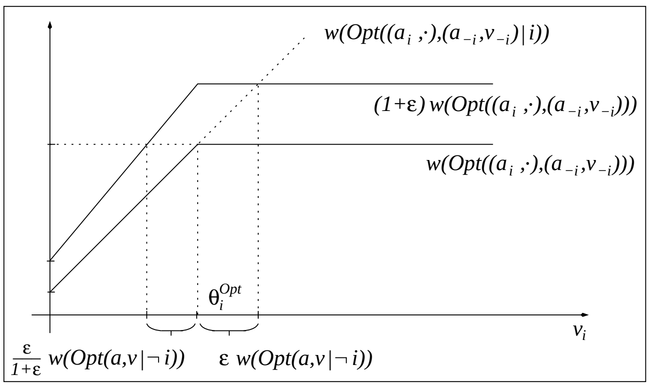

Figure 3: Range for the critical values of algorithm A.

```
Algorithm Greedy-1:
1  $\mathcal{T} := \emptyset$; $K := \{1, \ldots, k\}$;
2  forall $e \in E$ do $y_e := 1/b_e$;
3  repeat
4      forall $i \in K$ do $S_i :=$ argmin $\left\{ \sum_{e \in S} y_e \;\middle|\; S \in \mathcal{S}_i \right\}$;
5      $j :=$ argmax $\left\{ \frac{c_i}{d_i \sum_{e \in S_i} y_e} \;\middle|\; i \in K \right\}$;
6      $\mathcal{T} := \mathcal{T} \cup \{S_j\}$; $K := K \setminus \{j\}$;
7      forall $e \in S_j$ do $y_e := y_e \cdot \left(\mathrm{e}^{B-1} m\right)^{q_{S_j}(e)/(b_e - 1)}$;
8  until $\sum_{e \in E} b_e y_e \geq \mathrm{e}^{B-1} m$ or $K = \emptyset$;
9  return $\mathcal{T}$.
```

Figure 4: Truthful mechanism for network (multicast) routing. $\mathrm{e} \approx 2.718$ **is Euler number.**

COROLLARY 1. *Using the same notation as above for any truthful $(1+\varepsilon)$-approximation mechanism M_A it holds that*

$$1 - \frac{\varepsilon}{1+\varepsilon}(n+2) \leq \frac{p^A(S(A))}{p^{Opt}(S(Opt))} \leq 1 + \varepsilon(n+2).$$

Thus, given an FPTAS for any minimization problem from Section 2, it is also FPTAS with respect to the total VCG payment.

The second claim above follows since we can choose the difference in payment to be any polynomially small fraction of the VCG payments by choosing ε appropriately. Hence, if A is an FPTAS with respect to the underlying minimization problem, then A can also be viewed as an FPTAS with respect to total VCG payment if combined with the associated critical value payment scheme.

A modified version of these results applies to maximization. The difference is, however, that we cannot derive an approximation ratio since now payment is not guaranteed to be any polynomial fraction of the obtained social welfare. Hence, the last step of our above discussion will fail here and we can only bound the difference in payment by a polynomial fraction of social welfare.

4. A TRUTHFUL PTAS FOR THE GAP

We will show how to construct a monotone PTAS for the *generalized assignment problem* (GAP) with a constant number of bins using the technique of *partial enumeration*. It has been shown by Chekuri and Khanna [9] that no FPTAS is possible for this problem even if we do not require monotonicity. For the GAP we are given a set of n objects with associated values v_i and m bins that objects can be put into, each having w.l.o.g. capacity 1. A matrix $P = (p_{ij})$ describes the space $0 < p_{ij} \leq 1$ that object i occupies if put into bin j. We are interested in a feasible assignment $\varphi : [n] \rightarrow ([m] \cup \{0\})$, where $[n] = \{1, \ldots, n\}$, with $\sum_{i:\varphi(i)=j} p_{ij} \leq 1$ for $j \in [m]$, that maximizes $\sum_{i:\varphi(i)\neq 0} v_i$. Now let $\overline{\varphi}$ be a partial assignment of $k \leq n$ of the objects and assume that in bin j a capacity of β_j is left unused. We define algorithm $GAP_{\overline{\varphi}}$ based on this partial assignment. For all objects that are not yet assigned we round their sizes in bin j to multiplicities of β_j/n^2, i.e., $\overline{p_{ij}} = \lceil p_{ij} n^2/\beta_j \rceil \cdot (\beta_j/n^2)$. We then compute an optimal assignment of these remaining objects based on their rounded sizes by using a simple dynamic programming approach.

LEMMA 5. *Algorithm $GAP_{\overline{\varphi}}$ is monotone and bitonic with respect to social welfare.*

We let $w(GAP_{\overline{\varphi}}(I))$ refer to the objective function value obtained by $GAP_{\overline{\varphi}}$ on instance I and define

$$GAP_k(I) = \text{argmax}\{w(GAP_{\overline{\varphi}}(I)) \,|\, \overline{\varphi} \in \Phi_k\},$$

where Φ_k is the set of assignments of at most k objects.

THEOREM 3. *For any $k \in \mathbb{N}$, algorithm GAP_k has running time $O(k \cdot m^{k+1} \cdot n^{2m+k+1})$ and we have that $w(GAP_k(I)) \geq (1 - \frac{m}{k+1}) \cdot w(Opt(I))$. GAP_k defines a truthful PTAS for GAP with a constant number of bins.*

To see this, consider the optimal assignment *opt* for a given problem instance and let $\overline{opt}$ be the partial assignment of the k objects with largest value that are mapped to any bin by *opt*. Round all other objects' sizes as done by $GAP_{\overline{opt}}$. Observe, that in bin j the total error due to rounding is at most β_j/n, i.e., the total rounded size of objects in bin j is $\leq 1 + \beta_j/n$. If, however, the total rounded size of objects is larger than β_j, then there is an object with rounded size at least β_j/n in bin j. Removing this object from each bin gives a feasible assignment w.r.t. rounded sizes and defines a lower bound on the objective value obtained by $GAP_{\overline{opt}}$. Since each of the (at most m) removed objects decreases the objective value by at most $w(Opt(I))/(k+1)$ the approximation ratio follows. Our argument on the MAX–operator from Section 2 gives monotonicity of GAP_k, and exactness needed for Proposition 1 is obvious.

5. TRUTHFUL MECHANISMS FOR NETWORK (MULTICAST) ROUTING

In the unsplittable flow problem (UFP), we are given an undirected graph $G = (V, E)$, $|E| = m$, $|V| = n$, with edge capacities b_e, $e \in E$, and a set K of $k \geq 1$ commodities described by terminal pairs $(s_i, t_i) \in V \times V$ and a demand d_i and a value c_i. Like most of the previous work on UFP, we assume that $\max_i d_i \leq \min_e b_e$. W.l.o.g., let $d_i \in [0, 1]$ for each $i \in K = \{1, \ldots, k\}$, and $b_e \geq 1$ for all $e \in E$. Let $B = \min_e\{b_e\}$. This is a B-bounded UFP [5]. A feasible solution is a subset $K' \subseteq K$ and a single flow s_i-t_i-path for each commodity $i \in K'$ such that the capacities are not exceeded. The goal is to maximize the total value of the commodities in K'. A generalization is allocating bandwidth for multicast communication, where commodities are defined by sets of terminals that should be connected in form of a multicast tree.

Let $\mathcal{S}_i$ be the set of all s_i-t_i-paths in G, and $\mathcal{S} = \bigcup_{i=1}^{k} \mathcal{S}_i$. Given $S \in \mathcal{S}_i$, let $q_S(e) = d_i$ if $e \in S$, and $q_S(e) = 0$ otherwise. We will refer to a path $S \in \mathcal{S}$ also as a set; note that $S \subseteq E$. The UFP problem is modeled as the following integer linear program (ILP):

$$\begin{aligned}
\max \quad & \textstyle\sum_{i=1}^k c_i \cdot \left(\sum_{S\in\mathcal{S}_i} x_S\right) && && (1)\\
\text{s.t.} \quad & \textstyle\sum_{S:S\in\mathcal{S}, e\in S} q_S(e) x_S \le b_e && \forall e\in E && (2)\\
& \textstyle\sum_{S\in\mathcal{S}_i} x_S \le 1 && \forall i \in\{1,\dots,k\} && (3)\\
& x_S \in \{0,1\} && \forall S\in\mathcal{S}. && (4)
\end{aligned}$$

The linear programming (LP) relaxation is the same linear program with constraints (4) replaced with $x_S \ge 0$ for all $S\in\mathcal{S}$. The corresponding dual linear program is then:

$$\begin{aligned}
\min \quad & \textstyle\sum_{e\in E} b_e y_e + \sum_{i=1}^k z_i && && (5)\\
\text{s.t.} \quad & z_i + \textstyle\sum_{e\in S} q_S(e) y_e \ge c_i && \forall i\in\{1,\dots,k\}\,\forall S\in\mathcal{S}_i && (6)\\
& z_i, y_e \ge 0 && \forall i\in\{1,\dots,k\}\,\forall e\in E. && (7)
\end{aligned}$$

Based on these LPs, we specify in Figure 4 our primal-dual mechanism for routing, called Greedy-1. The proof of Theorem 4 below is in Section 5.1. Greedy-1 ensures feasibility by using y_e's: if an added set exceeded the capacity b_e of some $e\in E$, then this would imply the stopping condition already in the previous iteration.

THEOREM 4. *Greedy-1 outputs a feasible solution, and it is a $(\frac{e\gamma B}{B-1}(m)^{1/(B-1)})$-approximation algorithm if there is a polynomial time algorithm that finds a γ-approximate set S_i in line 4.*

In case of UFP we take $\gamma = 1$, as the shortest s_i-t_i-path computation finds set S_i in line 4 of Greedy-1. For multicast routing, this problem corresponds to the NP-hard Steiner tree problem, for which we can take $\gamma = 1.55$ [29]. As our algorithms are monotone in demands and valuations as required in Proposition 1 (see Lemma 9 and Section 5.1), they can be used as truthful mechanisms in the form of auctions for allocating network resources. The commodities correspond to bidders, the terminal nodes of bidders are known, but the bidders might lie about their demands and valuations. Our mechanisms can be used to sell bandwidth for unicast communication in networks. In the multicast routing the set of terminals for each bidder is known but the demands and valuations are unknown.

COROLLARY 2. *Given any $\epsilon > 0$, $B \ge 1+\epsilon$, Greedy-1 is a truthful $O(m^{1/(B-1)})$-approximation mechanism for UFP (unicast routing) as well as for the multicast routing problem, where the demands and valuations of the bidders are unknown.*

Awerbuch et al. [3] gave randomized online truthful mechanisms for uni- and multicast routing, obtaining an expected $O(\log(\mu m))$-approximation if $B = \Omega(\log m)$, where μ is the ratio of the largest to smallest valuation. Their approximation holds in fact w.r.t. the revenue of the auctioneer, but they assume known demands. Bartal et al. [6] give a truthful $O(B\cdot(m/\theta)^{1/(B-2)})$-approximation mechanism for UFP with unknown valuations and demands, where $\theta = \min_i\{d_i\}$. Our ratio does not depend on θ. A monotone version of an algorithm of Azar and Regev [5] is a truthful $O(B\cdot n^{\frac{1}{B-1}})$-approximation for UFP. Note, that our ratio is $O(1)$ when $B = \Theta(\log n)$, whereas the previous ratios were only $O(\log n)$.

If we do not insist on truthfulness, we are able to further improve the approximation ratios in Corollary 2. This requires a slight change in Greedy-1 and adequate changes in the previous analysis.

THEOREM 5. *There is a simple greedy primal-dual $O(m^{\frac{1}{\lfloor B+1\rfloor}})$-approximation algorithm for UFP (unicast routing), for the multicast routing, as well as for the column-restricted PIPs.*

Randomized rounding achieves a $O(d^{1/(B-1)})$-approximation for UFP [8], with d an upper bound on the length of the routing paths, e.g., $d = |V|$. LP-based rounding gives an $O(m^{\frac{1}{\lfloor B+1\rfloor}})$-approximation for column-restricted PIPs [19]. Theorem 5 gives a very simple combinatorial algorithm with comparable ratios.

5.1 Analysis

Approximation ratio, correctness. Our analysis is partly inspired by [12]. Given (not necessarily feasible) dual variables (y,z), and $S\in\mathcal{S}_i\subseteq\mathcal{S}$, let $f_{y,z}(S) = c_S/(z_i + \sum_{e\in S} q_S(e) y_e)$. Let $d_1(y) = \sum_{e\in E} b_e y_e$, $d_2(z) = \sum_{i=1}^k z_i$; $d(y,z) = d_1(y) + d_2(z)$.

We now specify, for the analysis, how are the dual variables handled. Let $(y^{\ell-1}, z^{\ell-1})$ be the dual variables and $p_{\ell-1}$ be the value of the primal solution at the beginning of ℓth iteration of Greedy-1. The initial values are $z_i^0 = 0$ for each $i\in K$, $y_e^0 = 1/b_e$ for each $e\in E$, and the primal variables are set to $x_S = 0$ for each $S\in\mathcal{S}$.

Let $S_j \in \mathcal{S}_j$ be the set found in line 5 in ℓth iteration of Greedy-1. Line 6 updates primal variables: $x_{S_j} := x_{S_j} + 1$, $p_\ell := p_{\ell-1} + c_j$; z-variables are modified in line 6: $z_j^\ell := z_j^{\ell-1} + c_j$. Each commodity is considered at most once, thus each $z_i^{\ell-1}\in\{0, c_i\}$ in any iteration ℓ. y's are updated in line 7: $y_e^\ell := y_e^{\ell-1}\cdot(e^{B-1}m)^{q_{S_j}(e)/(b_e-1)}$ for each $e\in S_j$; $q_{S_j}(e) = d_j$ if $e\in S_j$.

Suppose we are still at iteration ℓ and set S_j was selected. Let $(y,z) = (y^{\ell-1}, z^{\ell-1})$ and define

$$f(y,z) = \max\{f_{y,z}(S) : S\in\mathcal{S} \text{ and } \exists j' \text{ s.t. } S\in\mathcal{S}_{j'}, z_{j'} = 0\}.$$

If we denote $f'(y,z) = f_{y,z}(S_j)$, then since each S_i found in line 4 was a γ-approximation, we have $1/f(y,z) \le 1/f'(y,z) \le \gamma/f(y,z)$. Then, the following is a straightforward observation.

LEMMA 6. *Let (y,z) be the current dual variables at iteration ℓ of Greedy-1. Then vector $(\gamma\cdot f'(y,z)\cdot y, z)$ is a feasible fractional solution to the dual linear program.*

For simplicity, $d_1(y^\ell)$, $d_2(z^\ell)$, $f(y^\ell, z^\ell)$ and $f'(y^\ell, z^\ell)$ are denoted by $d_1(\ell)$, $d_2(\ell)$, $f(\ell)$ and $f'(\ell)$, respectively. The algorithm stops at the first iteration t s.t. $d_1(t) \ge e^{B-1}m$. If Greedy-1 stops and $\sum_{e\in E} b_e y_e \ge e^{B-1}m$ is false, then this means that Greedy-1 has chosen at leat one set from each commodity, and thus the solution found is optimal. From now on, we analyze the approximation of Greedy-1, when it stops with condition $\sum_{e\in E} b_e y_e \ge e^{B-1}m$.

LEMMA 7. *Let d be the value of an optimal fractional solution to the dual LP, and $\mathcal{T}$ be the solution output by Greedy-1, then*

$$\frac{d}{\sum_{S\in\mathcal{T}} c_S} \le e\gamma\frac{B}{B-1}(m)^{1/(B-1)}.$$

PROOF. For any iteration $\ell\ge 1$ of Greedy-1, let $S_\ell\in\mathcal{S}_{i_\ell}$ be the set selected in ℓth iteration, i.e., set $S_j\in\mathcal{S}_j$ from line 5, $i_\ell = j$.

$$\text{Let} \quad \Delta_e = (b_e - 1)\left(\left(e^{B-1}m\right)^{1/(b_e-1)} - 1\right),$$

$$\text{and} \quad \Delta = (B-1)\left(\left(e^{B-1}m\right)^{1/(B-1)} - 1\right).$$

For any iteration $\ell \ge 1$ we have that

$$d_1(\ell) = \sum_{e\in E} b_e y_e^\ell = \sum_{e\in E} b_e y_e^{\ell-1}\left(1 + \frac{\Delta_e}{b_e-1}\right)^{q_{S_\ell}(e)}$$

$$\le \sum_{e\in E} b_e y_e^{\ell-1}\left(1 + q_{S_\ell}(e)\frac{\Delta_e}{b_e-1}\right),$$

where the last inequality follows from the fact that function $f(x) = (1+a)^x$ is convex, $f(0) = 1$, $f(1) = 1+a$, and so $f(x)\le 1+ax$ for any $x\in[0,1]$; recall, that by our assumption, $q_{S_\ell}(e)\in[0,1]$.

We observe, that since $B \geq 1$ and $m \geq 1$, function $g(x) = x\left(\left(\mathrm{e}^{B-1}m\right)^{1/x} - 1\right)$ is non-increasing when $x \geq 0$; note, that $\lim_{x\to 0+} g(x) = +\infty$. This follows by simple calculus, omitted here. Thus, we have $\Delta_e \leq \Delta$, for each e. We can further write

$$d_1(\ell) \leq \sum_{e\in E} b_e y_e^{\ell-1}\left(1 + \frac{q_{S_\ell}(e)\Delta}{b_e - 1}\right) = \sum_{e\in E} b_e y_e^{\ell-1} +$$

$$\Delta \sum_{e\in S_\ell} \frac{b_e}{b_e - 1} q_{S_\ell}(e) y_e^{\ell-1} \leq \sum_{e\in E} b_e y_e^{\ell-1} + \frac{\Delta B}{B-1}\sum_{e\in S_\ell} q_{S_\ell}(e) y_e^{\ell-1}$$

$$= d_1(\ell-1) + \frac{\Delta B c_{S_\ell}}{(B-1)f'(\ell-1)} = d_1(\ell-1) + \frac{\Delta B}{B-1}\frac{p_\ell - p_{\ell-1}}{f'(\ell-1)},$$

thus

$$d_1(\ell) \leq d_1(0) + \frac{\Delta B}{B-1}\sum_{j=1}^{\ell}\frac{p_j - p_{j-1}}{f'(j-1)}.$$

By Lemma 6, $(\gamma \cdot f'(j-1)\cdot y^{j-1}, z^{j-1})$ is a feasible fractional dual solution. We can therefore write that $d \leq \gamma \cdot f'(j-1)\cdot d_1(j-1) + d_2(j-1)$ which then gives $f'(j-1) \geq \frac{d - d_2(j-1)}{\gamma d_1(j-1)}$.

If there exists an iteration, say j, such that $d_2(j-1) \geq \frac{d}{c}$, where $c = \frac{\gamma B}{B-1}(\mathrm{e}(m)^{1/(B-1)} - 1) + 1$, then, since $B \geq 1$, $\gamma \geq 1$, $c \leq \frac{\mathrm{e}\gamma B}{B-1}(m)^{1/(B-1)}$, we already have a $\frac{\mathrm{e}\gamma B}{B-1}(m)^{1/(B-1)}$-approximate solution at that iteration. (Note, that $\frac{\mathrm{e}\gamma B}{B-1}(m)^{1/(B-1)}$ is the final ratio we want to show in Theorem 4.) This can be seen as follows: $d_2(j-1)$ corresponds to the sum of the values of the sets selected from each commodity that has been inspected in the first $j-1$ iterations. Otherwise, $d_2(j-1) < \frac{d}{c}$ for all iterations of the algorithm, and we have $1/f'(j-1) \leq \frac{\gamma d_1(j-1)}{d - d_2(j-1)} \leq \frac{c}{c-1}\frac{\gamma d_1(j-1)}{d}$. We can now continue upper-bounding the value of $d_1(\ell)$:

$$\begin{aligned} d_1(\ell) &\leq d_1(0) + \frac{\Delta B}{B-1}\sum_{j=1}^{\ell}\frac{p_j - p_{j-1}}{f'(j-1)} \\ &\leq d_1(0) + \frac{\rho}{d}\sum_{j=1}^{\ell}(p_j - p_{j-1})d_1(j-1), \end{aligned}$$

where $\rho = \frac{c\gamma\Delta B}{(c-1)(B-1)}$. Let now $h(\cdot)$ be defined by the following recurrence: $h(0) = d_1(0)$, and

$$h(\ell) = h(0) + \frac{\rho}{d}\sum_{j=1}^{\ell}(p_j - p_{j-1})h(j-1).$$

Then we can easily observe, that $d_1(\ell) \leq h(\ell)$ for all ℓ, and

$$h(\ell) = h(\ell-1) + \frac{\rho}{d}(p_\ell - p_{\ell-1})h(\ell-1) =$$

$$h(\ell-1)\left(1 + \frac{\rho}{d}(p_\ell - p_{\ell-1})\right) \leq h(\ell-1)\cdot \mathrm{e}^{\frac{\rho}{d}(p_\ell - p_{\ell-1})},$$

where we used the fact that $1 + x \leq \mathrm{e}^x$ for $x \geq 0$. By using $h(0) = m$, the last recurrence implies:

$$d_1(\ell) \leq h(\ell) \leq m\mathrm{e}^{\rho p_\ell/d},$$

and since $d_1(t) \geq \mathrm{e}^{B-1}m$, we obtain finally $\mathrm{e}^{B-1}m \leq m\mathrm{e}^{\rho p_t/d}$, and so $\frac{d}{p_t} \leq \frac{\rho}{B-1} = \frac{c\gamma\Delta B}{(c-1)(B-1)^2} = c = \frac{\gamma B}{B-1}(\mathrm{e}(m)^{1/(B-1)} - 1) + 1 \leq \frac{\mathrm{e}\gamma B}{B-1}(m)^{1/(B-1)}$. $\square$

LEMMA 8. *The solution $\mathcal{T}$ output by Greedy-1 is feasible.*

PROOF. Let us consider an execution of Greedy-1, and suppose, that the current solution is feasible so far. Let S be the first set that violates the feasibility when added to the current solution in iteration, say, ℓ. This means there is an element $e \in S$ such that if $\mathcal{E}$ is a family of all sets that contain e and have been added to the solution before set S, then $\sum_{S'\in\mathcal{E}} q_{S'}(e) \leq b_e$ and $q_S(e) + \sum_{S'\in\mathcal{E}} q_{S'}(e) > b_e$. Since each $q_S(e) \in [0,1]$, this means that $\sum_{S'\in\mathcal{E}} q_{S'}(e) > b_e - 1$. Now, we can lower bound the product of b_e and the dual variable for e in the previous iteration $\ell - 1$ by:

$$b_e \cdot y_e^{\ell-1} = b_e \cdot y_e^0 \cdot \left(1 + \frac{\Delta_e}{b_e - 1}\right)^{\sum_{S'\in\mathcal{E}} q_{S'}(e)} =$$

$$\left(1 + \frac{\Delta_e}{b_e - 1}\right)^{\sum_{S'\in\mathcal{E}} q_{S'}(e)} > \left(1 + \frac{\Delta_e}{b_e - 1}\right)^{b_e - 1} = \mathrm{e}^{B-1}m.$$

By the stopping condition of Greedy-1, this implies that the previous iteration $\ell - 1$ was in fact the last iteration of the algorithm, and so set S was not added to the solution. $\square$

PROOF. (of Theorem 4) Lemma 8 implies feasibility of the output solution. We now argue about the approximation ratio of the overall algorithm. By Lemma 7, and using the weak LP duality, the approximation ratio of Greedy-1 is at most: $\frac{\mathrm{e}\gamma B}{B-1}(m)^{1/(B-1)}$. The number of iterations of our algorithm is at most the number of commodities, since each commodity is considered at most once. This finishes the proof of Theorem 4. $\square$

Truthfulness. We assume that commodities correspond to bidders, that the terminal nodes of bidders are known, but the bidders might lie about their demands and valuations. More formally, given bidder $i \in K$, her type is $(a_i, v_i) = (d_i, c_i)$, and the terminals s_i, t_i are known to the mechanism. Relation $\preceq$ is defined just as $\leq$. Similarly, in the multicast routing we assume that the set of terminals for each bidder is known but the demands and valuations are unknown. We prove below that Greedy-1 is monotone and exact as desired in Proposition 1.

LEMMA 9. *Algorithm Greedy-1 is monotone and exact with respect to any bidder's i type (d_i, c_i), even if a γ-approximation algorithm is used in line 4.*

PROOF. Let us fix a bidder i, and let the bids of the other bidders be fixed. Suppose i bids (d_i, c_i) and was selected to the output solution, and this happened in iteration $t \in \mathbb{N}$. Let now c_i increase to some $c_i' \geq c_i$, and let the demand d_i decrease to $d_i' \leq d_i$, as needed for the monotonicity. If commodity i has been allocated in some iteration $t' < t$, then we are done. Otherwise, we will argue that i will be allocated in iteration t as before. Since i has not been selected up to iteration t, the same partial solution is computed when i's type is (d_i', c_i') instead of type (d_i, c_i); same is true for the γ-approximation algorithm in line 4 in iteration t. Now, we know that for type (d_i, c_i) bidder i maximized the quantity $\frac{c_j}{d_j\sum_{e\in S_j} y_e}$ in iteration t, where S_j was the γ-approximate solution found. Thus, bidder i must also maximize $\frac{c_j}{d_j\sum_{e\in S_j} y_e}$ when having type (d_i', c_i'), and so i will still be allocated.

Exactness of Greedy-1 is easy to see, i.e., it just corresponds to routing exactly the demand of d_i along a chosen s_i-t_i-path. $\square$

6. WINNER DETERMINATION IN CA'S WITH GENERAL BIDDERS

We will show in this section how to extend our algorithm and analysis from Section 5 to combinatorial auctions (CA) with general bidders. A crucial issue in combinatorial auctions is how the types (bids) of the agents (bidders) are represented, i.e., the bidding language. We will allow each bidder to specify her bids through an *oracle*, see [6] for a similar approach. Suppose, we have k bidders $K = \{1, \ldots, k\}$. Let U, $|U| = m$, be the set of all kinds of goods for sale, s.t. each $e \in U$ is available in $b_e \in \mathbb{N}$ units; $B = \min_e\{b_e\}$. Let $c_S > 0$ be the valuation of bidder $i \in K$ for $S \in \mathcal{S}_i \subseteq 2^U$, where $\mathcal{S}_i$ are all sets i may demand; $|\mathcal{S}_i|$ may be exponential. Given $\gamma \geq 1$, "prices" $y_e > 0$, for each $e \in U$, and a bidding threshold $z_i \geq 0$, a *γ-oracle* of bidder i either decides that $c_S \leq z_i$ for all $S \in \mathcal{S}_i$ (i.e., i cannot buy anything), or otherwise the oracle outputs $S' \in \mathcal{S}_i$ such that

$$\frac{\sum_{e\in S'} y_e}{c_{S'} - z_i} \leq \gamma \cdot \frac{\sum_{e\in S} y_e}{c_S - z_i}, \quad \forall S \in \mathcal{S}_i \text{ with } c_S > z_i.$$

γ-oracle(i, y, z_i) denotes the output of our γ-oracle for bidder i, where $y = (y_e : e \in U)$. There are, e.g., simple polynomial time 1-oracles for all bidding languages used by Bartal et al. [6]. For example, we have used the *shortest path* 1-oracle for UFP, and the *multicast tree* 1.55-oracle in Section 5.

The winner determination problem in CAs (WDCA) is defined as the following integer linear program (ILP):
$\max\{\sum_{i=1}^{k} \sum_{S\in\mathcal{S}_i} c_S \cdot x_S \,|\, \sum_{S:S\in\mathcal{S},e\in S} x_S \leq b_e \ \forall e \in U,$
$\sum_{S\in\mathcal{S}_i} x_S \leq 1 \ \forall i \in K, \ x_S \in \{0,1\} \ \forall S\}$.

Consider the XOR-bids language [25]. Each bidder can submit an arbitrary number of pairs (bids) (S, c_S), where $S \subseteq U$, and c_S is the valuation. Implicit here is that the bidder wants to receive at most one of these bids. If each bidder, e.g., submits a list of XOR-bids, then there is a trivial linear time 1-oracle. We use the fact that any complicated set of preferences given through a γ-oracle is equivalent to a (possibly exponential) number of XOR-bids, see Nisan [25]. This allows us to simulate the γ-oracle in an ILP where XOR-bids are represented as constraints $\sum_{S\in\mathcal{S}_i} x_S \leq 1$. Our primal-dual greedy algorithm for WDCA, called Greedy-2, extends Greedy-1. It uses our full primal-dual machinery and maintains explicitly the y and z-dual variables. The main difficulty here comes from the fact that one commodity (bidder) can have now unbounded ratio of the largest to smallest valuation on it's sets. This makes troubles in defining the z-variables that would make constraint (6) feasible. To deal with this problem we explicitly put z's into the γ-oracle and take a sequence of sets from one commodity whose values form an arithmetic progression with factor $1 + 2\gamma$ and repair this infeasibility afterwards – see Fig. 5. Thus, Greedy-2 may consider each commodity more than once, but it nevertheless has time polynomial in m, k, and $\log_{(1+2\gamma)}(c_{\max}/c_{\min})$ due to handling the z-variables in lines 5,6, and 8. Analysis in Section 5 can be extended to give the following.

THEOREM 6. *Suppose there is a polynomial time γ-oracle for each bidder. Then Greedy-2 is a simple primal-dual* $\mathrm{e} \cdot (2\gamma + 1) \cdot (m)^{1/(B+1)}$*-approximation algorithm for the winner determination problem in multi-unit combinatorial auctions. This implies an $O(m^{1/(B+1)})$-approximation for $(0, 1)$-PIPs.*

We would like to point out that our algorithm from Theorem 6 is a unified combinatorial greedy algorithm for many specific packing problems for which specialized combinatorial greedy algorithms were known – see, e.g., [14, 18].

The problem of approximating WDCA has been widely studied. Though, it has not been used before, we could use randomized rounding to obtain an $O(\gamma m^{1/(B+1)})$-approximation to WDCA. This, however, requires solving an LP relaxation of the ILP, where some constraints of the dual LP are represented by γ-oracles. This could be done by using the Ellipsoid algorithm. Theorem 6 implies the same approximation by a simple combinatorial greedy algorithm. To our knowledge, the best known combinatorial algorithm for WDCA follows by the work of Awerbuch et al. [4] and Bartal et al. [6], and gives an $O(B \cdot (m)^{1/(B-2)})$-approximation. Theorem 6 gives an $O(m^{1/(B+1)})$-approximation to (0,1)-PIPs. Chekuri and Khanna [10] show that the best possible ratio for (0,1)-PIPs is $\Omega(m^{\frac{1}{B+1}})$, unless NP= ZPP. Previously, this bound for (0,1)-PIPs was only matched by using LP-based randomized rounding [27, 31].

```
Algorithm Greedy-2:
1   $\mathcal{T} := \emptyset$; $K := \{1, \ldots, k\}$; $S' := \emptyset$;
2   forall $e \in U$ do $y_e := 1/b_e$; forall $i \in K$ do $z_i := 0$;
3   repeat
4       forall $i \in K$ do $S_i := \gamma$-oracle$(i, y, z_i)$;
5       $j :=$ argmax $\left\{ \frac{c_{S_i} - z_i}{\sum_{e\in S_i} y_e} \,\middle|\, i \in K \right\}$;
6       if $c_{S_j} > z_j$ then
7           $\mathcal{T} := \mathcal{T} \cup \{S_j\}$; $S' := S_j$;
8           $z_j := (1 + 2\gamma) \cdot c_{S_j}$;
9           forall $e \in S_j$ do $y_e := y_e \cdot \left(\mathrm{e}^{B+1} m\right)^{1/(b_e+1)}$;
10  until $\left(\sum_{e\in U} b_e y_e \geq \mathrm{e}^{B+1} m\right)$ or
                                        $(\forall i \in K, S \in \mathcal{S}_i : c_S \leq z_i)$;
11  forall $i \in K$ do
        skip all, but the most valuable, sets from $\mathcal{T} \cap \mathcal{S}_i$;
12  if $\sum_{T\in\mathcal{T}\setminus\{S'\}} c_T \geq c_{S'}$ then
                        return $\mathcal{T} \setminus \{S'\}$ else return $\{S'\}$.
```

Figure 5: Winner determination in multi-unit CAs.

The algorithm from Theorem 6 can be extended to winner determination in multi-unit combinatorial auctions with multisets, allowing agents to bid on more than one unit of each good, i.e., a set is now $S = \{q_S(e) : e \in U\}$, where $q_S(e) \in [0, 1]$. We obtain an approximation ratio of $\mathrm{e}(2\gamma + 1)(m)^{1/B}$ in this case. This also implies a $O(m^{1/B})$-approximation for PIPs, which was achieved before only by LP-based randomized rounding [27, 31].

THEOREM 7. *If each bidder has a poly-time γ-oracle, then there is a simple primal-dual greedy* $\mathrm{e} \cdot (2\gamma + 1) \cdot (m)^{1/B}$*-approximation for WDCA with multi-sets. This implies an $O(m^{1/B})$-approximation for PIPs.*

7. TRUTHFUL SINGLE-MINDED CA'S

We show here that our algorithms from Section 6 imply truthful mechanisms for multi-unit combinatorial auctions among unknown single-mined bidders. From an algorithmic point of view, multi-unit CAs essentially correspond to NP-hard PIPs. The most general tool for approximating PIPs – randomized rounding as introduced by Raghavan and Thompson [27, 28], see also [31], does not give a truthful mechanism. To circumvent this problem, Archer et al. [1] introduce additional dropping probabilities for bidders, which makes their algorithm *monotone in the bids*. This approach yields a truthful mechanism but only in a probabilistic sense and only for "known bidders". We use our greedy algorithms for CAs represented in form of PIPs to obtain truthful mechanisms with approximation factors close to those of randomized rounding.

We model multi-unit CAs among single-minded bidders as a special case of ILP (1)-(4), where $E = U$. A bid of bidder $i \in K$ is $(a_i, v_i) = (S, c_S)$, $S \in \mathcal{S}_i$, and c_S is the valuation. Assume that

```
Algorithm Greedy-3:
1   $\mathcal{T} := \emptyset$;
2   forall $e \in U$ do $y_e := 1/b_e$;
3   repeat
4       $S := \text{argmax}\left\{ \frac{c_S}{\sum_{e\in U} q_S(e) y_e} \;\middle|\; S \in \mathcal{S} \setminus \mathcal{T} \right\}$;
5       $\mathcal{T} := \mathcal{T} \cup \{S\}$;
6       forall $e \in S$ do $y_e := y_e \cdot (\mathrm{e}^B m)^{q_S(e)/b_e}$;
7   until $\sum_{e\in U} b_e y_e \geq \mathrm{e}^B m$;
8   return $\mathcal{T}$.
```

Figure 6: Truthful mechanism for multi-unit CAs among unknown single-minded bidders. For CAs without multisets: $q_S(e) \in \{0,1\}$ for each $e \in U$, $S \in \mathcal{S}$.

$|\mathcal{S}_i| = 1$ for each bidder $i \in K$, and $\mathcal{S} = \bigcup_i \mathcal{S}_i$. The relation $R_\preceq$ is defined as $S \preceq S'$ iff $S \subseteq S'$. The algorithm is now simpler than Greedy-2, see Fig. 6. We can use our analysis from Section 6 to show that Greedy-3 is a $O(m^{1/B})$-approximation to the defined ILP, which is a (0,1)-PIP. Greedy-3 is exact and monotone for CAs with unknown single-minded bidders, as needed in Proposition 1 (see the proof of Lemma 9).

THEOREM 8. *Algorithm Greedy-3 is a truthful $O(m^{\frac{1}{B}})$-approximation mechanism for multi-unit combinatorial auctions among unknown single-minded bidders.*

Let us remark that the values of the dual variables y_e cannot be used for the prices offered to the bidders. However, prices following the critical-value payment can be calculated by k calls to Greedy-3, one for each bidder. Within a call not taking into account the bid of bidder i we compute in each iteration of Greedy-3 the minimum bid that would have resulted in bidder i being selected in line 4. Clearly, bidder i's critical value is just the minimum among these values. The best known truthful mechanism for this problem among unknown single-minded bidders is $O(B \cdot m^{1/(B-2)})$-approximate [6]. (It works in fact for more general bidders.)

Our result can be generalized towards CAs with unknown single-minded bidders with multi-sets. We will model a bid of bidder i as a pair (S, c_S), such that $S = \{q_S(e) : e \in U\}$, where $q_S(e) \in [0,1]$ is a fraction, i.e., the number of units after scaling, of good e the bidder requires. We model this problem as an ILP similar to ILP (1)-(4), assuming $|\mathcal{S}_i| = 1$ for each $i \in K$. We now use algorithm Greedy-3 with: line 6 changed into **forall** $e \in U$ **do** $y_e := y_e \cdot (\mathrm{e}^{B-1} m)^{q_S(e)/(b_e-1)}$, and line 7 changed into **until** $\sum_{e\in U} b_e y_e \geq \mathrm{e}^{B-1} m$. Our previous analysis can be used to obtain the following.

THEOREM 9. *Algorithm Greedy-3 modified as above is a truthful $O(m^{\frac{1}{B-1}})$-approximation mechanism for multi-unit combinatorial auctions with unknown single-minded bidders and multisets.*

The best previous truthful mechanism for this problem is due to Bartal et al. [6], and achieves an $O(B \cdot (m/\theta)^{1/(B-2)})$-approximation, where $\theta = \min\{q_S(e) : e \in U, S \in \mathcal{S},\, q_S(e) > 0\}$. Our ratio is independent from θ.

8. REFERENCES

[1] A. Archer, C.H. Papadimitriou, K. Talwar, and É. Tardos. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *14th ACM-SIAM SODA*, 2003.

[2] A. Archer and É. Tardos. Truthful mechanisms for one-parameter agents. In *IEEE FOCS*, pp. 482–491, 2001.

[3] B. Awerbuch, Y. Azar, A. Meyerson. Reducing truth-telling online mechanisms to online optimization. In *ACM STOC*, 503–510, 2003.

[4] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput-competitive online routing. In *Proc. 34th IEEE FOCS*, pages 32–40, 1993.

[5] Y. Azar and O. Regev. Strongly polynomial algorithms for the unsplittable flow problem. In *8th IPCO*, LNCS, pp. 15-29, 2001.

[6] Y. Bartal, R. Gonen, and N. Nisan. Incentive Compatible Multi-Unit Combinatorial Auctions. In the *Proc. 9th conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, USA, June, 2003.

[7] G. Calinescu. Bounding the payment of approximate truthful mechanisms. In *ISAAC*, 2004.

[8] A. Chakrabarti, C. Chekuri, A. Kumar and A. Gupta. Approximation Algorithms for the Unsplittable Flow Problem. In *APPROX*, 2002.

[9] C. Chekuri and S. Khanna. A PTAS for the Multiple Knapsack Problem. In *ACM-SIAM SODA*, 2000.

[10] C. Chekuri and S. Khanna. On Multidimensional Packing Problems. In *10th ACM-SIAM SODA*, pp. 185–194, 1999.

[11] E. Clarke. Multipart pricing of public goods. *Public Choice*, **8**, pp. 17–33, 1971.

[12] N. Garg and J. Könemann. Faster and Simpler Algorithms for Multicommodity Flow and Other Fractional Packing Problems. In *IEEE FOCS*, 1998.

[13] T. Groves. Incentives in teams. *Econometrica*, **41(4)**, 617–631, 1973.

[14] M.M. Halldórsson. A survey on independent set approximations. In *Proc. APPROX '98*, Springer LNCS 1444, pp. 1–14, 1998.

[15] R. Hassin. Approximation Schemes for the Restricted Shortest Path Problem. *Math. Oper. Res.*, **17(1)**, pp. 36–42, 1992.

[16] K. Jain and V. Vazirani. Applications of Approximation Algorithms to Cooperative Games. In *ACM STOC*, 2001.

[17] K. Jain and V. Vazirani. Equitable Cost Allocation via Primal-Dual Type Algorithms. In *ACM STOC*, 2002.

[18] J. Kleinberg. *Approximation algorithms for disjoint paths problems.* PhD thesis, MIT, 1996.

[19] S.G. Kolliopoulos and C. Stein. Approximating disjoint-path problems using greedy algorithms and packing integer programs. In *6th IPCO*, Springer LNCS, **1412**, 1998.

[20] A. Kothari, D. Parkes, and S. Suri. Approximately–strategyproof and tractable multi–unit auctions. *ACM Conference on Electronic Commerce (EC)*, 2003.

[21] D. Lehmann, L. O'Callaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *ACM Conference on Electronic Commerce (EC)*, 1999.

[22] M.V. Marathe, R. Ravi, R. Sundaram, S.S. Ravi, D.J. Rosenkrantz, and H.B. Hunt. Bicriteria network design problems. *J. of Algorithms*, **28**(1), 142–171, 1998.

[23] A. Mu'alem and N. Nisan. Truthful Approximation Mechanisms for Restricted Combinatorial Auctions. In *18th Nat'l AAAI Conf.*, 2002.

[24] N. Nisan and A. Ronen. Algorithmic Mechanism Design. In *ACM STOC*, pp. 129–140, 1999.

[25] N. Nisan. Bidding and Allocation in Combinatorial Auctions. In *2nd ACM Conference on Electronic Commerce (EC)*, 2000.

[26] C. A. Phillips. The Network Inhibition Problem. In *ACM STOC*, 1993.

[27] P. Raghavan. Probabilistic Construction of Deterministic Algorithms: Approximating Packing Integer Programs. *J. Comput. Syst. Sci.*, **37(2)**, 130-143, 1988.

[28] P. Raghavan and C.D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, **7**: 365–374, 1987.

[29] G. Robins and A. Zelikovsky. Improved Steiner Tree Approximation in Graphs. In *ACM-SIAM SODA*, 770–779, 2000.

[30] S. Sahni. Algorithms for Scheduling Independent Tasks. *Journal of the ACM*, **23(1)**, pp. 116–127, 1976.

[31] A. Srinivasan. Improved Approximation Guarantees for Packing and Covering Integer Programs, *SIAM J. Computing*, **29**, 648–670, 1999.

[32] K. Talwar. The Price of Truth: Frugality in Truthful Mechanisms. In *STACS*, 2003.

[33] W. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *J. Finance*, **16**, pp. 8–37, 1961.