TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik

Lehrstuhl für Informatik XVIII

Algorithmic Mechanism Design via Relaxation and Rounding

Salman Fadaei

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:	Prof. Dr. Alexander Pretschner
Prüfer der Dissertation:	1. Prof. Dr. Martin Bichler 2. Prof. Dr. Susanne Albers

Die Dissertation wurde am 19.05.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 05.08.2016 angenommen.

Algorithmic Mechanism Design via Relaxation and Rounding

Salman Fadaei

To Reza, Mohammadreza and Mahin

To Mandana and Siavash

ABSTRACT

The problem of maximizing social welfare for environments with distributed decision makers who are only concerned with selfish objectives, is hard but of central importance. The problem appears in several scenarios from combinatorial auctions to matching markets. The hardness of the problem partly arises from the inherent tension between social goals and the individual objectives. From computational complexity point of view, maximizing social welfare is NP-hard, and this adds to the hardness. The field of algorithmic mechanism design addresses these issues via welfare approximation, or welfare approximation supplemented with payment schemes. This thesis analyzes two problems in algorithmic mechanism design, and introduces efficient machineries that are applicable in algorithmic mechanism design.

First, we study a strategic variant of the generalized assignment problem (GAP). In GAP, a set of items has to be optimally assigned to a set of bins without exceeding the capacity of any singular bin. In the strategic variant of the problem we study, values for assigning items to bins are the private information of bidders, and the mechanism should provide bidders with incentives to truthfully report their values. The presented truthful algorithm improves upon the existing optimization algorithms for GAP in terms of simplicity and runtime while the approximation ratio closely matches the best approximation ratio given for GAP without strategic considerations.

Second, we study a problem of truthful mechanism design for a strategic variant of GAP in a payment-free environment. In this strategic variant, bins are held by strategic agents, and each agent may hide its compatibility with some items in order to obtain items of higher values. The compatibility between an agent and an item encodes the willingness of the agent to receive the item. The goal is to maximize social welfare while certifying no agent can benefit from hiding its compatibility with items. The model has applications in auctions with budgeted bidders. In contrast to the mechanism design with quasi-linear valuations, in this setting, payments are unavailable and the well-known Vickrey-Clarke-Groves (VCG) mechanism is inapplicable, thus we resort to the technique of welfare approximation.

Third, we study the meta-randomized rounding technique which decomposes a fractional solution of a relaxed linear program into a convex combination of integer solutions. This rounding technique is broadly used in combinatorial optimization, and mechanism design with or without money. However, the technique relies heavily on the ellipsoid method, which is notorious for its poor practical per-

formance. In this thesis, we present alternative decomposition techniques that are more practical. The first presented technique is a geometric-based approach that, given a fractional point, finds a convex decomposition which can get arbitrarily close to the fractional point. The second technique is based on the Dantzig-Wolfe decomposition. It finds a fractional solution to the relaxed linear program, and at the same an exact convex decomposition of the fractional solution.

ACKNOWLEDGEMENTS

I owe so much to many who helped me towards completion of this thesis. I am thankful to Martin my supervisor for his trust and support, and for the freedom he gave me in research. I thank Susanne Albers for her encouraging words in our meetings. Dennis thank you for working together with me. Per P., Zhen H., Andreas W., Florian S., and Usiel R. thank you for constructive discussions. I thank my parents for their incalculable support and patience during the whole time of my education. Thanks to my wife Mandana for her support and endurance. Last but not least, I thank my three-years-old son Siavash for bringing new meaning to our lives.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

A set of numbers $a_1, a_2, \ldots, a_n$ is given. The algorithm below finds the maximum among these numbers.

```
max := a_1;
for i ← 2 to n do
    if max < a_i then
        max := a_i;
```

Now, consider a setting in which these numbers belong to self-interested agents. Self-interested agents may misreport the numbers because of their own interests. In this situation, the algorithm above will be useful only if we certify that the true numbers are reported by the agents. To this end, we need to model and understand the interests of the agents.

The dominant approach to model an agent's interests or *preferences* is the utility theory. A *utility function* quantifies an agent's degree of preferences across a set of alternatives or outcomes. A utility function assigns a real number to an outcome which defines the level of happiness of an agent with the outcome. Working with the idea of utility functions, in lieu of preferences, is pervasive and without loss of generality according to the von Neumann–Morgenstern theorem. Given a set of outcomes, the preference relationship of an agent identifies an ordering over the outcomes or lotteries of outcomes. A lottery is a probability distribution over the outcomes. The von Neumann–Morgenstern theorem states that for any preference relationship that satisfies a set of reasonable properties, there exists a utility function that correctly represents the preference relationship (Shoham and Leyton-Brown, 2008). The von Neumann–Morgenstern theorem thus justifies the pervasive claim that a single-dimensional function (a utility function) suffices to describe preferences over an arbitrarily complicated set of alternatives. We benefit from this observation and work with utility functions.

Finding the maximum among a set of numbers is an algorithmic problem. If we assume each number is the *private* information of an agent, then we also want the agents to disclose their numbers truthfully to the algorithm above. Otherwise, the algorithm will be of no use. Hence, we face a problem of *mechanism design* rather than

just algorithm design. Mechanism design or *implementation theory* is sometimes called "inverse game theory". Thus, to better understand mechanism design, we start with some game-theoretic fundamentals.

1.1 GAME THEORY

Game theory is the mathematical study of interaction among independent self-interested agents or players of a game. Normal-form games are the most fundamental representation of strategic settings. In normal-form games, agents' moves are simultaneous. We briefly introduce normal-form games which will help us define useful concepts such as equilibria.

Definition 1 (Normal-form Games)**.** *A (finite, n-person) normal-form game is a tuple* $(\mathcal{I}, A, u)$*, where:*

- $\mathcal{I}$ *is a finite set of n players, indexed by i*
- $A = A_1 \times \ldots \times A_n$ *, where* A_i *is a finite set of actions available to player i. Each vector* $a = (a_1, \ldots, a_n) \in A$ *is called an action profile.*
- $u = (u_1, ..., u_n)$ *where* $u_i : A \mapsto \mathbb{R}$ *is a real-valued utility (or payoff) function for player i.*

Agents choose strategies to play in a game. A strategy can be to select an action and play it which is called a *pure strategy*. Agents can also randomize over the set of available actions according to some probability distribution. Such a strategy is called a *mixed strategy*. The set of all strategies for agent i is denoted by S_i. We use $s = (s_1, s_2, \ldots, s_n)$ and s_{-i} to denote a strategy profile of the agents, and a strategy profile of the agents other than i, respectively. Let S_{-i} denote the set of all s_{-i}'s.

We analyze games using *solution concepts*. Solution concepts are principles according to which we identify interesting subsets of the outcomes of a game. Solution concepts can be seen as a way to predict the outcome of a game assuming some behavior of rational agents who try to maximize their own utilities. The most significant solution concept in game theory is the *Nash equilibrium*. In the Nash equilibrium, we analyze the games from an individual agent's point of view. The strategy that an agent chooses in a Nash equilibrium is called the *best response*.

Definition 2 (Best Response)**.** *Agent i's best response to the strategy profile* s_{-i} *is a mixed strategy* $s_i^* \in S_i$ *such that* $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$ *for all strategies* $s_i \in S_i$*.*

Subsequently, the Nash equilibrium is defined as follows.

Definition 3 (Nash Equilibrium)**.** *A strategy profile* $s = (s_1, \ldots, s_n)$ *is a Nash equilibrium if, for all agents i,* s_i *is a best response to* s_{-i}*.*

The other strategy that is of central significance is the *dominant strategy*. To define this strategy, first we define weak domination.

Definition 4 (Weak Domination)**.** *Let s_i and s_i' be two strategies of agent i. Then s_i weakly dominates s_i' if for all s_{-i}, $u_i(s_i, s_{-i}) \geq u_i(s_i', s_{-i})$, and for at least one $s_{-i} \in S_{-i}$, it is the case that $u_i(s_i, s_{-i}) > u_i(s_i', s_{-i})$.*

Consequently, a weakly dominant strategy is defined as follows.

Definition 5 (Weakly Dominant Strategy)**.** *A strategy is weakly dominant for an agent if it weakly dominates any other strategy for that agent.*

A strategy profile $(s_1, \ldots, s_n)$ in which every s_i is weakly dominant for agent i is called *equilibrium in weakly dominant strategies*. Other types of dominant strategies, strictly and very weakly dominant strategies, can be defined accordingly. However, in this thesis, we only work with weakly dominant strategies and, for simplicity, we drop the modifier "weakly". An equilibrium in dominant strategy is a stronger concept than the Nash equilibrium. Every equilibrium in dominant strategies is a Nash equilibrium but not always a Nash equilibrium is an equilibrium in dominant strategies.

Back to our discussion about mechanism design, let us recall that mechanism design is in fact inverse game theory. In mechanism design, we assume unknown individual preferences (utilities), and attempt to design a game such that no matter what the unknown preferences are, the equilibrium (e.g. equilibrium in dominant strategies) of the game is guaranteed to have a certain set of properties. Mechanism design is perhaps the most "computation-driven" part of game theory since it concerns itself with the design of protocols for distributed decision makers. However, since the decision makers are not necessarily cooperative, one can think of mechanism design as an exercise in "incentive engineering".

1.2 MECHANISM DESIGN FOR COMBINATORIAL AUCTIONS

While mechanism design is a broad area in game theory, we study mechanism design for welfare maximization in combinatorial auctions. The problem of maximizing *social welfare* in *combinatorial auctions* (CAs) is of central importance in both theory and practice of mechanism design. In CA, there are m items for sale and n bidders competing for these items. Each bidder i has a private valuation $v_i(S)$ for each package S of items. Alternatively, we say v_i is the private type of bidder i. The social welfare of an allocation $S_1, \ldots, S_n$ of items to the bidders is $\sum_{i=1}^{n} v_i(S_i)$.

An outcome, in the setting of CA, is a pair of allocations as well as the payments to the bidders: $o = (S_1, \ldots, S_n, p_1, \ldots, p_n)$. In CA, bidders usually have quasi-linear utility functions. A Quasi-linear utility function is of the form $u_i(S_i) = v_i(S_i) - p_i$: the utility of a

bidder i for an allocation is the private valuation of the bidder less the payment for the allocation. With this type of utility functions, the mechanism can charge or reward the bidders by an arbitrary amount of money in order to incentivize the bidders to truthfully report their values.

A mechanism implements a *social choice function* in an *equilibrium*. A social choice function is a mapping from types to outcomes. In our setting, the social choice function is optimizing social welfare: $\max_{S_1,\ldots,S_n} \sum_{i=1}^{n} v_i(S_i)$.

Bidders choose strategies to bid in an auction. A strategy is a function from private types (valuations) to actions (bids). Bidders are rational, that is they bid to maximize their own utility. A strategy is dominant, if the bidder bids regardless of the bids submitted by other bidders. A strategy profile in which every bidder has a dominant strategy is called an *equilibrium in dominant strategies*. We seek a mechanism that maximizes social welfare in an equilibrium in dominant strategies.

In order to address a mechanism design problem, the *revelation principle* encourages us to focus on *incentive-compatible direct* mechanisms. In a direct mechanism, the only action available to each agent is to report its private information. A direct mechanism is incentive compatible or *truthful* in dominant strategies if every agent i has a dominant strategy to truthfully report its private type.

Definition 6 (Revelation Principle)**.** *If there exists any mechanism that implements a social choice function in dominant strategies then there exists a direct mechanism that implements the social choice function in dominant strategies and is truthful.*

In other words, the task of solving a mechanism design problem can be addressed by finding a direct and truthful mechanism. Thus, we limit ourselves to social-welfare maximizing mechanisms which are direct and incentive compatible in dominant strategies (truthful). In the present text, we use the terms ”truthfulness” and ”incentive compatibility” interchangeably, and by either we mean ”incentive compatibility in dominant strategies”. In order to use this useful observation, we have to take into consideration two important points. First, by using the revelation principle the computational burden of finding strategies for the agents is pushed onto the mechanism. This fact draws our attention to the computational complexity of the underlying algorithm of the mechanism. Second, since agents have to fully reveal their types, this can place a burden on the communication channel. In our mechanisms, we have considered these points, and the proposed mechanisms do not suffer from any of these issues. Given that we only work with *direct-revelation* mechanisms, from now on, we use the word mechanism rather than direct mechanism.

To be more specific about mechanism design for the CA setting, we say a mechanism is a protocol with three components. First, the

mechanism extracts a report from each bidder that describes the bidder's valuations. Second, the mechanism computes an allocation that determines the package of items to be allocated to each bidder (allocation rule). Third, the mechanism charges each bidder an amount of money (payment rule).

If each bidder, in anticipation of the outcomes of the allocation and payment rule, reports her valuation truthfully, it is said that the auction mechanism is incentive-compatible in dominant strategies or truthful.

Incentive compatibility in dominant strategies is stronger than the *Bayes-Nash incentive compatibility*. In Bayes-Nash incentive compatible mechanism, a bidder is truthful provided that other bidders bid truthfully. Incentive compatibility in dominant strategies assures that the strategy of each bidder, regardless of the bids submitted by other bidders, is to bid truthfully.

Back to our example of finding the maximum among numbers, if we assume the agents have quasi-linear utilities, we can model the problem as a mechanism design problem for selling one item truthfully to the agent whose private value for the item is maximum. With quasi-linear utility functions, Vickrey's mechanism called *second price auction* solves this mechanism design problem. In particular, it finds the maximum value reported by the agents and charges the corresponding agent an amount equal to the second highest number. It is well known that in a second price auction, each agent truthfully reports its private type (Vickrey, 1961).

Thus, the algorithm for finding the maximum works correctly provided that the agents have quasi-linear utilities and are told about the possible payment by the owner of the maximum number. This phenomena is quite amazing in that despite the private data and pure selfish behavior, the maximum number can be correctly found. All the field of mechanism design is just a generalization of this possibility.

1.3 VCG AUCTIONS

In combinatorial auctions, the goal of mechanism design is to maximize social welfare while providing the bidders with incentives to truthfully report their private valuations. In economics, the celebrated Vickrey-Clarke-Groves (VCG) mechanism provides both objectives. In particular, VCG finds the welfare maximizing allocation $S_1^*, \ldots, S_n^*$, and employs the following payment rule in order to guarantee incentive compatibility. Each bidder i pays her externality, the amount by which his allocated bundle reduced the total reported value of the bundles allocated to others: $\sum_{l=1,l\neq i}^{n} v_i(S_i') - \sum_{l=1,l\neq i}^{n} v_i(S_i^*)$. The first term is the maximum social welfare obtainable without bidder i; that is $(S_1', \ldots, S_n')$ maximizes social welfare in a market without

bidder i. The second term is the welfare of the market without bidder i under the social welfare maximizing allocation $S_1^*, \ldots, S_n^*$.

The success of VCG strongly relies on optimizing the social welfare. Computationally, this is not always possible because CA is a NP-hard problem. Even a restricted type of CAs in which bidders are single-minded is NP-hard since we can show a reduction from the independent-set problem to the CA problem with single-minded bidders (Blumrosen and Nisan, 2007). A single-minded bidder is interested only in a single specific package of items and gets a specific value if she gets the whole package (or any superset) and zero value for any other package. An essential question arises here. Can we use approximation algorithms – that are usually employed by computer scientists to tackle hard problems – and then use VCG-like payments to obtain truthfulness?

1.4 TRUTHFUL APPROXIMATION MECHANISMS

An algorithm is an α-approximation algorithm (or has approximation ratio α) if the the objective value of the computed solution is at least a factor α of the value of the optimal allocation (the allocation with maximum objective function value). Unfortunately, approximating the social welfare and then using the idea of VCG payments does not preserve truthfulness (Nisan and Ronen, 2001). For instance, assume in a single-item auction, we assign the item to the second highest bid (as an approximation to the highest value) at the price of the third highest bid. This auction has no equilibrium because for example the bidder with the highest value would decrease her bid below the second bid to obtain the item, and similarly, the second bidder would decrease her bid and so on.

Resolving the tension between approximation algorithms and incentive compatibility is the topic of *Algorithmic Mechanism Design* (Nisan and Ronen, 2001). In particular in algorithmic mechanism design we ask ourselves if we can theoretically guarantee a ratio for an incentive-compatible approximation algorithm with private input data that (almost) equals the ratio of and approximation algorithm presented for the same problem with public input data? This problem has been extensively studied in the last two decades for various types of valuations (Blumrosen and Nisan, 2007).

Not surprisingly, randomized algorithms have been proven to be more promising than deterministic algorithms in algorithmic mechanism design. For example, the two main frameworks presented by Lavi and Swamy (LS framework) and Dughmi et al. (convex rounding) are randomized and yield truthfulness in expectation. Truthfulness in expectation certifies that bidders who are risk-neutral (expected utility maximizer) have no incentive to report false valuations.

Generally speaking, the two frameworks rely on the idea of *relaxation* and *rounding*. The LS framework first optimizes a LP relaxation of the problem, and then uses a specific rounding technique called *meta-randomized rounding* originally proposed by Carr and Vempala (2000). Meta-randomized rounding yields a convex decomposition of a fractional point into polynomially-many integer points. The rounding technique has been successfully applied to mechanism design with (Lavi and Swamy, 2011) and without money (Dughmi and Ghosh, 2010). One drawback of the first presentations of the meta-randomized rounding is that, the technique relies on the ellipsoid method which is notoriously of low practical usability and is mostly of theoretical importance (Carr and Vempala, 2000; Lavi and Swamy, 2011).

The convex rounding technique optimizes a convex function which is the image of a rounding algorithm over a relaxed set of solutions. Convex rounding optimizes over the outcome of a rounding algorithm, and this way the rounding algorithm is integrated into the optimization problem. We will discuss both techniques in the next chapters.

1.5 APPROXIMATIONS TO OBTAIN TRUTHFULNESS

Not always quasi-linear utilities and transfer of money are available. There are many situations in which no money is involved because of the nature of the problem or because of the law. Truthful mechanism design without money under general preferences is a classic topic in social choice theory. Without money, mechanism designers face significant obstacles in truthful mechanism design. Think about the combinatorial auctions setting defined above. The main idea of truthfulness strongly relies on the payments. Payments enter the utility of the bidders and are leveraged to incentivize the bidders to bid truthfully. When payments are not available, the utility of the agents depend only on their valuation for the allocated package of items. For payment-free environments, it sounds unlikely for the mechanism designers to be able to provide agents with incentives to truthfully report their private types.

It is in fact the case; the Gibbard-Satterthwaite theorem proves that the class of truthful mechanisms is limited to dictatorships (Gibbard, 1973; Satterthwaite, 1975). In particular, it states that any truthful social choice function which selects an outcome among three or more alternatives has to be trivially aligned with the preference of a single agent. There have been a number of extensions analyzing more specific domains without money, all resulting in impossibility results (Pápai, 2001; Ehlers and Klaus, 2003; Hatfield, 2009). To circumvent the Gibbard-Satterthwaite impossibility, researchers have introduced restricted domains with additional assumptions to admit

truthful mechanisms. For example, when agents valuations are restricted to single-peaked preferences over a one-dimensional public space, returning the *median* of the peaks determines a truthful social choice (Moulin, 1980). Single-peaked preferences have a single most-preferred point in an interval, and are decreasing as one moves away from that peak.

Procaccia and Tennenholtz introduced the technique of welfare approximation as a means to drive truthful approximation mechanisms without money (Procaccia and Tennenholtz, 2013). This type of approximation is not meant to handle computational intractability but a method to achieve truthfulness. We show applications of this technique in mechanism design without money for some restricted domains of preferences.

1.6 THESIS STRUCTURE

This thesis is divided into seven chapters. In the present chapter, we introduced the problem of mechanism design for combinatorial auctions, and mentioned some of the challenges and existing techniques in the field. In Chapter 2, we describe a general technique to obtain truthful approximation algorithms which is based on relaxation and rounding. Chapter 2 contains the common denominator of the techniques used in the next chapters. In Chapter 3, we extend the idea of convex rounding and propose a truthful-in-expectation algorithm for a strategic variant of the generalized assignment problem. The approximation presented (almost) matches the best approximation ratio given for the generalized assignment problem. This chapter is based on the paper, Fadaei and Bichler (2014). In Chapter 4, we use the technique of welfare approximation to provide a truthful approximation mechanism for strategic variations of the generalized assignment problem in a payment-free environment. This chapter is based on the work, Fadaei and Bichler (2016). Since the meta-randomized rounding is so ubiquitous in algorithmic mechanism design, we maintain the next two chapters for the computational aspects of the meta-randomized rounding. In Chapter 5, we present a fast and geometric-based method for the implementation of the meta-randomized rounding. This chapter is based on the paper, Kraft, Fadaei, and Bichler (Kraft et al., 2014). Chapter 6, shows how to use a variant of the simplex method which is well known to be efficient in practice to implement the meta-randomized rounding. This chapter is based on the work, Fadaei (2015). Finally, we conclude with some related directions of research in Chapter 7.

2

RELAX AND ROUND

Usually, when faced with NP-hard problems, computer scientists turn to approximations or heuristics. An approximation algorithm runs in polynomial time in the size of the encoding of input data, and returns a provable approximation of the optimal solution. The approximation ratio is proven with respect to the worst-case analysis of the algorithm. An algorithm is an α-approximation algorithm (or has approximation ratio α) if the the objective value of the computed solution is at least a factor α of the value of the optimal solution.

Relaxation and Rounding (henceforth, *relax* and *round*) is a well-known and ubiquitous technique for designing approximation algorithms. The relax and round technique is also the most successful machinery to design *truthful* approximation algorithms that run in polynomial time. For quasi-linear utilities the idea of relax and round is omnipresent. The idea has also been applied to mechanism design without money.

Given the significance of the idea of relax and round in the literature, and that our truthful mechanisms utilize the idea, in this chapter, we elaborate further on it. We explain the idea for quasi-linear valuations first, and towards the end of the chapter, we explain how to apply the technique to mechanism design without money.

2.1 SETTING

In a market, a group of n bidders are vying for a set of items. A feasible solution is an allocation of items to bidders, satisfying (possibly) some given constraints. Each bidder i has a *private* valuation function v_i with value $v_i(S_i)$ for a feasible solution $(S_1, S_2, \ldots, S_n) \in \mathcal{S}$, where $\mathcal{S}$ denotes the set of all feasible solutions. For simplicity, we denote the value of bidder i for solution $x = (S_1, S_2, \ldots, S_n)$ by $v_i(x)$, where $v_i(x) = v_i(S_i)$. Let V_i denote the set of all valuation functions of bidder i, and $V = V_1 \times V_2 \times \ldots \times V_n$ denote the set of all profiles of valuations. Denote the profile of valuations of bidders other than i by v_{-i}. Define function $f : \mathcal{S} \to \mathbb{R}_+$ with $f(x) = \sum_i v_i(x)$ for all $x \in \mathcal{S}$.

The following optimization problem expresses the social welfare maximization problem.

$$\begin{aligned} \text{Maximize} \quad & f(x) \\ \text{subject to} \quad & x \in \mathcal{S} \end{aligned} \tag{1}$$

The social welfare of a solution $x \in \mathcal{S}$ thus equals $f(x)$. From now on, we analyze Problem 1 as a mechanism design problem. Specifically, we wish to maximize social welfare while making sure each bidder i reports her valuation function v_i truthfully. Given that v_i's are private, we build program 1 from the reported valuations (bids). If the optimization Problem 1 can be solved in polynomial time, then VCG can be employed as a dominant-strategy incentive-compatible (truthful) mechanism. However, in many cases including combinatorial auctions, Problem 1 is NP-hard, and we can only hope for approximations of the optimal solution. In what follows, we describe how to employ the idea of relax and round to achieve mechanisms that satisfy the weaker notion of truthfulness in expectation.

2.2 RELAXATION

The idea of relax and round proceeds as follows. We find a relaxation of Problem 1, solve it and then carefully round the outcome so as to obtain particular properties. The relaxed problem is as follows.

$$\begin{aligned} \text{Maximize} \quad & L(x) \\ \text{subject to} \quad & x \in P \end{aligned} \tag{2}$$

Where P is a relaxed set of solutions that contains $\mathcal{S}$, and may contain infeasible solutions. Polytope P is the intersection of a set of linear constraints in the positive orthant, and is a *packing* polytope: if $x \in P$ and $y \leq x$ then $y \in P$. Function $L : P \to \mathbb{R}_+$ is concave or linear in x, and for every $x \in \mathcal{S}$, it is the case that $L(x) \geq \alpha f(x)$ for some α, $0 < \alpha \leq 1$. As we see below, α will be the approximation ratio of the algorithm. Function L is separable, i.e. $L(x) = \sum_i L_i(x)$, however $L_i(x)$ need not necessarily be concave. Let us call L *relaxed objective function*. Recall, the mechanism designer has only access to the reported valuations, thus all value computations are carried out with respect to the reported valuations.

Given that function L is concave and all constraints of polytope P are linear, Problem 2 is a convex optimization problem that can be solved efficiently. Once we solved Problem 2, we proceed to the next step, rounding.

2.3 ROUNDING

A randomized rounding scheme $r : P \to \mathcal{S}$ returns a randomized solution $X \sim r(x)$ for any $x \in P$. The rounding scheme is *oblivious*,

i.e. it does not depend on the valuations. Solution $X = (S_1, \ldots, S_n)$ is always feasible, i.e. $Pr[X \notin \mathcal{S}] = 0$. With regard to the expected value of the rounded solution (with respect to reported valuations), one of the following cases may occur:

(a) $\mathbb{E}[f(X)] > L(x)$. In this case, apply an oblivious rounding scheme r' to X in order to obtain $X' = (S'_1, \ldots, S'_n)$ such that $\mathbb{E}[f(X')] = L(x)$.

(b) $\mathbb{E}[f(X)] < L(x)$ but $\mathbb{E}[f(X)] \geq \beta L(x)$ for some β, $0 < \beta \leq 1$. Apply an oblivious rounding r' to X in order to obtain X' such that $\mathbb{E}[f(X')] = \beta L(x)$. In this case, the approximation ratio of the algorithm will be $\alpha\beta$.

(c) $\mathbb{E}[f(X)] = L(x)$. No more action is needed, $r'(X) = X$.

We will refer to the three cases of the rounding, (a), (b), and (c) in the following.

2.4 TRUTHFUL-IN-EXPECTATION MECHANISM

Here, we explain how to use the foregoing idea of relax and round to define a truthful-in-expectation mechanism. A mechanism comprises an allocation rule $\mathcal{A}$ and a payment rule p. A mechanism $(\mathcal{A}, p)$ is truthful-in-expectation if for every player i with true valuation v_i and reported valuation v'_i, we have

$$\mathbb{E}[v_i(\mathcal{A}(v)) - p_i(v)] \geq \mathbb{E}[v_i(\mathcal{A}(v'_i, v_{-i})) - p_i(v'_i, v_{-i})]. \tag{3}$$

The expectation in (3) is taken over the coin flips of the mechanism. In order to achieve a truthful-in-expectation mechanism, we propose the following allocation rule.

Algorithm 2: Allocation Rule.

1. Let $x^* = \arg\max_{x \in P} L(x)$
2. Let $X' \sim r'(r(x^*))$; `/* r' is chosen according to the occurring case: (a), (b), or (c). */`

return X'

To define the payments, let $x^* = \arg\max_{x \in P} L(x)$, and $(S_1, \ldots, S_n) \sim r'(r(x^*))$. The payment rule determines the payment by each bidder. The expected VCG payment of any bidder k is equal to

$$p_k = \max_{x \in P} L^{-k}(x) - \mathbb{E}[\sum_{i, i \neq k} v_i(S_i)]. \tag{4}$$

Where $L^{-k}(x)$ is the relaxed objective function for a market in which bidder k is discarded, and the allocation rule is run for the society of all other bidders. Equivalently, we can say $L^{-k}(x)$ is defined assuming $v_k(x) = 0$, $\forall x \in P$. Let $x^*_{-k} = \arg\max_{x \in P} L^{-k}(x)$, and $(T_1, \ldots, T_n) \sim r'(r(x^*_{-k}))$. If we let any bidder k pay an amount

of $\sum_{i,i\neq k} v_i(T_i) - \sum_{i,i\neq k} v_i(S_i)$, then using linearity of expectation, the expected value of this payment is equal to the expression in (4).

Theorem 1. *The allocation rule above (Algorithm 2) supplemented with the payment rule (4), constitute a truthful-in-expectation mechanism with a provable approximation ratio of α (or $\alpha\beta$).*

Proof. Let $x^* = \arg\max_{x\in P} L(x)$. Fix bidder k and v_{-k} the valuations of bidders other than k. Let $(S_1, \ldots, S_n) \sim r'(r(x^*))$. Let $C = \max_{x\in P} L^{-k}(x)$. The expected utility of bidder k from reporting true values v_k is the following.

$$\begin{aligned}\mathbb{E}[u_k(v)] &= \mathbb{E}[v_k(S_k)] - p_k \\ &= \mathbb{E}[v_k(S_k)] - (C - \mathbb{E}[\textstyle\sum_{i,i\neq k} v_i(S_i)]) \\ &= L(x^*) - C.\end{aligned}$$

Recall, $L(x^*) = \mathbb{E}[\sum_i v_i(S_i)]$ by the construction of the rounding. The second term C is independent of the valuations of bidder k, thus the bidder cannot influence it. In order to increase her expected utility, the bidder has to increase the first term, but the first term is already the maximum possible value. Hence, the bidder cannot improve her expected utility. To be more specific, let bidder k report v'_k rather than v_k. Let $x' = \arg\max_{x\in P} L'(x)$, where L' is the relaxed objective function obtained from the new reported valuations (v'_k, v_{-k}). Let $(S'_1, \ldots, S'_n) \sim r'(r(x'))$. The expected utility of the bidder, in this case, will be lower as shown below.

$$\begin{aligned}\mathbb{E}[u_k(v'_k, v_{-k})] &= \mathbb{E}[v_k(S'_k)] - (C - \mathbb{E}[\textstyle\sum_{i,i\neq k} v_i(S'_i)]) \\ &= \mathbb{E}[v_k(S'_k)] + \mathbb{E}[\textstyle\sum_{i,i\neq k} v_i(S'_i)] - C \\ &= L(x') - C \\ &\leq L(x^*) - C \\ &= \mathbb{E}[u_k(v)]\end{aligned}$$

For the first equality, recall $L'(x') = \mathbb{E}[v'_k(S'_k)] + \mathbb{E}[\sum_{i,i\neq k} v_i(S'_i)]$, however, the utility of bidder k is calculated with respect to true valuation v_k. Thus, the bidder has no incentive to report a false valuation function.

The approximation ratio admits a simple proof. Consider an optimal integral solution $Y^* = (S^*_1, \ldots, S^*_n)$. Since $Y^* \in P$, we have $L(Y^*) \leq L(x^*)$, where $x^* = \arg\max_{x\in P} L(x)$. By definition of L, we have $L(Y^*) \geq \alpha f(Y^*)$. Let $X' \sim r'(r(x^*))$. By the construction of the rounding, we have $\mathbb{E}[f(X')] = L(x^*) \geq L(Y^*) \geq \alpha f(Y^*)$, the desired conclusion. Similarly, we can show the approximation ratio of $\alpha\beta$ for case (b) of the rounding. This completes the proof. □

Therefore, we explained how one can use the relax and round technique to define an allocation and a payment rule to obtain a truthful-in-expectation mechanism.

The technique of relax and round that we explained here is an abstraction of the existing truthful approximation algorithms that employ relaxation and rounding technique. Here, we abstract away from many details of the technique. These details are varied for different applications. For example, in our explanation, we gave no specification about how to define function L, or the rounding algorithms r and r'. It is instructive to note that, from the provided explanation, we observe that the relax and round technique has an extra step (calling r') when used for designing *truthful* approximation mechanisms compared to the usage of the technique for approximation algorithms. This can also be seen as a reason for the fact that designing truthful approximation algorithms is more stringent than designing (untruthful) approximation algorithms. Several truthful mechanisms that are grounded in mathematical optimization follow a variation of the foregoing relax and round technique. We briefly explain this observation in the following.

The framework proposed by Lavi and Swamy (LS framework) can be described as follows. The LS framework, first relaxes the underlying integer program of the problem to a linear program (LP), and solves the LP. Then the solution of the LP is scaled down by a specific factor (the integrality gap of the underlying polytope), and the meta-randomized rounding is applied to the scaled-down solution (Lavi and Swamy, 2011). For more details about LS framework, we refer the reader to chapters 5 and 6.

In the language of the relax and round technique that we described above, the LP in the LS framework is closely related to Problem 2. Function L is in fact the objective function of the LP, therefore L is a linear function. Polytope P is the region of feasible solutions of the LP scaled down by the integrality gap. Therefore, P does not contain all solutions of $\mathcal{S}$. However, P has a special property: for any $v \in V$, there exists a point $x \in P$ such that $L(x) \geq \alpha \cdot \max_{y \in \mathcal{S}} \sum_i v_i(y)$, where $1/\alpha$ is the integrality gap of the LP. Moreover, in the LS framework for every $x \in \mathcal{S}$, $L(x) = f(x)$. The meta-randomized rounding used in the LS framework results in case (c) of the rounding step. Therefore, no second rounding r' is required.

Convex rounding is the other general framework for designing truthful approximation mechanisms (Dughmi et al., 2011b). In convex rounding, the allocation rule optimizes directly on the outcome of the rounding algorithm, rather than over the outcome of the relaxation algorithm. The convex rounding can be described by the relax and round technique as follows. Function L is a concave function. Polytope P is simply a relaxed set of feasible solutions. The rounding scheme ends in case (c) of the rounding step.

Archer et al. propose a truthful-in-expectation mechanism for combinatorial auctions with single parameter agents using the idea of relax and round (Archer et al., 2004). The authors employ a linear

programming relaxation for the problem, and the rounding case (b) occurs. Similarly, Dobzinski et al. propose a truthful-in-expectation mechanism for combinatorial auctions with subadditive bidders. They also use a linear programming relaxation and the rounding case (b) occurs (Dobzinski et al., 2010).

In the solution presented for the generalized assignment problem in this thesis, we use a concave objective function and a polytope which contains all integral solutions as well as infeasible solutions, and the rounding case (a) happens. In addition, we employ the following new observation. From the explanation above, we know that the rounding schemes r and r' are oblivious. It is possible to extend this idea to non-oblivious rounding schemes given that the following condition holds. A rounding scheme $r : P \times V \to \mathcal{S}$ preserves truthfulness if for any misreported valuation v_i', we have $\mathbb{E}[f(r(x, v_i', v_{-i}))] \leq \mathbb{E}[f(r(x, v))]$. This condition assures that the expected value of the social welfare is maximized by truthful bidding. See Chapter 3 for more details.

In the literature, the allocation rule defined in Algorithm 2 is termed Maximal-In-Distributional-Range (MIDR) algorithm. A MIDR algorithm optimizes over a distributional range of the solutions. To preserve truthfulness, the distributional range of the solutions must be fixed independently of the valuations of the bidders (Dobzinski and Dughmi, 2009). The distributional range implied by the relax and round technique described above is the image of the rounding scheme:

$$\textit{Distributional Range} \equiv \bigcup_{x \in P} \{r'(r(x))\}. \tag{5}$$

The (distributional) range in (5) is independent of bidders' valuations. Since we optimally maximize over this range, Algorithm 2 is in fact a MIDR algorithm.

2.5 MECHANISM DESIGN WITHOUT MONEY

To use the relax and round technique for mechanism design without money, we do as follows. First, we relax the underlying integer program of the problem to a linear program. Let P denote the region of the feasible solutions to the linear program. We find a fractional point in P by using an algorithm that is *fractionally truthful*. A fractionally truthful algorithm $\mathcal{A}$ takes $v \in V$ and P as input, and computes a point $x \in P$ such that for any bidder i and untruthful valuation v_i', we have $v_i(\mathcal{A}(v)) \geq v_i(\mathcal{A}(v_i', v_{-i}))$. To maintain a good approximation of the total value, the value of the computed point x must be at least as good as α times ($0 < \alpha \leq 1$) the optimal solution to the relaxed problem.

Next, we round x using a randomized rounding scheme r with the following properties. If $X \sim r(\mathcal{A}(v, P))$ we must have (i) (feasibility)

$Pr[X \in \mathcal{S}] = 1$ and (ii) (truthfulness) $\forall i$, $\mathbb{E}[v_i(X)] = \beta v_i(x)$ for $0 < \beta \leq 1$.

This technique results in a truthful-in-expectation $(\alpha\beta)$-approximation algorithm. We refer the reader to Chapter 4 for more details.

3

GENERALIZED ASSIGNMENT PROBLEM

We propose a truthful-in-expectation, $(1 - 1/e)$-approximation mechanism for a strategic variant of the generalized assignment problem (GAP). In GAP, a set of items has to be optimally assigned to a set of bins without exceeding the capacity of any singular bin. In the strategic variant of the problem we study, values for assigning items to bins are the private information of bidders and the mechanism should provide bidders with incentives to truthfully report their values. The approximation ratio of the mechanism is a significant improvement over the approximation ratio of the existing truthful mechanism for GAP.

The proposed mechanism comprises a novel convex optimization program as the allocation rule as well as an appropriate payment rule. To implement the convex program in polynomial time, we propose a fractional local search algorithm which approximates the optimal solution within an arbitrarily small error leading to an approximately truthful-in-expectation mechanism. The presented algorithm improves upon the existing optimization algorithms for GAP in terms of simplicity and runtime while the approximation ratio closely matches the best approximation ratio given for GAP when all inputs are publicly known.

3.1 INTRODUCTION

We analyze the generalized assignment problem (GAP) in an environment where valuations are private information of distributed decision makers. In GAP, a set of m items has to be assigned to a set of n bins. Each bin associates a different value and weight to each item and has a limited capacity. An allocation may assign each bin a subset of items not exceeding the capacity of the bin. For each of these subsets, the valuation is additive in the values of items contained in the subset. The goal is to find a feasible assignment of items to bins to maximize social welfare, the sum of generated values by the assignment.

GAP has also been defined in the literature as a (closely related) minimization problem. In the minimization GAP, the assignment of items to bins incurs costs; the goal of this optimization problem is to find a feasible assignment of minimum total cost. From an optimiza-

tion point of view, these two variants of GAP are equivalent (Martello and Toth, 1992).

GAP is a well-known problem in combinatorial optimization and operations research. It can be considered as a generalization of the problem of finding a maximum weight matching in a weighted bipartite graph, the assignment problem (Kuhn, 1955; Ferland, 1998). GAP has many real-world applications including applications in resource scheduling problems such as machine scheduling, classroom scheduling and employee scheduling (Zimokha and Rubinstein, 1988). GAP is commonly applied in transportation and routing (Ruland, 1999; Fisher and Jaikumar, 1981). There is a long list of reported applications of GAP in telecommunication, production planning and facility location applications (Bressoud et al., 2003; Dobson and Nambimadom, 2001; Ross and Soland, 1977; Klastorin, 1979). GAP can also be applied in supply chain and logistics. Kalagnanam et al. (2001) discuss the computational complexity of clearing markets in a double auction and formulates the problem with GAP. For a survey of applications of GAP, we refer the reader to Öncan (2007).

In many situations, weights and capacities are intrinsic attributes of items and bins and are therefore readily known and verifiable. For example, in process industries such as paper and steel, standard geometries such as width, length and weight are used, and buyers typically bid for rolls of paper or steel of a desired width (Kalagnanam et al., 2001).

By contrast, the associated value of a specific assignment has to be extracted through communication with buyers. In service areas, for example, the cost of providing a service to a certain group (weight) at an affordable cost (capacity) are known, although the business value of a service is only known to the recipient of the service.

Examples like these motivate the study of a strategic variant of GAP where valuations are assumed to be private information known only to the bidders while weights and capacities are publicly known. There are two obstacles in finding a solution to this strategic variant. First, GAP is a NP-hard problem. Hence, computing its optimal social welfare is intractable even if the valuations are known. Second, maximizing the social welfare necessitates knowing valuations, but this is private information and can be truthfully extracted solely by providing incentives using payment rules. Overcoming both of these obstacles simultaneously is the subject of *algorithmic mechanism design* (Nisan and Ronen, 2001).

In this work, we propose a solution by which the two obstacles, maximizing the social welfare of GAP as well as extracting true valuations of bidders, are surmounted. The solution provides bidders with incentives to report their valuations truthfully and runs in polynomial time approximating the social welfare with a provable ratio of at least $1 - 1/e$.

3.1.1 *Challenges in Algorithmic Mechanism Design*

In algorithmic mechanism design, a mechanism designer wishes to solve an optimization problem, but the inputs to this problem are the private information of self-interested players. The mechanism designer must thus design a mechanism that solves the optimization problem while encouraging the bidders to truthfully reveal their information. The game-theoretic concept of truthfulness guarantees that a bidder is better off truthfully interacting with the mechanism regardless of what the other bidders do.

The well-known *Vickrey-Clarke-Groves* (VCG) technique provides truthfulness as well as social welfare maximization in every combinatorial auction. The VCG technique, however, is applicable only when the optimal social welfare can be computed to optimality. Yet, in many cases, including our problem, optimizing social welfare is computationally intractable which makes the VCG technique inapplicable. Usually, when faced with computational intractability, computer scientists turn to approximations or heuristics. The VCG technique, unfortunately, cannot be directly applied to approximate solutions (Nisan and Ronen, 2007). In order to resolve the clash between approximation and truthfulness, the maximal-in-distributional-range (MIDR) allocation rules are introduced.

MIDR is the only known general approach for designing randomized truthful mechanisms. An MIDR algorithm fixes a set of distributions over feasible solutions (the distributional range) independently of the valuations reported by the self-interested players, and outputs a random sample from the distribution that maximizes expected (reported) welfare (Dobzinski and Dughmi, 2009). The best option for a mechanism designer is to devise a MIDR containing an approximation of the optimal social welfare that (very closely) matches the best approximation guarantee known for the problem for which the underlying data are publicly known. Finding this type of MIDR or designing an approximation truthful mechanism is not always possible. Several authors have shown that it is impossible to achieve the same approximation factor in truthful mechanisms (Lavi et al., 2003; Papadimitriou et al., 2008; Dobzinski and Vondrák, 2013; Dughmi and Vondrák, 2015; Dobzinski and Vondrák, 2012).

Looking more closely at the approximation algorithms presented for GAP, we observe that no algorithm can serve as a MIDR allocation rule, although GAP without incentives has been studied extensively in the literature. Chekuri and Khanna (2005) explicitly state that the algorithm of Shmoys and Tardos (1993) can be adapted to provide a 2-approximation. Later, Fleischer et al. (2006) improved the factor to $1 - 1/e$. Using a reduction to submodular maximization subject to a matroid constraint, Calinescu et al. (2011) achieved a ratio of $1 - 1/e - o(1)$ without using the ellipsoid method which was pivotal

in the work done by Fleischer et al. (2006). An algorithm due to Feige and Vondrak (2006) yields an approximation factor of $1 - 1/e + \rho$, $\rho \approx 10^{-180}$ which is the best given approximation ratio for GAP. Chakrabarty and Goel (2010) provide the best-known hardness result showing it is NP-hard to approximate GAP to any factor better than 10/11.

According to our observations, all foregoing approximation algorithms comprise two algorithms: a *relaxation algorithm* and a *rounding algorithm*. In order to devise truthful mechanisms, Dughmi et al. (2011b) propose an approach which optimizes directly on the outcome of the rounding algorithm, rather than over the outcome of the relaxation algorithm. Since the rounding procedure is embedded into the objective function, this approach is not always computationally tractable. Assuming the optimization problem can be solved efficiently and the rounding scheme is independent of bidders' valuations, this approach always leads to a MIDR algorithm and is referred to as *convex rounding*.

Lavi and Swamy (2011) propose a framework for deriving MIDR mechanisms from linear programming relaxations. They solve the relaxed problem in the first step and then use a special rounding method (convex decomposition) to obtain a randomized integral allocation. Although Lavi and Swamy also use the common composition of relaxation and rounding algorithms, their special rounding procedure produces an expected allocation which is always identical to the scaled down input of the rounding algorithm, component-wise. Of interest, the rounding procedure used by Lavi and Swamy, guarantees truthfulness-in-expectation. Designing truthful mechanisms using the framework of Lavi and Swamy for a given problem is straightforward, however, this type of mechanisms is slow in practice and requires many black-box invocations of an existing approximation algorithm for the problem. Very recently, Azar et al. (2015) present a truthful-in-expectation 1/2-approximation algorithm for GAP with private values using the framework proposed by Lavi and Swamy and a new rounding technique.

3.1.2 *Results and Techniques*

It is possible to use the framework developed by Lavi and Swamy (2011) to design a truthful-in-expectation 1/2-approximation mechanism for GAP; in order to guarantee an improved approximation ratio as well as a higher performance, we follow the convex rounding technique. The main challenge in using convex rounding is to design an appropriate rounding scheme which induces a convex optimization problem. Moreover, the rounding scheme should return a feasible solution containing a good approximation of the fractional value.

We design a rounding algorithm with the desired properties for GAP. Using the rounding algorithm to obtain a MIDR, we directly optimize over the outcome of the rounding procedure rather than over the outcome of the relaxation algorithm. Using this technique, we formulate GAP as a convex optimization problem where the objective function equals the expected value of a rounding procedure. In contrast to Dughmi et al. (2011b), our rounding algorithm uses some information from bidders' valuations. Our design does not violate the truthfulness since we use bidders' values solely to search for the optimum in a subset of the range containing the optimal solution, as explained in Section 3.5. This design can be viewed as slightly extending the convex rounding technique and is of independent interest.

We supplement the allocation rule with a payment rule which allows the guarantee of *non-negativity of payments* and *individual rationality ex post* rather than providing these important properties only *ex ante*.

The approximation ratio of our mechanism very closely matches the best approximation ratio presented for GAP with publicly known valuations. In particular, the proposed convex program contains $1 - 1/e$ ratio of optimum while the best presented approximation ratio of non-truthful algorithms is $1 - 1/e + \rho$, $\rho \approx 10^{-180}$.

In order to solve the convex program, we present a fractional local search algorithm which approximates the proposed convex optimization problem within an arbitrarily small error, in the sense of an FPTAS. This leads to an approximate MIDR.

Theorem 2. *There is a $(1-\epsilon)$-MIDR allocation rule that achieves a $(1 - 1/e - \epsilon)$-approximation to the social welfare of the generalized assignment problem, for every $\epsilon = 1/poly(n)$.*

Dughmi et al. (2011a) show how to transform an approximately MIDR allocation rule to an approximately truthful-in-expectation mechanism (see Definition 9). With this black box transformation, we obtain the following.

Theorem 3. *There is a $(1-\epsilon)$-truthful-in-expectation mechanism that achieves a $(1 - 1/e - \epsilon)$-approximation to the social welfare of the generalized assignment problem, for every $\epsilon = 1/poly(m, n)$.*

From an algorithmic point of view, the proposed algorithm has advantages over the previously known optimization algorithms for GAP in terms of runtime and simplicity. We do not employ the ellipsoid method which is identified as pivotal in the work of Fleischer et al. (2006). Our algorithm improves on the one proposed by Calinescu et al. (2011). In each iteration of the algorithm by Calinescu et al. (2011), a random sampling is required to compute the residual increase of assigning an item to a bin which subsequently increases runtime. The residual increase is treated as an approximate evaluation of gradient of the objective function at a point. This residual

increase is in fact calculated by taking the average of $(mn)^5$ independent samples, where m and n are the number of items and bins, respectively. We use a novel objective function which is specified exactly, rather than by random sampling, whereby it is possible to explicitly calculate the gradient of the objective function which helps to simplify the algorithm and improve the runtime. It should be noted that in our design, we benefited from the ideas developed in Fleischer et al. (2006), Calinescu et al. (2011), and Dughmi et al. (2011a).

3.1.3 *Structure*

In Section 3.2 we introduce necessary notation and definitions. In Section 3.3 and Section 3.4 we present the MIDR allocation rule and the payment rule, respectively for the setting where bins are held by strategic bidders. Section 3.5 explains why the presented mechanism is truthful. The required modification of the mechanism for the case where items are held by bidders is explained in Section 3.6. Finally, in Section 3.7 we conclude with a summary and a discussion about future research questions.

3.2 PRELIMINARIES

In the generalized assignment problem, there are n bins, I, and m items, J. Let v_{ij} denote the value of bin i for item j. Each bin i has a different weight w_{ij} for each item j and has a limited capacity C_i. Let $\mathcal{F}_i$ denote the collection of all feasible assignments to bin i ($\forall S \in \mathcal{F}_i : \sum_{j \in S} w_{ij} \leq C_i$). Each item may be assigned to at most one bin. In the final allocation, some items may remain unassigned.

We assume weights and capacities are publicly known, yet values of assigning items to bins are the private information of bidders. More formally, we assume values $\{v_{ij}\}_{i \in I, j \in J}$ are private information of bidders. In the following, we assume *bins* are held by bidders and thus each bidder i has private valuations $\{v_{ij}\}_{j \in J}$. We analyze the case where *items* are held by bidders and each bidder j has private valuations $\{v_{ij}\}_{i \in I}$ in Section 3.6.

An allocation $(S_1, \ldots, S_n)$, where $S_i \subseteq J$ denotes the subset assigned to bin i, is feasible if $\forall i \in I : S_i \in \mathcal{F}_i$ and $\{S_i\}_{i \in I}$ are mutually disjoint. The valuation of bin i is defined as $g_i : 2^J \to \mathbb{R}_{\geq 0}$ such that $g_i(S) = \sum_{j \in S} v_{ij}$ if $S \in \mathcal{F}_i$, else $g_i(S) = 0$. With an slight abuse of notation, we sometimes use $g_i(S)$ instead of $g_i(S_i)$, where $S = (S_1, \ldots, S_i, \ldots, S_n)$. The social welfare obtained from a feasible allocation $(S_1, \ldots, S_n)$ is $\sum_{i \in I} g_i(S_i)$. The goal is to find a feasible allocation of maximum total social welfare.

In light of the revelation principle, we limit our attention to direct revelation mechanisms. Every mechanism has two main components: an *allocation rule* and a *payment rule*. The allocation rule $\mathcal{A}$ is a function

which maps a reported valuation $v = (v_1, \ldots, v_n)$ to an allocation $(S_1, \ldots, S_n)$, where $\forall i : v_i = (v_{ij})_{j \in J}$. The payment rule is a function from reported valuations to a required payment from each bidder. Let p_i denote the payment rule function for bidder i.

Definition 7 (Maximal in Distributional Range (MIDR))**.** *Given reported valuations $v_1, \ldots, v_n$, and a previously-defined probability distribution over feasible sets $\mathcal{R}$, a MIDR returns an outcome sampled randomly from a distribution $D^* \in \mathcal{R}$ that maximizes the expected welfare $\mathbb{E}_{x \sim D}[\sum_i g_i(x)]$ over all distributions $D \in \mathcal{R}$ (Dobzinski and Dughmi, 2009).*

Analogously, we define $(1-\epsilon)$-MIDR as follows.

Definition 8 ($(1-\epsilon)$-MIDR)**.** *Given reported valuations $v_1, \ldots, v_n$, and a previously-defined probability distribution over feasible sets $\mathcal{R}$, a $(1-\epsilon)$-MIDR returns an outcome sampled randomly from a distribution $D^* \in \mathcal{R}$ that $(1-\epsilon)$-approximately maximizes the expected welfare $\mathbb{E}_{x \sim D}[\sum_i g_i(x)]$ over all distributions $D \in \mathcal{R}$.*

An approximately truthful-in-expectation mechanism is defined as follows.

Definition 9 ($(1-\epsilon)$ truthful-in-expectation)**.** *A mechanism is $(1-\epsilon)$-approximately truthful-in-expectation for GAP if, for every bidder i, (true) valuation function v_i , (reported) valuation function v_i', and (reported) valuation functions v_{-i} of the other bidders,*

$$\mathbb{E}[g_i(\mathcal{A}(v_i, v_{-i})) - p_i(v_i, v_{-i})] \geq (1-\epsilon)\,\mathbb{E}[g_i(\mathcal{A}(v_i', v_{-i})) - p_i(v_i', v_{-i})]. \tag{6}$$

The expectation in (6) is taken over the coin flips of the mechanism.

The goal of our work is to find an allocation and payment rule which constitute a truthful-in-expectation mechanism for GAP and approximates the social welfare as much as possible.

3.3 MIDR ALLOCATION RULE FOR GAP

We optimize directly over the expected value of the allocation produced by a rounding algorithm. We let the relaxed feasible set be $\mathcal{R}$ as follows: given a vector $x \in \{0,1\}^{I \times 2^J}$, let $x_{i,S}$ indicate whether subset S is assigned to bin i.

$$\mathcal{R} = \left\{ x \in [0,1]^{I \times 2^J} \mid \forall i : \sum_{S \in \mathcal{F}_i} x_{i,S} \leq 1; \forall i \in I, \forall S \in \mathcal{F}_i : x_{i,S} \geq 0 \right\}.$$

In $\mathcal{R}$ one randomized feasible set is assigned to each bin i. The sets assigned to different bins may overlap, however in the rounding step each item is assigned only once.

Our intent is to maximize the expected value of the rounded allocation over relaxed set $\mathcal{R}$. This leads to a MIDR allocation, as explained in Section 3.5. Let us call the rounding algorithm r_{greedy}. Algorithm 3 presents the desired MIDR algorithm.

Algorithm 3: MIDR allocation rule for the generalized assignment problem.

Data: $v = (v_{ij})_{i \in I, j \in J}$.
Result: Feasible allocation $(S_1, \ldots, S_n)$.
1. Let x^* maximize $\mathbb{E}_{(S_1,\ldots,S_n) \sim r_{\text{greedy}}(x)}[\sum_{i \in I} g_i(S_i)]$ over $x \in \mathcal{R}$.
2. Let $(S_1, \ldots, S_n) \sim r_{\text{greedy}}(x^*)$.

Following is a step-by-step procedure to implement Algorithm 3 and a presentation of the benefits of the outcome of the algorithm. We start by explaining the rounding algorithm.

3.3.1 *Greedy Rounding*

We choose a rounding algorithm which preserves a good ratio of the fractional solution and returns a feasible allocation in which each item is assigned only once. We first define helper function $\phi(\cdot)$ which maps a point in $\mathcal{R}$ to a point in $[0,1]^{I \times J}$. Let $\phi : \mathcal{R} \to [0,1]^{I \times J}$ be such that $y = \phi(x)$ iff $\forall i \in I, \forall j \in J : y_{ij} = \sum_{S : j \in S} x_{i,S}$.

The rounding procedure, defined as Algorithm 5, has two steps. In the first step, given a point $x \in \mathcal{R}$ the rounding procedure finds another point $x' \in \mathcal{R}$ such that $\forall i \in I, \forall j \in J : y'_{ij} = 1 - e^{-y_{ij}}$, where $y = \phi(x)$ and $y' = \phi(x')$. In the second step, the rounding procedure assigns subset S to bin i with probability $x'_{i,S}$ while resolving conflicts as explained in Algorithm 5.

We propose Algorithm 4 to perform the first step. Algorithm 4 takes a point $x \in \mathcal{R}$ and a desired vector $y' \in [0,1]^{I\times J}$, where $y' \preceq \phi(x)$ and returns another point $x' \in \mathcal{R}$ such that $y' = \phi(x')$.

Algorithm 4: An oblivious method for finding a dominated point in $\mathcal{R}$.

Data: $x \in \mathcal{R}$, and $y' \in [0,1]^{I\times J}$ such that $y' \preceq \phi(x)$.
Result: $x' \in \mathcal{R}$ such that $y' = \phi(x')$.
Initialize $x' := x$; $\delta = \phi(x') - y'$, where $\delta \in [0,1]^{I\times J}$.
foreach *bin* i **do**
  foreach *item* j **do**
    repeat
      Choose $x'_{i,S:j\in S} > 0$, arbitrarily;
      if $x'_{i,S} < \delta_{ij}$ **then**
        $\delta_{ij} := \delta_{ij} - x'_{i,S}$;
        if $S \setminus \{j\} \neq \emptyset$ then $x'_{i,S\setminus\{j\}} := x'_{i,S\setminus\{j\}} + x'_{i,S}$;
        $x'_{i,S} := 0$;
      else
        $x'_{i,S} := x'_{i,S} - \delta_{ij}$;
        if $S \setminus \{j\} \neq \emptyset$ then $x'_{i,S\setminus\{j\}} := x'_{i,S\setminus\{j\}} + \delta_{ij}$;
        $\delta_{ij} := 0$;
    until $\delta_{ij} = 0$;
return x'.

Algorithm 4 returns the desired outcome as confirmed by the following lemma.

Lemma 1. *Suppose $x \in \mathcal{R}$ with polynomially-many $x_{i,S} > 0$, and $y' \in [0,1]^{I\times J}$ such that $y' \preceq \phi(x)$. If we call Algorithm 4 on x and y', it returns $x' \in \mathcal{R}$ such that $\phi(x') = y'$ with only polynomially-many $x'_{i,S} > 0$.*

Proof. If the algorithm terminates, we will have $\forall i \in I, \forall j \in J$: $\delta_{ij} = 0$, and therefore $y' = \phi(x')$. Thus, we only need to show that the algorithm terminates in polynomial time and x' has polynomially-many positive components. We show the termination of the algorithm for one bin and one item and since the number of items and bins is polynomial, we obtain the desired conclusion.

Fix bin i and item j. We consider one iteration in which $x'_{i,S}$ with $j \in S$ is chosen. Two cases can occur. First, $x'_{i,S} < \delta_{ij}$. In this case, the number of positive components in x' does not increase, since $x_{i,S}$ becomes zero and at most another positive component is added: $x'_{i,S\setminus\{j\}}$. This case can occur as many times as the number of $x_{i,S:j\in S} > 0$, which are polynomially-many by assumption.

Second, $x'_{i,S} \geq \delta_{ij}$. In this case, only one new positive component may be added: $x'_{i,S\setminus\{j\}}$. However, this case can happen only once for item j, as δ_{ij} becomes zero in this step.

Thus, in total for bin i and item j, only one new positive component might be included in x' compared to x and the number of iterations is polynomial. This completes the proof. □

Thus, for the first step of the rounding algorithm, we call Algorithm 4 on inputs x and $y' \in [0,1]^{I\times J}$ where $\forall i \in I, \forall j \in J$: $y'_{ij} = 1 - e^{-y_{ij}}$ and $y = \phi(x)$, to obtain the desired point in $\mathcal{R}$. Notice, that $y' \preceq y$, as needed by Algorithm 4.

The following is a presentation of the greedy rounding algorithm, r_{greedy}.

Algorithm 5: Greedy rounding algorithm, r_{greedy}.

Data: $x \in \mathcal{R}$ with polynomially-many $x_{i,S} > 0$, $v = (v_{ij})_{i\in I, j\in J}$.
Result: Feasible allocation $(S_1, \ldots, S_n)$.
1. Let $y = \phi(x)$. Let $y' \in [0,1]^{I\times J}$ be such that $y'_{ij} = 1 - e^{-y_{ij}}$. Invoke Algorithm (4) with x and y' as the inputs and let x' be the result.
2. Assign set S to i with probability $x'_{i,S}$ independently for each bin i. If some item j is assigned to more than one bin, assign it to the bin among those bins with the maximum value v_{ij}. Let S_i be the set assigned to bin i.
return $(S_1, \ldots, S_n)$.

In order to analyze the performance of the rounding algorithm, we define a new function.

$$F : [0,1]^{I\times J} \to \mathbb{R}_{\geq 0}$$
$$F(y) = \sum_{j=1}^{m}\sum_{i=1}^{n} \left(v_{\sigma_j(i),j} - v_{\sigma_j(i+1),j}\right)\left(1 - \exp(-\sum_{k=1}^{i} y_{\sigma_j(k),j})\right).$$

Where $\sigma_j : I \to I$ is a permutation on I such that $v_{\sigma_j(i),j}$ is decreasing (non-increasing) when i runs from 1 to n, and $v_{\sigma_j(n+1),j} = 0$.

Function $F(\cdot)$ is useful in explaining the quality of the rounding algorithm as shown in the following.

Lemma 2. $\forall x \in \mathcal{R} : \underset{(S_1,\ldots,S_n)\sim r_{greedy}(x)}{\mathbb{E}} \left[\sum_{i\in I} g_i(S_i)\right] = F(\phi(x))$.

Proof. Assume $x \in \mathcal{R}$. Let x' be the outcome of Step 1 of Algorithm 5. Let $y = \phi(x)$ and $y' = \phi(x')$. We calculate the expected value achieved from the assignment of item j in the integral allocation.

Fix item j. For simplicity, we assume that $\sigma_j(i) = i$. This means that bins with smaller indices have higher valuations for j. We find the expected value returned from item j; for other items, the argument is similar. With probability y'_{1j} the set assigned to bin 1 contains j thus j is assigned to 1. Recall that $y'_{1j} = \sum_{S:j\in S} x'_{1,S}$. Therefore, with probability y'_{1j}, the value of returned allocation is v_{1j}. With probability $(1 - y'_{1j})y'_{2j}$ the set assigned to bin 1 does not contain the item but the set assigned to bin 2 contains the item and therefore item j is assigned to bin 2. This case leads to a returned value of $(1 - y'_{1j})y'_{2j}v_{2j}$.

Continuing in a similar manner for other bins, the achievable expected value becomes $y'_{1j}v_{1j} + (1 - y'_{1j})y'_{2j}v_{2j} + \ldots + \prod_{k=1}^{n-1}(1 - y'_{kj})y'_{nj}v_{nj}$, which in turn equals $\sum_{i=1}^{n}(v_{ij} - v_{i+1,j})(1 - \prod_{k=1}^{i}(1 - y'_{kj}))$. The equality of the two terms can be observed by simply extending the latter.

Taking into account that $y'_{ij} = 1 - e^{-y_{ij}}$, by summing over all items we obtain the desired conclusion, using linearity of expectation. □

Therefore, we need to optimize $F(\phi(x))$ over $x \in \mathcal{R}$. Optimizing $F(\phi(x))$ over $x \in \mathcal{R}$ is essentially the same as optimizing $F(y)$ over $y \in \mathcal{P}$, where

$$\mathcal{P} = \left\{ y \in [0,1]^{I \times J} \mid y = \phi(x) \;\&\; x \in \mathcal{R} \right\}.$$

As a result, what remains is to explain how to solve $\max_{y \in \mathcal{P}} F(y)$, and the quality of the solution.

3.3.2 *The Approximation Ratio*

We show the quality of our method by comparing $\max_{y \in \mathcal{P}} F(y)$ to the optimal solution to the configuration LP of GAP. The configuration LP of GAP is as follows:

GAP-CLP:

$$\begin{aligned}
&\max \sum_{i \in I, S \in \mathcal{F}_i} x_{i,S} g_i(S) \\
&\forall j \in J : \sum_{i \in I, S \in \mathcal{F}_i : j \in S} x_{i,S} \leq 1, \\
&\forall i \in I : \sum_{S \in \mathcal{F}_i} x_{i,S} \leq 1, \\
&\forall i \in I, \forall S \in \mathcal{F}_i : x_{i,S} \geq 0,
\end{aligned}$$

To be able to compare GAP-CLP to $F(y)$, we first introduce a new variable into the program and then rearrange the objective function. Let $y \in [0,1]^{I \times J}$ be such that $\forall i \in I, \forall j \in J : y_{ij} = \sum_{S \in \mathcal{F}_i : j \in S} x_{i,S}$. Using this new variable we define polytope $\mathcal{P}'$ as in the following:

$$\begin{aligned}
\mathcal{P}' = \; & \Big\{ y \in [0,1]^{I \times J} \mid \\
& \forall j \in J : \textstyle\sum_{i \in I} y_{ij} \leq 1; \textbf{(1)} \\
& \forall i \in I, \forall j \in J : y_{ij} = \textstyle\sum_{S \in \mathcal{F}_i : j \in S} x_{i,S}; \\
& \forall i : \textstyle\sum_{S \in \mathcal{F}_i} x_{i,S} \leq 1; \\
& \forall i \in I, \forall S \in \mathcal{F}_i : x_{i,S} \geq 0 \Big\}.
\end{aligned}$$

We notice that $\mathcal{P}' \subseteq \mathcal{P}$ since $\mathcal{P}'$ has an additional constraint (Constraint 1). We rearrange the objective function of GAP-CLP to be a function of items (y) rather than subsets, (x).

$$\begin{aligned}
\sum_{i \in I, S \in \mathcal{F}_i} x_{i,S} g_i(S) &= \sum_{i \in I, S \in \mathcal{F}_i} x_{i,S} \sum_{j \in S} v_{ij} \\
&= \sum_{i \in I, j \in J} v_{ij} \sum_{S \in \mathcal{F}_i : j \in S} x_{i,S} \\
&= \sum_{i \in I, j \in J} v_{ij} y_{ij}
\end{aligned}$$

Consequently, solving GAP-CLP is equivalent to finding $\max_{y\in\mathcal{P}'} \sum_{i\in I, j\in J} v_{ij}y_{ij}$. We are now ready to compare $\max_{y\in\mathcal{P}} F(y)$ with the optimal integral solution to the GAP (denoted by OPT).

Lemma 3. $\max_{y\in\mathcal{P}} F(y) \geq (1-\frac{1}{e})OPT$.

Proof. We observe that

$$\max_{y\in\mathcal{P}} F(y) \geq \max_{y\in\mathcal{P}'} F(y) \geq (1-\frac{1}{e}) \max_{y\in\mathcal{P}'} \sum_{i\in I, j\in J} v_{ij}y_{ij} \geq (1-\frac{1}{e})OPT.$$

The first inequality holds since $\mathcal{P}' \subseteq \mathcal{P}$. The last inequality holds because $\max_{y\in\mathcal{P}'} \sum_{i\in I, j\in J} v_{ij}y_{ij}$ in fact provides a solution to GAP-CLP which is obviously greater than OPT. For the second inequality, consider item j and $y \in \mathcal{P}'$. For simplicity, we assume $\forall i : \sigma_j(i) = i$. We have $\sum_{i=1}^{n} y_{ij} \leq 1$, since $y \in \mathcal{P}'$. Considering the fact that $1 - e^{-x} \geq (1-\frac{1}{e})x$ for $x \in [0,1]$, we obtain

$$\begin{array}{ll}
(v_{1j} - v_{2j})(1 - e^{-y_{1j}}) & \geq (v_{1j} - v_{2j})(1-\frac{1}{e})y_{1j} \\
(v_{2j} - v_{3j})(1 - e^{-y_{1j}-y_{2j}}) & \geq (v_{2j} - v_{3j})(1-\frac{1}{e})(y_{1j} + y_{2j}) \\
\ldots & \\
(v_{n-1,j} - v_{nj})(1 - e^{-\sum_{k=1}^{n-1} y_{kj}}) & \geq (v_{n-1,j} - v_{nj})(1-\frac{1}{e})(\sum_{k=1}^{n-1} y_{kj}) \\
(v_{nj})(1 - e^{-\sum_{k=1}^{n} y_{kj}}) & \geq (v_{nj})(1-\frac{1}{e})(\sum_{k=1}^{n} y_{kj})
\end{array}$$

Summing both sides, we obtain

$$\sum_{i=1}^{n} \big(v_{i,j} - v_{i+1,j}\big)\big(1 - \exp(-\sum_{k=1}^{i} y_{k,j})\big) \geq \big(1-\frac{1}{e}\big)\sum_{i\in I} v_{ij}y_{ij}.$$

Obtaining this inequality for all items then, and summing them up, we obtain the desired conclusion. □

Thus, what remains is to show how to maximize $F(y)$ over $y \in \mathcal{P}$, the topic of Section 3.3.3.

3.3.3 *Solving the Convex Optimization Problem*

We wish to solve $\max_{y\in\mathcal{P}} F(y)$ which is essentially equivalent to the following mathematical optimization problem:

GAP-CP:

$$\begin{aligned}
\text{Maximize} & \sum_{j=1}^{m}\sum_{i=1}^{n} \big(v_{\sigma_j(i),j} - v_{\sigma_j(i+1),j}\big)\big(1 - \exp(-\sum_{k=1}^{i} y_{\sigma_j(k),j})\big) \\
& \forall i \in I, \forall j \in J : \sum_{S\in\mathcal{F}_i : j\in S} x_{i,S} = y_{ij}, \\
& \forall i \in I : \sum_{S\in\mathcal{F}_i} x_{i,S} \leq 1, \\
& \forall i \in I, \forall S \in \mathcal{F}_i : x_{i,S} \geq 0.
\end{aligned}$$

First, we show that GAP-CP is a convex optimization problem. All constraints in the program are linear thus we only need to show that the objective function, $F(y)$, is concave/convex as shown by the following theorem.

Lemma 4. *$F(y)$ is a concave function.*

Proof. $F(y)$ is concave in y, because it is a non-negative weighted sum of functions which are compositions of the concave function $1 - e^{-x}$ with affine function $x \to \sum_{k=1}^{i} y_{\sigma_j(k),j}$ (Boyd and Vandenberghe, 2009). □

In order to solve the convex optimization problem, we present a fractional local search algorithm. Our algorithm gets arbitrarily close to the optimal solution. The difficulty in solving the convex optimization problem mostly arises from the exponential number of variables in the convex program. As a result, we are able to implement a $(1-\epsilon)$-MIDR allocation rule, for any $\epsilon = 1/\text{poly}(n)$.

Our algorithm employs a polynomial number of iterations to get as close as a predefined precision to the optimal solution. In every iteration of the algorithm, we need to find $y^* \in \mathcal{P}$ which maximizes $y \cdot \nabla F(y)$ [1] over all $y \in \mathcal{P}$. According to Proposition 1, maximizing $y \cdot v$ over all $y \in \mathcal{P}$ for every cost function v, is equivalent to finding set $S_i^* \in \mathcal{F}_i$ for every bin i which maximizes $\sum_{j \in S_i^*} v_{ij}$.

Proposition 1. $\max_{y \in \mathcal{P}} \sum_{i \in I, j \in J} v_{ij} y_{ij} = \sum_{i \in I} \max\{\sum_{j \in S} v_{ij} : S \in \mathcal{F}_i\}$.

Proof.

$$\begin{aligned}
\max_{y \in \mathcal{P}} \sum_{i \in I, j \in J} v_{ij} y_{ij} &= \max_{x \in \mathcal{R}} \sum_{i \in I, j \in J} \Big(v_{ij} \sum_{S \in \mathcal{F}_i : j \in S} x_{i,S} \Big) \\
&= \max_{x \in \mathcal{R}} \sum_{i \in I} \sum_{S \in \mathcal{F}_i} \Big(x_{i,S} \sum_{j \in S} v_{ij} \Big) \\
&= \sum_{i \in I} \max\{\sum_{j \in S} v_{ij} : S \in \mathcal{F}_i\}.
\end{aligned}$$

The first equality holds since for every $y \in \mathcal{P}$, there exists $x \in \mathcal{R}$ where $y = \phi(x)$. The last equality holds since if $x \in \mathcal{R}$ then $\sum_{S \in \mathcal{F}_i} x_{i,S} \leq 1$. □

Finding $\max\{\sum_{j \in S} v_{ij} : S \in \mathcal{F}_i\}$ is essentially solving a knapsack subproblem for bin i. To do so, we invoke the FPTAS for the knapsack problem. We say for any $v_i = (v_{ij})_{j \in J}$ and $0 < \epsilon < 1$, ***KnapsackFptas***(v_i, ϵ) returns subset $S_i \in \mathcal{F}_i$ where $\sum_{j \in S_i} v_{ij} > (1-\epsilon) \max\{\sum_{j \in S} v_{ij} : S \in \mathcal{F}_i\}$.

We store the computed vector in each iteration in a set $\mathcal{Z}$. We keep the size of $\mathcal{Z}$ to be of at most $\frac{1}{\delta}$ for $\delta = \frac{1}{n}$; δ will be defined later.

1 We remind the reader that ∇F, the gradient of F, is a vector whose coordinates are the first partial derivatives $\frac{\partial F}{\partial y_{ij}}$. We denote by $\frac{\partial F}{\partial y_{ij}}\big|_y$ the gradient coordinate (i,j) evaluated at point y.

As long as $|\mathcal{Z}| < \frac{1}{\delta}$ we simply add the current vector to $\mathcal{Z}$. When $|\mathcal{Z}| = \frac{1}{\delta}$, in each iteration we add the current vector and remove one vector from $\mathcal{Z}$ which has the least value with respect to the current gradient. The solution returned by the algorithm, x, has the property that $y = \phi(x)$ is a convex combination of the vectors in $\mathcal{Z}$: $y = \delta \cdot \sum_{z \in \mathcal{Z}} z$. We continue updating $\mathcal{Z}$ until the increase in $F(y)$ is below a predefined threshold.

Now, we present the main algorithm. Let M denote $\max\{v_{ij} : i \in I; j \in J\}$.

Algorithm 6: Fractional local search algorithm

Data: $v = (v_{ij})_{i \in I, j \in J}$, $0 < \epsilon \leq 1/n$.
Result: $x \in \mathcal{R}$ such that
$F(\phi(x)) \geq (1 - o(1)) \max\{F(y) | y \in \mathcal{P}\}$.
0. Initialize $x := \vec{0}$; $y := \vec{0}$; $\mathcal{Z} = \emptyset$; $\delta = \frac{\epsilon}{6mn^2}$;
1. $u := \nabla F(y)$; $z := \vec{0}$; ; /* $x \in [0,1]^{I \times 2^J}$ and $y, u, z \in [0,1]^{I \times J}$ */
2. **foreach** *bin i* **do**
  Let $S :=$***KnapsackFptas***(u_i, ϵ); for all $j \in S$ update $z_{ij} := 1$;
3. **if** $(z - y) \cdot \nabla F(y) > \epsilon M$ **then**
  if $(|\mathcal{Z}| < \frac{1}{\delta})$ **then**
    Update $y := y + \delta z$; $\mathcal{Z} := \mathcal{Z} \cup \{z\}$;
  else
    Update $y := y + \delta(z - z^{min})$, where $z^{min} = \arg\min_{z' \in \mathcal{Z}} z' \cdot u$;
    $\mathcal{Z} := (\mathcal{Z} \setminus \{z^{min}\}) \cup \{z\}$;
  Go back to Step 1;
4. **foreach** $z \in \mathcal{Z}$ **do**
  foreach *bin i* **do**
    Update $x_{i,S} := x_{i,S} + \delta$, where $S = \{j | z_{ij} = 1\}$;
return x.

Lemma 5. *Algorithm 6 produces a solution x such that $x \in \mathcal{R}$.*

Proof. We observe that the set $\mathcal{Z}$ contains at most $\frac{1}{\delta}$ elements; as long as $|\mathcal{Z}| < \frac{1}{\delta}$, one element z is included into the set and when $|\mathcal{Z}| = \frac{1}{\delta}$, one element is added and one element is removed from the set.

Towards the end of Algorithm 6 (Step 4), for each $z \in \mathcal{Z}$ and each bin i, one positive component ($x_{i,S}$) is increased up to δ which in turn means for each bin i we have $\sum_{S \in \mathcal{F}_i} x_{i,S} \leq \frac{1}{\delta} \cdot \delta = 1$. That means $x \in \mathcal{R}$, the desired conclusion. □

Lemma 6. *Algorithm 6 returns $x \in \mathcal{R}$ such that $F(\phi(x)) \geq (1 - o(1)) \max\{F(y) | y \in \mathcal{P}\}$.*

Proof. Assume x is the outcome of Algorithm 6. According to Lemma 5, $x \in \mathcal{R}$. Let $y = \phi(x)$. Let z be the calculated vector in the last iteration in Step 2, i.e. $(z - y) \cdot \nabla F(y) \leq \epsilon M$.

Let $y^* = \arg\max_{y \in \mathcal{P}} F(y)$. According to Proposition 1, $z \cdot \nabla F(y) \geq (1 - \epsilon) \max_{w \in \mathcal{P}} w \cdot \nabla F(y)$. Hence, $z \cdot \nabla F(y) \geq (1 - \epsilon) y^* \cdot \nabla F(y)$. Thus, we get

$$\begin{aligned} F(y^*) - F(y) &\leq (y^* - y) \cdot \nabla F(y) \\ &\leq \tfrac{1}{1-\epsilon}(z \cdot \nabla F(y) - y \cdot \nabla F(y)) + \tfrac{\epsilon}{1-\epsilon} y \cdot \nabla F(y) \\ &\leq \tfrac{\epsilon}{1-\epsilon} M + \epsilon F(y^*) \end{aligned}$$

The first inequality is because of the concavity of F. The second inequality is by rearranging and using inequality $z \cdot \nabla F(y) \geq (1-\epsilon) y^* \cdot \nabla F(y)$. The third inequality holds since $(z - y) \cdot \nabla F(y) \leq \epsilon M$, and $y \cdot \nabla F(y) < (1-\epsilon) F(y^*)$. If $y \cdot \nabla F(y) \geq (1-\epsilon) F(y^*)$ then, by concavity of F, $F(y) \geq y \cdot \nabla F(y) \geq (1-\epsilon) F(y^*)$, and therefore Lemma 6 holds.

Now, using $\epsilon = \frac{1}{n}$, we obtain $F(y^*) - F(y) \leq \frac{1}{n-1} M + \frac{1}{n} F(y^*) \leq \frac{2}{n-1} F(y^*)$. Hence, when Algorithm 6 terminates $F(y) \geq (1 - o(1)) F(y^*)$, the desired conclusion. □

The change in y in each iteration is either δz or $\delta(z - z^{min})$ for $|\mathcal{Z}| < \frac{1}{\delta}$ and $|\mathcal{Z}| = \frac{1}{\delta}$, respectively. The change in gradient, however, has a certain upper bound when y changes by a certain amount, as Lemma 7 shows.

Lemma 7. *For any y and y' with $||y - y'||_\infty \leq \delta$ and any i and j,*

$$e^{-n\delta} \cdot \left.\frac{\partial F}{\partial y_{ij}}\right|_y \leq \left.\frac{\partial F}{\partial y_{ij}}\right|_{y'} \leq e^{n\delta} \cdot \left.\frac{\partial F}{\partial y_{ij}}\right|_y .$$

Proof. Consider the gradient of F. For simplicity, we assume that $\sigma_j(i) = i$.

$$\frac{\partial F}{\partial y_{ij}} = \sum_{l=i}^{n} (v_{lj} - v_{l+1,j}) \exp(-\sum_{k=1}^{l} y_{kj}). \tag{7}$$

Considering $\sum_{k=1}^{l} y'_{kj} \leq \sum_{k=1}^{n} y_{kj} + n\delta$ and from (7) we obtain $\left.\frac{\partial F}{\partial y_{ij}}\right|_{y'} \geq e^{-n\delta} \cdot \left.\frac{\partial F}{\partial y_{ij}}\right|_y$.

Similarly, from $\sum_{k=1}^{l} y'_{kj} \geq \sum_{k=1}^{n} y_{kj} - n\delta$ and (7), we arrive at $\left.\frac{\partial F}{\partial y_{ij}}\right|_{y'} \leq e^{n\delta} \cdot \left.\frac{\partial F}{\partial y_{ij}}\right|_y$. This completes the proof. □

The following lemma is useful in showing the progress of the algorithm.

Lemma 8. *For any y and y' with $||y - y'||_\infty \leq \delta$,*

$$(y' - y) \cdot \nabla F(y') \geq (y' - y) \cdot \nabla F(y) - 3\delta n^2 m M.$$

Proof. Let $y' - y = z^+ - z^-$ where for all i and j, $0 \leq z^+_{ij} \leq \delta$ and $0 \leq z^-_{ij} \leq \delta$. From Lemma 7, $z^+ \cdot \nabla F(y') \geq e^{-n\delta} z^+ \cdot \nabla F(y)$ and

$z^- \cdot \nabla F(y') \leq e^{n\delta} z^- \cdot \nabla F(y)$. From these inequalities and using inequalities $e^{-x} \geq 1 - x$ and $e^x \leq 1 + 2x$ for $0 \leq x \leq 1$ and $\delta < 1/n$, we get

$$\begin{aligned}(y' - y) \cdot \nabla F(y') &= (z^+ - z^-) \cdot \nabla F(y') \\ &\geq (e^{-n\delta} z^+ - e^{n\delta} z^-) \cdot \nabla F(y) \\ &\geq ((1 - n\delta) z^+ - (1 + 2n\delta) z^-) \cdot \nabla F(y) \\ &= (z^+ - z^-) \cdot \nabla F(y) - n\delta (z^+ + 2z^-) \cdot \nabla F(y) \\ &\geq (y' - y) \cdot \nabla F(y) - 3\delta n^2 m M.\end{aligned}$$

The last inequality holds because for every $z \in [0,1]^{I \times J}$, $z \cdot \nabla F(y) \leq nmM$. This is true since for any y, we have $\frac{\partial F}{\partial y_{ij}} \leq M$ and in the best possible case for z, every bin packs the m items and produces a value of mM. This completes the proof. □

Lemma 9. *In each iteration, the value of $F(y)$ increases by at least $\frac{\epsilon^2}{12mn^2} M$.*

Proof. As long as the algorithm continues we have $(z - y) \cdot \nabla F(y) > \epsilon M$. First, we consider the case where $|\mathcal{Z}| < \frac{1}{\delta}$. We have

$$\begin{aligned}F(y + \delta z) &\geq F(y) + \delta z \cdot \nabla F(y + \delta z) \\ &\geq F(y) + \delta e^{-n\delta} z \cdot \nabla F(y) \\ &\geq F(y) + \delta e^{-n\delta} \epsilon M\end{aligned}$$

The first inequality is because of the concavity of F. The second inequality holds because of Lemma 7. The third inequality is because $(z - y) \cdot \nabla F(y) > \epsilon M$ implies that $z \cdot \nabla F(y) > \epsilon M$, as we always have $\nabla F(y) \geq \vec{0}$.

Now, using $\delta = \frac{\epsilon}{6mn^2}$, we obtain

$$F(y + \delta z) \geq F(y) + \frac{\epsilon^2}{12mn^2} M.$$

Second, we consider the case where $|\mathcal{Z}| = \frac{1}{\delta}$. We have

$$\begin{aligned}F(y + \delta(z - z^{min})) &\geq F(y) + \delta(z - z^{min}) \cdot \nabla F(y + \delta(z - z^{min})) \\ &\geq F(y) + \delta(z - z^{min}) \cdot \nabla F(y) - 3\delta^2 n^2 m M \\ &\geq F(y) + \delta \epsilon M - 3\delta^2 m n^2 M\end{aligned}$$

The first inequality is because of the concavity of F. The second inequality holds because of Lemma 8. The third inequality is because $(z - z^{min}) \cdot \nabla F(y) \geq (z - y) \cdot \nabla F(y)$, as shown in the following.

By definition of z^{min}, $z^{min} \cdot \nabla F(y) \leq z' \cdot \nabla F(y)$ for all $z' \in \mathcal{Z}$. Thus, $|\mathcal{Z}| \cdot z^{min} \cdot \nabla F(y) \leq \sum_{z' \in \mathcal{Z}} z' \cdot \nabla F(y)$, which in turn means $z^{min} \cdot \nabla F(y) \leq y \cdot \nabla F(y)$. Observe that $y = \delta \cdot \sum_{z' \in \mathcal{Z}} z'$.

Now, using $\delta = \frac{\epsilon}{6mn^2}$, we obtain

$$F(y + \delta(z - z^{min})) \geq F(y) + \frac{\epsilon^2}{12mn^2} M.$$

This completes the proof. □

Lemma 10. *After at most* $12m^2n^2/\epsilon^2$ *iterations, Algorithm 6 terminates.*

Proof. Since M denotes $\max\{v_{ij} : i \in I; j \in J\}$, mM is an upper bound for $\max_{y\in\mathcal{P}} F(y)$. Recall that

$$\max_{y\in\mathcal{P}} F(y) = \max_{x\in\mathcal{R}} \mathop{\mathbb{E}}_{(S_1,\ldots,S_n)\sim r_{\text{greedy}}(x)} \Big[\sum_{i\in I} g_i(S_i)\Big] \leq mM.$$

Based on Lemma 9, in each iteration the growth in value is at least $\frac{\epsilon^2}{12mn^2}M$, Algorithm 6 thus in at most $12m^2n^2/\epsilon^2$ iterations, reaches the value of mM, which is an upper bound on the best solution. This concludes the proof. □

We thus achieve a $(1-\epsilon)$-MIDR allocation rule that runs in polynomial time. This concludes the proof of Theorem 2.

3.3.4 *Simplifying the Rounding Procedure*

As we note, it is possible to simplify the rounding procedure (Algorithm 5), further. The simplified rounding is as follows.

Given $x \in \mathcal{R}$, let $y = \phi(x)$. We assign set S to each bin i independently with probability $x_{i,S}$. Next, for each item j we do as follows. If item j is assigned to bin i, we let the bin hold the item with probability $\frac{1-e^{-y_{ij}}}{y_{ij}}$. This means, we withdraw the item from the bin with the complementary probability $1 - \frac{1-e^{-y_{ij}}}{y_{ij}}$ to make sure that the probability of assigning item j to bin i is not y_{ij} but $1 - e^{-y_{ij}}$ which is necessary to maintain the MIDR property. According to the MIDR principle, the expected value of the randomized integral assignment should equal the calculated fractional value. Finally, if some item j is assigned to more than one bin, we assign it to the bin among those bins with the maximum value v_{ij}.

In order to use the allocation rule algorithm (Algorithm 3) as an optimization algorithm, one can employ a more simple rounding algorithm. The simpler rounding requires only Step 2 of Algorithm 5. For an optimization purpose, there is no need to also execute Step 1 of Algorithm 5. Thus, after finding a fractional solution x by invoking Algorithm 6, we assign set S to each bin i with probability $x_{i,S}$ and resolve conflicts according to the technique explained in Algorithm 5. This improves the runtime for the optimization purpose.

3.4 COMPUTING PAYMENTS

Supplementing the MIDR allocation rule of Section 3.3 with VCG payments yields a truthful-in-expectation mechanism. We compute payments in order to also enforce non-negativity of payments and individual rationality, *ex post*.

To compute the VCG fractional payment p_i^{frac} for bidder i, we need to compute two components: first, the Clarke pivot, $h_i(v_{-i})$, which is the best achievable social welfare by bidders other than i, and second, the value gained by bidders other than bidder i in the current fractional solution. We can calculate $h_i(v_{-i})$ by rewriting GAP-CP for the market without bidder i, i.e. $v_{ij} = 0$ for all j. To compute the value gained by other bidders in the fractional allocation, $F_{-i}(y^*)$, we set $\forall j \in J : v_{ij} = 0$ in $F(y^*)$, assuming that y^* is the outcome of Algorithm 6. Function $F(y)$ is explicitly known to us and we can set in it v_{ij} to 0. Finally, $p_i^{frac} = h_i(v_{-i}) - F_{-i}(y^*)$.

Example 1. *Consider a setting in which two bidders (1 and 2) have valuations for two items as follows:* $v_{11} = 8, v_{12} = 5$ *and* $v_{21} = 4, v_{22} = 10$. *In this case,*

$$F(y) = (8-4)(1-e^{-y_{11}}) + 4(1-e^{-y_{11}-y_{21}}) + (10-5)(1-e^{-y_{22}}) + 5(1-e^{-y_{12}-y_{22}}).$$

Now, assume $y_1^* = (0.6, 0.3)$ *and* $y_2^* = (0.4, 0.7)$. *Then,*

$$F_{-1}(y^*) = (0-4)(1-e^{-0.6}) + 4(1-e^{-1}) + (10-0)(1-e^{-0.7}) + 0(1-e^{-1}).$$

The value gained by bidder i in the fractional allocation is therefore $w_i^{frac} = F(y^*) - F_{-i}(y^*)$. Assuming that S_i is the subset assigned to bidder i by the rounding procedure, we can compute the randomized payment for bidder i, p_i, satisfying individual rationality and non-negativity of payments as follows.

$$p_i = \begin{cases} \frac{g_i(S_i)}{w_i^{frac}} p_i^{frac} & \text{if } w_i^{frac} > 0, \\ 0 & \text{if } w_i^{frac} = 0. \end{cases}$$

3.5 TRUTHFULNESS

It has been proven that if the allocation algorithm of a mechanism is maximal-in-range and the payment rule calculates the payments by applying the same payment idea as in VCG, then the mechanism is truthful (Nisan and Ronen, 2007). Mechanisms using the idea of VCG to calculate payments are termed VCG-based mechanisms.

Obviously, our mechanism is a VCG-based mechanism as we calculate the payments following the idea of VCG. In order to show truthfulness, it suffices to show that the allocation algorithm is maximal-in-range. Since our mechanism is randomized, we need to show that the allocation rule is maximal in distributional range. To do so, we describe the *distributional range* of allocations over which the allocation rule optimizes social welfare to optimality. By MIDR definition, the range of allocations has to be chosen before any valuation has been seen.

Let Π denote the set of all permutations on all bins. The rounding algorithm (Algorithm 5) actually works with a specific permutation on bins for each item. In particular, the rounding algorithm uses σ_j for each item j which reorders the bins in decreasing order of their values for item j. We recall that σ_j is formally defined in Subsection 3.3.1.

We can look at the rounding algorithm as a function which takes the permutation π_j for each item j as input. Let us call this parameterized rounding algorithm r and define it as Algorithm 7.

Algorithm 7: Rounding algorithm, r.

Data: $x \in \mathcal{R}$, $\pi_j \in \Pi$ for each item j.
Result: Feasible allocation $(S_1, \ldots, S_n)$.
1. Let $y = \phi(x)$. Let $y' \in [0,1]^{I \times J}$ be such that $y'_{ij} = 1 - e^{-y_{ij}}$. Invoke Algorithm (4) with x and y' as the inputs and let x' be the result.
2. Independently for each bin i, assign set S to i with probability $x'_{i,S}$. If some item j is assigned to more than one bin, then *assign item j to the bin among those bins that precedes others in π_j*.
return $(S_1, \ldots, S_n)$.

Algorithm 5 thus can be rewritten as $r_{\text{greedy}}(x) = r(x, \sigma_1, \sigma_2, \ldots, \sigma_m)$.

For each point $x \in \mathcal{R}$, $r(x, \pi_1, \ldots, \pi_m)$ rounds point x to an integer point by taking into account permutation $\pi_j \in \Pi$ for each item j. We let the *domain* of function r comprise all $x \in \mathcal{R}$ as well as all $\pi_j \in \Pi$, $\forall j \in J$. The *range* of function r then is the range over which our allocation algorithm optimizes the social welfare. Formally, we define the range as follows.

$$\textit{Range} \equiv \bigcup_{x \in \mathcal{R}, \pi \in \Pi^J} \{r(x, \pi)\}.$$

The range is clearly independent of private values as it only takes into account points x and the permutations. Maximizing over this range defines a maximal in distributional range algorithm. In order to maximize over the range we don't need to search the full range. Rather, it suffices to look for the maximum only in the part of the range containing the maximum.

We utilize this fact in our MIDR. In particular, in Algorithm 3 we maximize $r(x, \sigma_1, \ldots, \sigma_m)$ over $x \in \mathcal{R}$. There is no need to take into account other permutations since the value of $r(x, \sigma_1, \ldots, \sigma_m)$ is always as high as that of $r(x, \pi_1, \ldots, \pi_m)$ where at least for one j $\pi_j \neq \sigma_j$. To rephrase, maximizing $r(x, \sigma_1, \ldots, \sigma_m)$ over $x \in \mathcal{R}$ is equivalent to maximizing $r(x, \pi_1, \ldots, \pi_m)$ over $x \in \mathcal{R}$ for all $\pi_j \in \Pi$, $j \in J$. This is true because for each item j when there is a tie (Step 2 of Algorithm 7) assigning the item to the bin with the highest value for the item, obviously produces a higher value. To the best our knowledge, this is

the first time that such an observation has been used for maximizing over a range.

3.6 STRATEGIC ITEMS

Here, we mention the required modifications for the case where items are held by bidders rather than the bins. Such bidders are called unit-demand bidders. To better expose the changes we index bidders by j in the following. The allocation rule $\mathcal{A}$ takes reported valuations $v = (v_1, \ldots, v_m)$, $v_j = (v_{ij})_{i \in I}$ for all $j \in J$ and returns $(i_1, i_2, \ldots, i_m)$ where i_j is the bin to which item j is assigned. Let p_j denote the payment rule function for bidder j. We use $v_j(i_1, \ldots, i_j, \ldots, i_m)$ instead of v_{ij} for the sake of simplicity in the definition below.

Definition 10 (truthful-in-expectation)**.** *A mechanism is truthful-in-expectation for GAP (when items are held by bidders) if, for every bidder j, (true) valuation function v_j , (reported) valuation function v'_j, and (reported) valuation functions v_{-j} of the other bidders,*

$$\mathbb{E}[v_j(\mathcal{A}(v_j, v_{-j})) - p_j(v_j, v_{-j})] \geq \mathbb{E}[v_j(\mathcal{A}(v'_j, v_{-j})) - p_j(v'_j, v_{-j})]. \quad (8)$$

The expectation in (8) is taken over the coin flips of the mechanism.

For this type of bidders, we maximize over the range defined in Section 3.5 to obtain a MIDR. That is, we use Algorithm 3 as the allocation rule. We need however a different payment rule for this type of bidders.

In order to calculate VCG fractional payment p_j^{frac}, we need to calculate Clarke pivot $h_j(v_{-j})$ and the value gained by bidders other than j in the current fractional solution, $F_{-j}(y^*)$.We calculate $h_j(v_{-j})$ by running Algorithm 6 after evaluating v_{ij} to 0 for all $i \in I$. Also, we have $F_{-j}(y^*) = \sum_{l=1, l \neq j}^{m} \sum_{i=1}^{n} \left(v_{\sigma_l(i),l} - v_{\sigma_l(i+1),l}\right)\left(1 - \exp(-\sum_{k=1}^{i} y^*_{\sigma_l(k),l})\right)$. By letting $w_j^{frac} = F(y^*) - F_{-j}(y^*)$ and assuming that item j is assigned to bin i in the rounded solution, the payment of bidder j is calculated as follows.

$$p_j = \begin{cases} \frac{v_{ij}}{w_j^{frac}} p_j^{frac} & \text{if } w_j^{frac} > 0, \\ 0 & \text{if } w_j^{frac} = 0. \end{cases}$$

3.7 CONCLUSION

We studied the problem of mechanism design for a strategic variant of GAP where valuations are assumed to be private information known only to the bidders while weights and capacities are publicly known. Given that GAP is NP-hard, and that VCG is not trivially truthful with suboptimal solutions, we resorted to approximation mechanisms.

We proposed a solution by which the two obstacles, maximizing the social welfare of GAP as well as extracting true valuations of bidders, are surmounted. The solution provides bidders with incentives to report their valuations truthfully and runs in polynomial time approximating the social welfare with a provable ratio of at least $1 - 1/e$.

In comparison to the approximation algorithms presented for GAP without incentive issues, our proposed algorithm has advantages in terms of runtime and simplicity while presenting the same approximation ratio. Our work also shows that the convex rounding technique is a powerful machinery for designing truthful approximation mechanisms and might find other applications in the field.

A problem which remains to be solved is the analysis of the strategic version of GAP in which weights and capacities are also private. We conjecture there is no constant ratio truthful mechanism for this problem.

Another problem to be solved is to find a truthful mechanism for GAP with private values which stipulates that no item in the final allocation may remain unassigned.

4

MECHANISM DESIGN WITHOUT MONEY

In this chapter, we study a problem of truthful mechanism design for a strategic variant of the generalized assignment problem (GAP) in a both payment-free and prior-free environment. In GAP, a set of items has to be optimally assigned to a set of bins without exceeding the capacity of any singular bin. In the strategic variant of the problem we study, bins are held by strategic agents, and each agent may hide its compatibility with some items in order to obtain items of higher values. The compatibility between an agent and an item encodes the willingness of the agent to receive the item. Our goal is to maximize total value (sum of agents' values, or social welfare) while certifying no agent can benefit from hiding its compatibility with items. The model has applications in auctions with budgeted bidders. For two variants of the problem, namely *multiple knapsack problem* in which each item has the same size and value over bins, and *density-invariant GAP* in which each item has the same value density over the bins, we propose truthful 4-approximation algorithms. For the general problem, we propose an $O(\ln(U/L))$-approximation mechanism where U and L are the upper and lower bounds for value densities of the compatible item-bin pairs. [1]

4.1 INTRODUCTION

Truthful mechanism design without money under general preferences is a classic topic in social choice theory. Truthfulness ensures that no agent can be better off by manipulating its true preferences. When searching for truthful mechanisms *without money*, one has to look at restricted domains of preferences. The reason for this, is the Gibbard-Satterthwaite theorem which states that any truthful social choice function which selects an outcome among three or more alternatives has to be trivially aligned with the preference of a single agent (namely, the dictator) (Gibbard, 1973; Satterthwaite, 1975). Thus, exploring domains for which there exist truthful mechanisms is of central importance in the field of social choice theory.

1 In this chapter, for the sake of simplicity, an α-approximation means a solution whose value is at least $1/\alpha$ times the optimal total value.

For example, when valuations of agents are restricted to single-peaked preferences over a one-dimensional public space, returning the *median* of the peaks determines a truthful social choice (Moulin, 1980). Another example is the two-sided matching, in which a set of men has a strict preference ordering over a set of women, and vice versa. A matching is an assignment of men to women where each side is assigned to only one element of the other side. The *deferred acceptance algorithm* finds a stable matching which is truthful for the proposing side, but not necessarily truthful for the other side (Roth and Sotomayor, 1992).

One way to circumvent the impossibility result is relaxing the social choice function. Procaccia and Tennenholtz introduced the technique of welfare approximation as a means to derive truthful approximation mechanisms without money (Procaccia and Tennenholtz, 2013). This type of approximation is not meant to handle computational intractability, but a method to achieve truthfulness by relaxing the goal of optimizing social welfare (approximating social welfare), and thus circumventing the Gibbard-Satterthwaite impossibility theorem. The approach is to maximize welfare without considering incentives, and refer to this as optimal value. Then it is said that a truthful mechanism returns (at most) an α-approximation of the optimal if its value is always greater than or equal to $1/\alpha$ times the optimal value ($\alpha \geq 1$). Several works, subsequent to the work of Procaccia and Tennenholtz, employ this technique (Dughmi and Ghosh, 2010; Chen et al., 2013; Koutsoupias, 2014). We apply this technique to a restricted strategic setting defined below.

4.1.1 *Model*

Consider a strategic variant of the generalized assignment problem termed GAP-BS in a both prior-free and payment-free environment. In GAP-BS, there are m items J and n bins (knapsacks) I. Each bin i has a capacity C_i and associates a value v_{ij} and a size w_{ij} to any item j. A feasible assignment may allocate a subset of items S to bin i such that $\sum_{j \in S} w_{ij} \leq C_i$. A feasible assignment may assign each item at most once.

In GAP-BS, we assume tuple $T = (\{v_{ij}\}_{ij}, \{w_{ij}\}_{ij}, \{C_i\}_i)$ is public, but each bin is held by a strategic agent. The private information that each agent/bin holds is the set of its compatible items. The compatibility between an agent and an item encodes the willingness of the agent to receive the item. In particular, consider a bipartite graph G where one side corresponds to items and the other side corresponds to the bins. The edges of G, $E \subseteq I \times J$ represent the compatible item-bin pairs. The private type of a bin i is therefore the set of edges in the graph incident on i, E_i. A bin i receives value $v_i(S) = \sum_{j \in S:(i,j) \in E} v_{ij}$ from package S if $\sum_{j \in S} w_{ij} < C_i$ and 0, otherwise. The total value

of a feasible assignment $(S_1, S_2, \ldots, S_n)$ equals the sum of values received by the bins from the assignment: $\sum_{i \in I} v_i(S_i)$. We seek a total value-maximizing algorithm that provides each bin i with incentives to truthfully report its compatible items E_i rather than any $E'_i \subset E_i$.[2] In other words, given a truthful mechanism, bins have no incentive to hide their compatibility with some items.

Let $\mathcal{A}$ denote a randomized algorithm which takes instance (T, E) and computes $X \in \{0,1\}^E$, an assignment of items to bins. Notice, the assignment itself is a deterministic assignment (each bin receives a deterministic set of items), but algorithm $\mathcal{A}$ is internally randomized, i.e., $\mathcal{A}$ returns a solution which is randomly chosen according to a probability distribution over feasible assignments. Thus, the computed assignment may change by running $\mathcal{A}$, twice on the same input. Randomized algorithm $\mathcal{A}$ is said to be truthful-in-expectation (truthful, henceforth unless mentioned explicitly) if for $X \sim \mathcal{A}(T, E)$, we have

i. (feasibility) $\forall j \in J$, $Pr[\sum_{i \in I} X_{ij} \leq 1] = 1$ and $\forall i \in I$, $Pr[\sum_{j \in J} w_{ij} X_{ij} \leq C_i] = 1$

ii. (truthfulness) for any i and $E'_i \subset E_i$, we have $\mathbb{E}[\sum_{j \in J:(i,j) \in E} v_{ij} X_{ij}] \geq \mathbb{E}[\sum_{j \in J:(i,j) \in E} v_{ij} X'_{ij}]$, where $X' \sim \mathcal{A}(T, E'_i \cup E_{-i})$.

E_{-i} always denotes $E \setminus E_i$. Note that, the expected value of the bin in both cases is calculated with respect to true item-bin compatibilities, E. To sum, our objective is to propose a randomized algorithm $\mathcal{A}$ for GAP-BS which is truthful, and always returns a feasible assignment whose value approximates the optimal total value as high as possible.

Many real-world decision problems can be modeled by variants of knapsack problems, therefore we believe that our model can be applied broadly. As an example, we refer to the *maximum budgeted allocations* (MBA) problem (Chakrabarty and Goel, 2010). In MBA, a set of indivisible items has to be assigned to a set of bidders. Each bidder i reports her willingness to pay b_{ij} for item j by bidding for the item, while she has a budget constraint B_i. Each bidder i on receiving a package S of items, pays $\sum_{j \in S} b_{ij}$. Each bidder i has the rigid constraint B_i on her payment. The goal in MBA is to find a distribution of items among the bidders which maximizes the total revenue (the sum of the payments by the bidders while respecting their budget constraints). MBA arises in auctions with budgeted bidders and has several applications (Chakrabarty and Goel, 2010).

In MBA, bidders want to get as much as they can without spending more than their budget. For instance, advertisers wish to maximize the impressions, clicks, or sales generated by their advertising,

2 In fact, our results certify that each bin i reports exactly E_i and has no incentives to report any other set of edges E'_i. However, for the sake of simplicity in the exposition of the results, we focus on untruthful reports that are made by hiding some edges, $E'_i \subset E_i$.

subject to budget constraints. Similarly, bidders who have no direct utility for leftover money (e.g. because the money comes from a corporate budget) will buy as much as possible. This types of bidders are called *value maximizers*, and have recently drawn the attention of researchers in mechanism design (Wilkens et al., 2016; Cavallo and Krishnamurthy, 2015).

Consider a strategic variant of MBA in which each bidder, in order to obtain a more valuable package of items, strategizes in the following way. Each bidder may strategically hide her interests in buying some items by not bidding for those items. In this setting, the auctioneer wishes to certify that each bidder truthfully reveals her willingness to buy items. In other words, a truthful mechanism in this setting will encourage participation of the bidders in the auction.

We model this setting by GAP-BS, in which each bidder is represented by a bin, budgets B_i by capacities C_i, the bids b_{ij} by the values of bins for the items v_{ij}, and the payment by a bidder i for item j by the weight of the item on the bin, w_{ij}. Thus, in this setting of GAP-BS, we have $v_{ij} = w_{ij}$ for all i and j. For this problem, since the value density of each item is the same over all bins, we provide a truthful 4-approximation algorithm.

4.1.2 *Discussion About the Assumptions*

Aside from the applications of the model discussed above, we emphasize that our assumptions (which imply a highly structured domain) are necessary to escape the impossibility results such as the Gibbard-Satterthwaite theorem and its variations (Barbera and Peleg, 1990). For example, we resort to welfare approximations because as stated by Theorem 4, no deterministic (or randomized) algorithm whose value is optimal, exists for GAP-BS. The lower bounds in Theorem 4 were also derived for a different setting with strategic items (Dughmi and Ghosh, 2010), however, here we reproduce and adapt the theorem for our setting.

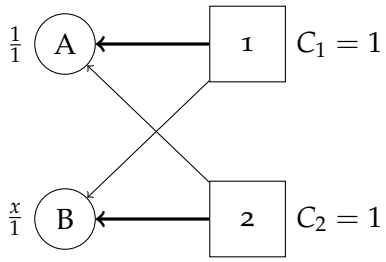
Theorem 4. *No truthful deterministic algorithm with an approximation ratio better than* 2 *exists for GAP-BS. Moreover, no truthful-in-expectation randomized algorithm with an approximation ratio better than* 1.09 *exists for GAP-BS.*

Proof. Consider a small market with two bins and two items shown in Figure 1 (a). In this market, bins have capacity 1, and are both compatible with the two items. Item B is more valuable to both bins ($x > 1$), but each item has size 1. This market can be viewed as an instance for both the multiple-knapsack problem (Subsection 4.2.1), and the density-invariant GAP (Subsection 4.2.2).

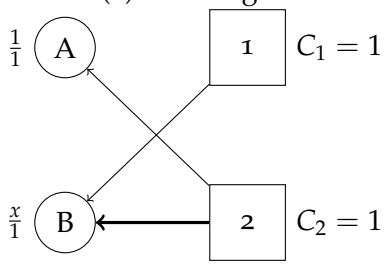
Regarding deterministic mechanisms, an arbitrary truthful mechanism has to assign item B to one bin. Without loss of generality,

assume B is assigned to bin **2**, i.e., the tie is broken deterministically (alphabetically) in favor of bin **2**. Now, consider reports in (b) of Figure 1. The mechanism, in case (b), cannot assign B to bin **1** as it violates truthfulness. Thus, the mechanism assigns B to **2**, and this results in an approximation ratio of $\frac{x+1}{x}$ which tends to 2 when x gets very close to 1.

An arbitrary truthful-in-expectation mechanism, in case (a), assigns B to one bin with a probability less than or equal $1/2$. Without loss of generality, let bin **1** be that bin. The utility of bin **1**, in this case, will be at most $\frac{x+1}{2}$ for $x > 1$. Assume, in case (b), item B is assigned to bin **1** with probability q, resulting a $q \cdot x$ expected value for bin **1**. Truthfulness stipulates no increase in the utility of bin **1** in case (b), i.e., $\frac{x+1}{2} \geq q \cdot x$, thus $q \leq \frac{x+1}{2x}$. In case (b), the total expected value will be $q(1+x) - (1-q)x = x + q$, thus the approximation ratio will be $\frac{x+1}{x+q}$. In order to obtain a smaller approximation ratio, we plug in $q = \frac{x+1}{2x}$. The ratio gets a value of $1 + \frac{1}{4\sqrt{2}+5} \approx 1.094$ for $x = 1 + \sqrt{2}$, the desired conclusion.



(a) True edges.



(b) Manipulated edges.

Figure 1: Circles represent items and squares represent bins. The $\frac{\text{value}}{\text{size}}$ of each item is on its left. Each bin has a capacity of 1. Selected assignments are in bold.

□

Now, we consider a setting in which bins/agents have private values for items. This setting is more general than GAP-BS in that, in this setting, the agents can manipulate their valuations for items. This is in contrast to GAP-BS in which the agents can only hide their valuations for some items by hiding their compatibility with those items. For this general setting, no deterministic (or randomized) truthful algorithm, with an interesting approximation ratio, exists. To see this,

consider a simple market with one item, and a set of agents. This market is equivalent to the single-item auction, but without money. We observe that no mechanism without money can find the (true) highest valuation for the item, as the agents can report arbitrarily high values for the item. That is, no truthful algorithm can do any better than the algorithm which allocates the item to the bin which is uniformly chosen at random. Such an algorithm provides a trivial approximation ratio of $1/n$, n being the number of agents.

In a parallel setting, Dughmi and Ghosh (2010) and Chen et al. (2013) studied the generalized assignment problem in an environment in which transfer of money is not allowed. While the setting proposed by these authors is close to ours, the main difference between the two settings is that in their setting, *items* are held by strategic agents, but in our model *bins* are held by the agents. Consequently, the solutions proposed by these authors are not directly applicable to GAP-BS, however in our solutions to GAP-BS we benefit from the techniques developed therein.

4.1.3 *Results and Technique*

In addition to GAP-BS, we also analyze two variants, namely the multiple knapsack problem in which each item has the same size and value over bins, and density-invariant GAP in which each item has the same value density (value per size) over the bins.

We observe that the relaxation and rounding technique is applicable to these problems. The relaxation and rounding technique is a welfare approximation technique (Procaccia and Tennenholtz, 2013) based on linear programming relaxations. To apply the technique, we start with a linear programming relaxation of the problem. Then, we need an algorithm which returns a fractional solution to the relaxation with an acceptable approximation ratio. The algorithm has to be fractionally truthful, i.e., no agent can increase its fractional value by untruthful reports. Finally, a rounding scheme which preserves truthfulness is applied to the fractional solution to obtain an integer solution. It should be noted that the relaxation and rounding technique has been previously applied to mechanism design without money in a different setting (Dughmi and Ghosh, 2010).

We apply the technique successfully to our problems by proposing fractionally truthful algorithms with acceptable approximation ratios. For the rounding scheme, we use a rounding method called *randomized meta-rounding*, originally proposed by Carr and Vempala (2000), and later applied by Lavi and Swamy (2011) to mechanism design (with quasi-linear valuations). Using the relaxation and rounding technique, for two variants of GAP-BS, the multiple knapsack problem, and density-invariant GAP, we propose truthful 4-approximation algorithms. For GAP-BS, we show an $O(\ln(U/L))$-approximation mech-

anism where U and L are the upper and lower bounds for value densities of the compatible item-bin pairs.

4.2 GENERALIZED ASSIGNMENT PROBLEM

We start with a linear programming relaxation of GAP-BS.

$$\begin{aligned} \text{Maximize} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} v_{ij} x_{ij} && \text{(LP[E])} \\ \text{subject to} \quad & \textstyle\sum_{i=1}^{n} x_{ij} \leq 1 \quad \forall j \in J \\ & \textstyle\sum_{j=1}^{m} w_{ij} x_{ij} \leq C_i \quad \forall i \in I \\ & x_{ij} \geq 0 \quad \forall i, j \\ & x_{ij} = 0 \quad \forall (i,j) \notin E. \end{aligned}$$

Our technique is as follows. We design a *fractionally truthful* approximation algorithm which returns a feasible solution to LP[E]. A fractionally truthful algorithm allocates fractional assignments to bins, and no bin can improve its fractional value by an untruthful report. In particular, a fractionally truthful algorithm $\mathcal{A}^F$ takes (T, E) and returns $x \in [0,1]^E$, a feasible solution to LP[E] with the following property. For each bin i, if the bin reports $E_i' \subset E_i$, we will have $\sum_{j:(i,j)\in E} v_{ij} x_{ij} \geq \sum_{j:(i,j)\in E} v_{ij} x'_{ij}$, where $x' = \mathcal{A}^F(T, E_i' \cup E_{-i})$ and $E_{-i} = E \setminus E_i$. Next, we round the fractional solution using a special rounding technique which makes sure that each bin obtains a fixed fraction of its fractional value in expectation. The *randomized meta-rounding* is capable of maintaining this fixed fraction.

To use the randomized meta-rounding, we have to scale down the fractional solution by factor 2, which is essentially the integrality gap of the LP[E] (Shmoys and Tardos, 1993). Assuming $x^* = \mathcal{A}^F(T, E)$, the randomized meta-rounding represents $x^*/2$ as a convex combination of polynomially-many feasible integer solutions. Looking at the provided convex combination as a probability distribution over integer solutions, we sample a randomized solution X which is always feasible, and its expected value is $1/2$ of the fractional value of x^*. This is confirmed by Theorem 5.

Theorem 5. *If there exists a fractionally truthful α-approximation algorithm for GAP-BS, then there exists a truthful (2α)-approximation solution for GAP-BS.*

Proof. Let $\mathcal{A}^F$ denote a fractionally truthful algorithm for GAP-BS that takes an instance (T, E) and returns a feasible solution to LP[E]. Let x^* be the outcome of $\mathcal{A}^F$ on instance (T, E). Let $\{X^l\}_{l\in L}$ denote the set of feasible integer solutions to LP[E], where L indexes all feasible integer solutions. The integrality gap of LP[E] equals 2 (Shmoys and Tardos, 1993), thus we scale down the fractional solution by factor 2. The meta-randomized rounding applied to $x^*/2$ computes a prob-

ability distribution over feasible integer solutions whose support is polynomial (Carr and Vempala, 2000; Lavi and Swamy, 2011):

$$\frac{x^*}{2} = \sum_{l \in L} \lambda_l X^l, \quad \sum_{l \in L} \lambda_l = 1, \text{ and } \quad \forall l \in L, \lambda_l \geq 0.$$

We treat the convex decomposition above as a probability distribution according to which solution X^l has probability λ_l of being selected. Let X be a solution sampled from the above distribution. Obviously X is feasible by the construction of the distribution. We also have $\mathbb{E}[X_{ij}] = \frac{1}{2}x^*_{ij}$ for all i and j from the construction of the distribution. By the linearity of expectation, the expected value of a bin is $\mathbb{E}[\sum_{j:(i,j)\in E} v_{ij}X_{ij}] = \frac{1}{2}\sum_{j:(i,j)\in E} v_{ij}x^*_{ij}$. Therefore, the expected value of the solution is exactly half of the value of the fractional solution.

For truthfulness, fix bin i and $E_{-i} = E \setminus E_i$. Suppose the bin reports $E'_i \subset E_i$ rather than E_i. Let $x' = \mathcal{A}^F(T, E'_i \cup E_{-i})$, and X' be the solution returned by the meta-randomized rounding from $x'/2$. We have

$$\begin{aligned}\mathbb{E}[\textstyle\sum_{j:(i,j)\in E} v_{ij}X_{ij}] &= \tfrac{1}{2}\textstyle\sum_{j:(i,j)\in E} v_{ij}x^*_{ij} \\ &\geq \tfrac{1}{2}\textstyle\sum_{j:(i,j)\in E} v_{ij}x'_{ij} \\ &= \mathbb{E}[\textstyle\sum_{j:(i,j)\in E} v_{ij}X'_{ij}]\end{aligned}$$

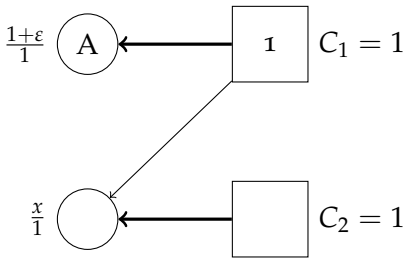
The inequality is because $\mathcal{A}^F$ is fractionally truthful. Therefore, the bin cannot improve its expected value by hiding some of its edges. This completes the proof. □

4.2.1 *Multiple Knapsack Problem*

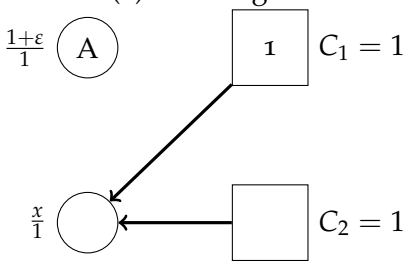
We consider a variant of GAP-BS in which neither the size nor the value of each item depends on the bins. Formally, for each item j we have $v_{ij} = v_j$ and $w_{ij} = w_j$ for all bins i. First, we observe an algorithm that returns a (fractional) optimal solution to LP[E] is not fractionally truthful. This can be seen in the example shown in Figure 2. In (a) of this figure, the edges are reported truthfully, and the value-maximizing allocation, assigns A to bin $\mathbf{1}$ and the other item to the other bin. In (b) of this figure, bin $\mathbf{1}$ hides its compatibility with item A and as a consequence it is better off (in expectation) when the mechanism maximizes the total value. In (b) of this figure, the tie can be broken randomly or deterministically (alphabetically) in favor of bin $\mathbf{1}$. In any case, bin $\mathbf{1}$ is better off by manipulation.

We propose Algorithm 8. We choose bin i in an arbitrary order and (fractionally) assign compatible items to it according to the decreasing order of value densities of items v_j/w_j until the capacity of the bin is exhausted or all compatible items are exhausted. Then we proceed to the next bin with remaining (fractional) items.

Algorithm 8 is fractionally truthful. It is well known that assigning items according to decreasing order of value densities, when fractional assignments are allowed, produces the highest fractional value



(a) True edges.



(b) Manipulated edges.

Figure 2: Circles represent items and squares represent bins. The value/size of each item is on its left. Value maximizing assignments are in bold. $x \gg 1$.

Algorithm 8: Multiple Knapsack Problem.

1. Sort items according to the decreasing order of value densities v_j/w_j, breaking ties arbitrarily.
2. **foreach** *bin i chosen in an arbitrary order* **do**
   For each unassigned (fractional) item j where $(i, j) \in E$ in the order defined above, fractionally assign as much of the item to bin i until the item is exhausted or the bin is full.

return the resulting assignment x.

for the bin. Since bins wish to maximize their values and the algorithm is aligned with this goal, thus the bins have no incentive to lie, and the algorithm is fractionally truthful.

With regard to the total value, we show that Algorithm 8 returns a 2-approximate fractional solution. We compare the outcome of the algorithm with the optimal solution to the LP formulation of the problem shown below.

$$\begin{aligned}
\text{Maximize} \quad & \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} v_j x_{ij} && \text{(MKP-LP[E])} \\
\text{subject to} \quad & \textstyle\sum_{i=1}^{n} x_{ij} \leq 1, && \forall j \in J \\
& \textstyle\sum_{j=1}^{m} w_j x_{ij} \leq C_i, && \forall i \in I \\
& x_{ij} \geq 0, && \forall i, j \\
& x_{ij} = 0, && \forall (i, j) \notin E.
\end{aligned}$$

Lemma 11. *Algorithm 8 returns a* 2*-approximation solution to MKP-LP[E].*

Proof. We will construct a feasible dual solution with a value at most twice the value obtained by the algorithm, then by calling the weak

duality theorem, the claim will follow. Assume x is the outcome of Algorithm 8. Using x we can construct a feasible solution to the dual of MKP-LP[E] given below.

$$\begin{aligned} \text{Minimize} \quad & \textstyle\sum_{j=1}^{m} p_j + \sum_{i=1}^{n} u_i C_i && \text{(MKP-LPD[E])} \\ \text{subject to} \quad & p_j + u_i w_j \geq v_j, && \forall (i,j) \in E \\ & u_i \geq 0, && \forall i \\ & p_j \geq 0, && \forall j. \end{aligned}$$

Initially, let $p = \vec{0}$ and $u = \vec{0}$. If item j gets exhausted, set $p_j = v_j$. Furthermore, for all *full* bins i, set $u_i = v_j/s_j$, j being the last item (fractionally) assigned to i. We can observe that this satisfies the constraint corresponding to each edge (i,j). In particular, if bin i is full, then for each j incident on i, either j gets exhausted with this assignment or does not. If j is exhausted we have $p_j = v_j$ and therefore the constraint holds. If j is not exhausted, we have $v_j/w_j \leq u_i$ since items are assigned in decreasing order of value density and thus the constraint holds. If bin i is not full, every item j which is assigned to it is exhausted by this assignment. That is we have $p_j = v_j$ and the constraint thus holds. For every item j which is not assigned to the bin but $(i,j) \in E$, we have $p_j = v_j$ since the item is exhausted due to another assignment. In sum, we have constructed a feasible dual solution using x.

Now, we bound the value of the dual solution with respect to the primal solution. First, we observe that $\sum_{i,j} v_j x_{ij} \geq \sum_j p_j \sum_i x_{ij}$, since $p_j = v_j$ if j is fully exhausted and $p_j = 0$, otherwise. Second, $\sum_{i,j} v_j x_{ij} = \sum_i \sum_j \frac{v_j}{w_j}(w_j x_{ij}) \geq \sum_i u_i \sum_j (w_{ij} x_{ij})$, since if $x_{ij} > 0$ then $v_j/w_j \geq u_i$. Therefore, we obtain

$$\begin{aligned} 2\textstyle\sum_{i,j} v_j x_{ij} &\geq \sum_j p_j \sum_i x_{ij} + \sum_i u_i \sum_j (w_j x_{ij}) \\ &= \sum_j p_j + \sum_i u_i C_i \end{aligned}$$

Notice, only for items j which get exhausted ($\sum_i x_{ij} = 1$) we have $p_j > 0$ and only for full bins ($\sum_j w_j x_{ij} = C_i$) we have $u_i > 0$. The final term is the value of the dual, the desired conclusion. □

Finally, we call Theorem 5 and obtain the following.

Theorem 6. *There exists a truthful 4-approximation mechanism for the multiple knapsack problem in our model.*

4.2.2 *Truthful Mechanism for GAP-BS*

Now, we attempt to design a truthful algorithm for GAP-BS, but first solve the problem with an additional assumption. We assume that the *value density* of each item is the same over all bins. More formally, there exists a value d_j for each item j such that for all bins i, we have $\frac{v_{ij}}{w_{ij}} = d_j$. This assumption will be relaxed in Subsection 4.2.3. We

design a truthful 4-approximation mechanism for GAP-BS under this extra assumption.

The proposed algorithm can be viewed as a variant of the *deferred acceptance algorithm* designed for matching marketplaces. Each item j has a preference list $\mathcal{L}_j$ according to decreasing order of v_{ij} where $(i, j) \in E$, breaking ties arbitrarily. The preference list of a bin is defined according to the decreasing order of value densities. Once a (fractional) item and a bin are matched, the assignment will never be broken.

Algorithm 9: GAP with Equal Density

Data: Preference lists of the items, $\{\mathcal{L}_j\}_j$.
Result: A feasible solution x to LP[E].
1. Sort items according to their decreasing order of value densities d_j, breaking ties arbitrarily.
2. **foreach** *item j chosen according to the order above* **do**
   Fractionally assign as much of the item to the bins chosen according to the order specified by $\mathcal{L}_j$, until the item is exhausted or all the bins in $\mathcal{L}_j$ are full.
return the resulting assignment x.

To show the approximation factor of the solution, we can construct a feasible dual solution whose value is at most twice the value obtained by Algorithm 9, then by calling the weak duality theorem, the following lemma holds.

Lemma 12. *Algorithm 9 returns a 2-approximation solution to LP[E] when each item has the same value density over bins.*

Proof. An argument similar to that of Lemma 11 in addition to some required modifications will show the claim. Assume x is the outcome of Algorithm 9. Using x we can construct a feasible solution to the dual of LP[E] (LPD[E] given below) which is not greater than twice the value of x. Then we call the weak LP-duality theorem and conclude that x is a 2 approximate solution to LP[E].

$$
\begin{aligned}
&\text{LPD[E]:} \\
&\text{Minimize} \quad \textstyle\sum_{j=1}^{m} p_j + \sum_{i=1}^{n} u_i C_i \\
&\text{subject to} \quad p_j + u_i w_{ij} \geq v_{ij}, \qquad \forall (i,j) \in E \\
&\qquad\qquad u_i \geq 0, \qquad\qquad \forall i \\
&\qquad\qquad p_j \geq 0, \qquad\qquad \forall j.
\end{aligned}
$$

Initially, let $p = \vec{0}$ and $u = \vec{0}$. If item j gets exhausted when assigned to bin i, set $p_j = v_{ij}$. Furthermore, for all *full* bins i, set $u_i = d_j$, j being the last item (fractionally) assigned to i. We can observe that

this satisfies the constraint corresponding to each edge (i, j). In particular, if bin i is full, then for each j incident on i, j either gets exhausted with this assignment or does not. If j is exhausted, we have $p_j = v_{ij}$ and therefore the constraint holds. If j is not exhausted, we have $v_{ij}/w_{ij} = d_j \leq u_i$ since items are assigned in decreasing order of value density and thus the constraint holds. If bin i is not full, every item j which is assigned to it is exhausted by this assignment. That is we have $p_j = v_{ij}$ and the constraint thus holds. For every item j which is not assigned to the bin but $(i, j) \in E$, we have $p_j \geq v_{ij}$ since the item is exhausted due to an assignment $(i', j) \in E$ with $v_{i'j} \geq v_{ij}$. Therefore, we have constructed a feasible dual solution using x.

Now, we bound the value of the dual solution with respect to the primal solution. First, we observe that $\sum_{i,j} v_{ij}x_{ij} \geq \sum_j p_j \sum_i x_{ij}$, since p_j lower bounds the value of any edge on which any part of item j is assigned ($x_{ij} > 0$) because the item goes to bins according to the order specified by $\mathcal{L}_j$. Second, $\sum_{i,j} v_{ij}x_{ij} = \sum_i \sum_j \frac{v_{ij}}{w_{ij}}(w_{ij}x_{ij}) \geq \sum_i u_i \sum_j (w_{ij}x_{ij})$, since if $x_{ij} > 0$ then $\frac{v_{ij}}{w_{ij}} = d_j \geq u_i$. Therefore, we obtain

$$\begin{aligned} 2\sum_{i,j} v_{ij}x_{ij} &\geq \sum_j p_j \sum_i x_{ij} + \sum_i u_i \sum_j (w_{ij}x_{ij}) \\ &= \sum_j p_j + \sum_i u_i C_i \end{aligned}$$

Notice, only for item j which gets exhausted ($\sum_i x_{ij} = 1$), we have $p_j > 0$ and only for full bins ($\sum_j w_j x_{ij} = C_i$) we have $u_i > 0$. The final term is the value of the dual, the desired conclusion. □

Regarding the truthfulness of the algorithm, it would have been easier to prove the truthfulness if we - similar to the algorithm for the multiple knapsack problem - allowed the bins to propose to items; however, one can design instances showing that such an algorithm will result in an arbitrarily low total value. For example, see Fig. 3. In this example when bin **1**, whose capacity is strongly bigger than ϵ, $0 < \epsilon < 1$, starts choosing its desired items, it obtains all items and the resulting allocation will be of very low value.

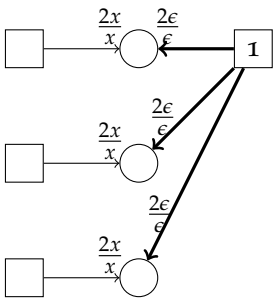


Figure 3: Value/size of each item-bin pair is on the edges. x is an arbitrary big value. Selected assignments are in bold.

In order to show that Algorithm 9 is *fractionally truthful*, we look at Algorithm 9 as a variant of the deferred acceptance algorithm where items propose capacities to bins. Each bin then may *accept* or *reject* the whole or part of the proposed capacity by an item depending on its current empty capacity. Let C_{ij} denote the capacity proposed by item j to bin i.

The truthfulness proof proceeds as follows. We first show that in an instance with 2 items and 2 bins (2×2), truthfulness holds. This instance contains the core of the truthfulness proof for the general case. Truthfulness for simpler cases is trivial. A straightforward generalization of the argument for 2×2 shows truthfulness for settings with m items and 2 bins ($2 \times m$) for any $m > 2$. For the general case of ($n \times m$) we provide an inductive argument.

Lemma 13. *Algorithm 9 is fractionally truthful for* 2×2 *settings.*

Proof. Let $\mathbf{1}$, $\mathbf{2}$ denote the bins and $\mathbf{p}$ and $\mathbf{q}$ denote the items. Let us assume $\mathbf{p}$ precedes $\mathbf{q}$ in proposing to the bins, i.e. $d_p \geq d_q$. Fix this order of proposing items as well as the reports by bin $\mathbf{2}$. We argue bin $\mathbf{1}$ is never better off by hiding some of its edges E_1.

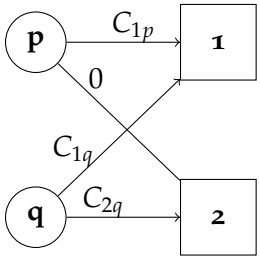
Assume $(\mathbf{1}, \mathbf{q}) \in E_1$. Then bin $\mathbf{1}$ may receive a proposal from $\mathbf{q}$ but obviously the bin receives no proposal from the item if the bin reports $(\mathbf{1}, \mathbf{q}) \notin E_1$. Thus, hiding compatibility with $\mathbf{q}$ might only make a loss for the bin.

Now we analyze the behavior of the algorithm for a similar change in report for item $\mathbf{p}$. We need to show that when $(\mathbf{1}, \mathbf{p}) \in E_1$ (case I) the obtained value is at least as good as when $(\mathbf{1}, \mathbf{p}) \notin E_1$ (case II) for the bin. Then we conclude that when truely $(\mathbf{1}, \mathbf{p}) \in E_1$, the bin has no incentive to report $(\mathbf{1}, \mathbf{p}) \notin E_1$.

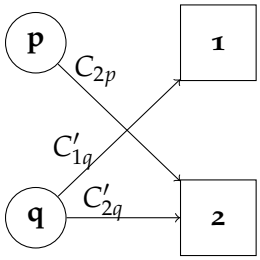
In case I, if no fraction of $\mathbf{p}$ is assigned to bin $\mathbf{1}$ (the bin rejects $\mathbf{p}$), then everything remains the same as in case II. If only a fraction of $\mathbf{p}$ accepted by the bin, then the bin has to be full: thus the utility of the bin is maximum and can't be better off in case II. What remains to show is that the bin is not worse off when it accepts $\mathbf{p}$ fully in case I.

This situation is depicted in Figure 4. In the figure, (a) and (b) correspond to case I and case II, respectively. Considering the information provided in Fig. 4, we need to show that $C'_{1q} \leq C_{1q} + C_{1p}$. This will mean, in case II, the bin actually receives less capacity from items with less (or equal) value densities than in case I, which in turn means a lower value for bin $\mathbf{1}$. Notice, to arrive at this inequality we used the assumption that the order of proposing items is fixed in the two setups. To show the inequality, we first observe two facts about Algorithm 9.

Observation 1. *If a set of items together propose a capacity of* $C^0 \leq C$ *to a bin with capacity* C*, the bin will accept the whole proposed capacity. If we first let a capacity* C^1 *propose to the bin and afterwards let the foregoing*



(a) Case I. **p** is exhausted when it is assigned to **1**.



(b) Case II. **1** is not in the preference list of **p**. At least a fraction of **q** is assigned to **1** ($C'_{1q} > 0$).

Figure 4: Two cases where the bin is and is not on the preference list of the item. The amount of proposed and accepted capacities are shown on the edges.

items propose the capacity C^0, the bin will reject a capacity of at most C^1 from the items that propose after the first capacity.

Proof. Assume C^1 and C^0 in order propose to the bin. If the bin gets full by accepting C^1 we must have $C^1 \geq C$, then the bin will reject exactly a capacity of C^0 of the next items. Now because $C^0 \leq C \leq C^1$, the claim holds. If not (the bin still has an empty capacity of C^E after accepting C^1), the bin accepts C^1 fully and rejects an amount equal to $\max\{0, C^0 - C^E\}$ from the next proposing capacities. We have $C^1 + C^E = C \geq C^0$, therefore $C^0 - C^E \leq C^1$. Thus, in this case the rejected capacity will be upper bounded by C^1. This completes the proof. □

Observation 2. *Let **1** and **2** be two subsequent bins in $\mathcal{L}_j$. If bin **1** rejects the proposed capacity C_{1j} by item j then, this is an upper bound to C_{2j}, the capacity that will be proposed by item j to **2**, i.e. $C_{1j} \geq C_{2j}$.*

Proof. First, we must have $w_{1j} \geq w_{2j}$ since $\frac{v_{1j}}{w_{1j}} = \frac{v_{2j}}{w_{2j}}$ by the assumption of equal density over bins and $v_{1j} \geq v_{2j}$ as **1** precedes **2** in $\mathcal{L}_j$. Rejecting C_{1j} means that this fraction of the item remains: C_{1j}/w_{1j}. Then what will be proposed to **2** is $C_{2j} = w_{2j} \cdot (C_{1j}/w_{1j}) \leq C_{1j}$. □

Back to the argument about cases I and II, we notice that in case II there is an increase of amount C_{2p} in the proposed capacity to **2** compared to case I. The capacity rejected by bin **2** is thus upper bounded by C_{2p} according to Observation 1. Therefore, we have $C_{2q} - C'_{2q} \leq$

C_{2p}. Moreover, according to Observation 2, the rejected capacity upper bounds the proposed capacity to the next bin. Hence, we have $C'_{1q} - C_{1q} \leq C_{2q} - C'_{2q}$. Therefore, we obtain $C'_{1q} \leq C_{1q} + C_{2p} \leq C_{1q} + C_{1p}$. The last inequality holds again because of Observation 2 (see it as bin $\mathbf{1}$ rejecting C_{1p}, an upper bound to C_{2p}). This completes the proof of Lemma 13. □

A simple generalization of the argument for 2×2 markets shows truthfulness for the $2 \times m$ markets with $m > 2$. A useful observation here is that we only need to show that bin $\mathbf{1}$ will always report E_1 rather than $E_1 \setminus \{e_j\}$ for every $e_j \in E_1$. If we show this, we have in fact shown that reporting E_1 is better than reporting $E_1 \setminus \{e_j\}$. This also shows that reporting $E_1 \setminus \{e_j\}$ is better than hiding one edge from $E_1 \setminus \{e_j\}$, i.e. reporting $E_1 \setminus \{e_j, e_{j'}\}$ and so on. For the general case we provide an inductive argument. We assume that in a $(n-1) \times m$ setting bins are truthful and prove that in a $n \times m$ setting truthfulness holds as well.

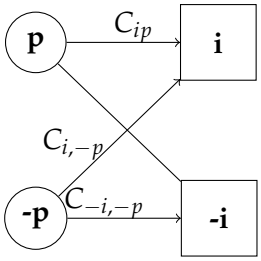
Lemma 14. *If Algorithm 9 is truthful for markets with m items and $n-1$ bins, it will be truthful for $n \times m$ markets.*

Proof. Consider bin $\mathbf{i}$ and fix the reports of other bins denoted by **-i**. We assume $(\mathbf{i},\mathbf{p}) \in E_i$ (case I) and show that the bin will never be better off by reporting $(\mathbf{i},\mathbf{p}) \notin E_i$ (case II). We compare the utility of the bin in the two cases under a fixed order of proposing items. The two cases are depicted in Figure 5. Since the items before $\mathbf{p}$ are assigned similarly in both cases, we only consider the items which are processed after $\mathbf{p}$ denoted by **-p**.

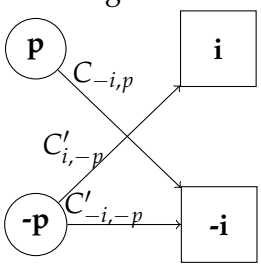
We show that $C'_{i,-p} \leq C_{ip} + C_{i,-p}$, where $C_{i,-p} = \sum_{q \in -p} C_{i,q}$ and $C'_{i,-p} = \sum_{q \in -p} C'_{i,q}$. This means that bin $\mathbf{i}$ in case II actually receives less capacity from items with less (or equal) value densities than case I, which in turn implies lower value for the bin.

Consider case II. We look closer at the bin(s) to which item $\mathbf{p}$ will be assigned. We assume $\mathbf{p}$ is (fractionally) assigned to at least one bin otherwise we have $C_{i,-p} = C'_{i,-p}$ and thus the claim holds. Let bin $\mathbf{1}$ be the first bin to which $\mathbf{p}$ will be assigned.

We assume bin $\mathbf{1}$ gets full at some point otherwise this bin accepts the extra capacity (C_{1p}, the capacity proposed by item $\mathbf{p}$ to bin $\mathbf{1}$) without rejecting any capacity and therefore we have $C_{i,-p} = C'_{i,-p}$ and thus the claim holds. When bin $\mathbf{1}$ gets full, some of the currently proposing items to bin $\mathbf{1}$ will stop proposing to it and go to the next bin in their preference list. Let us call these capacities C_1. C_1 is upper bounded by C_{1p} according to Observation 1 which in turn is upper bounded by C_{ip} based on Observation 2: $C_1 \leq C_{1p} \leq C_{ip}$. If C_1 directly proposes to bin $\mathbf{i}$, the bin won't be better off in case II because $C_1 \leq C_{ip}$. The situation is worse for bin $\mathbf{i}$, if C_1 goes to the other bins. One can view this situation as bin $\mathbf{i}$ rejecting capacity C_1 in a $(n-1) \times m$ setting where bin $\mathbf{1}$ (which is now full) and its absorbed



(a) Case I. **p** is exhausted when it is assigned to **i**. **i** may get a fraction or nothing from other items **-p**.



(b) Case II. **i** hides its compatibility with **p**. At least a fraction of **-p** is assigned to **i**. **p** is (fully) accepted by **-i**.

Figure 5: Two cases where the bin shows or hides its compatibility with an item.

capacities are eliminated. According to our induction assumption, this strategy will not make bin **i** better off in a $(n-1) \times m$ setting. This completes the proof. □

Taking into account, Lemma 13 and Lemma 14, we obtain the following.

Lemma 15. *Algorithm 9 is fractionally truthful.*

Finally, by calling Theorem 5, we obtain the following.

Theorem 7. *There exists a truthful* 4*-approximation mechanism for* GAP-BS *when each item has the same value density over all bins.*

4.2.3 *Unequal Value Densities*

We presented a truthful 4-approximate mechanism for GAP-BS when each item has a unique value density over all bins. Now we explain how to relax this assumption at the expense of a logarithmic loss in the total value. Consider those edges in E, $e = (k, l)$ and $e' = (k', l')$ whose value densities are respectively upper and lower bounds over all value densities:

$$L = \frac{v_{k'l'}}{w_{k'l'}} \leq \frac{v_{ij}}{w_{ij}} \leq \frac{v_{kl}}{w_{kl}} = U, \ \forall (i,j) \in E.$$

Let us assume U and L are publicly known. This assumption will be removed later. Knowing this information we choose a density value d uniformly at random from the set $D = \{U, \frac{U}{2}, \frac{U}{4}, \ldots, \frac{U}{2^{O(\ln(U/L))}}\}$. Then

we define a new valuation $\hat{v}$ as follows. For every edge (i, j) in E with $\frac{v_{ij}}{w_{ij}} < d$ we set $\hat{v}_{ij} = 0$, or equivalently the edge is discarded from the graph. For every $\frac{v_{ij}}{w_{ij}} \geq d$, define $\hat{v}_{ij}$ such that $\frac{\hat{v}_{ij}}{w_{ij}} = d$. Notice that always $\hat{v}_{ij} \leq v_{ij}$. Now we have an instance of GAP-BS with equal densities for which there exists a truthful 4-approximate mechanism according to Theorem 7. To ensure truthfulness, in the end, if item j is assigned to bin i by the subroutine for equal value densities, we withdraw the item with probability $1 - \frac{\hat{v}_{ij}}{v_{ij}}$. In other words, we let the bin hold the item with probability $\frac{\hat{v}_{ij}}{v_{ij}}$. If item j is assigned to bin i, the generated value for the bin will be v_{ij}, but if we let the bin hold the item with probability $\frac{\hat{v}_{ij}}{v_{ij}}$, then the expected value will be $\hat{v}_{ij}$. This way, we make sure that each item has the same value density over all bins as it is required by the subroutine to guarantee truthfulness.

Set D contains $O(\ln(U/L))$ densities, and each density has the probability of $p = \frac{1}{O(\ln(U/L))}$ to be chosen. At least half of each valuation v_{ij} with probability p is counted in the expected total value; therefore, we obtain an $O(\ln(U/L))$ approximation factor.

To remove the assumption that U and L are public, we certify that the bins k and k' wouldn't hide the corresponding edges. To this end, we run one of the following three algorithms with probability $1/3$. *i*) Let bin k (the owner of edge e) choose all its desired items and assign nothing to the other bins. *ii*) Let bin k' (the owner of e') choose all its desired items and assign nothing to the other bins. *iii*) Exclude bin k and k' and run the algorithm above for all other bins using U and L obtained from the two excluded bins. One can observe that the two bins k and k' cannot do any better by hiding their edges. Also, it is easy to observe that the approximation factor is still $O(\ln(U/L))$. Thus, we obtain the following.

Theorem 8. *There exists a truthful $O(\ln(U/L))$ approximate mechanism for GAP-BS.*

We leave open the question of whether there exists a truthful mechanism with a constant factor of approximation for GAP-BS.

5

FAST META-RANDOMIZED ROUNDING

Approximating the optimal social welfare while preserving truthfulness is a well studied problem in algorithmic mechanism design. Assuming that the social welfare of a given mechanism design problem can be optimized by an integer program whose integrality gap is at most α, Lavi and Swamy (2011) propose a general approach to designing a randomized α-approximation mechanism which is truthful in expectation. Their method is based on decomposing an optimal solution for the relaxed linear program into a convex combination of integer solutions. Unfortunately, the decomposition technique developed by Lavi and Swamy relies heavily on the ellipsoid method, which is notorious for its poor practical performance. To overcome this problem, we present an alternative decomposition technique which yields an $\alpha(1+\epsilon)$ approximation and only requires a quadratic number of calls to an integrality gap verifier.

5.1 INTRODUCTION

Optimizing the social welfare in the presence of self-interested players poses two main challenges to algorithmic mechanism design. On the one hand, the social welfare consists of the player's valuations for possible outcomes of the mechanism. However, since these valuations are private information, they can be misrepresented for personal advantage. To avoid strategic manipulation, which may harm the social welfare, it is important to encourage truthful participation. In mechanism design, this is achieved through additional payments which offer each player a monetary incentive to reveal his true valuation. Assuming that the mechanism returns an optimal outcome with respect to the reported valuations, the well known Vickrey, Clarke and Groves (VCG) principle (Vickrey, 1961; Clarke, 1971; Groves, 1973) provides a general method to design payments such that each player maximizes his utility if he reports his valuation truthfully. On the other hand, even if the player's valuations are known, optimizing the social welfare is NP-hard for many combinatorial mechanism design problems. Since an exact optimization is intractable under these circumstances, the use of approximation algorithms becomes necessary.

Unfortunately, VCG payments are generally not compatible with approximation algorithms.

To preserve truthfulness, so called maximal-in-range (MIR) approximation algorithms must be used (Nisan and Ronen, 2007). This means there must exist a fixed subset of outcomes, such that the approximation algorithm performs optimally with respect to this subset. Given that the players are risk-neutral, the concept of MIR algorithms can be generalized to distributions over outcomes. Together with VCG payments, these maximal-in-distributional-range (MIDR) algorithms allow for the design of randomized approximation mechanisms such that each player maximizes his expected utility if he reveals his true valuation (Dobzinski and Dughmi, 2009). This property, which is slightly weaker than truthfulness in its deterministic sense, is also referred to as truthfulness in expectation.

A well-known method to convert general approximation algorithms which verify an integrality gap of α into MIDR algorithms is the linear programing approach of Lavi and Swamy (2011). Conceptually, their method is based on the observation that scaling down a packing polytope by its integrality gap yields a new polytope which is completely contained in the convex hull of the integer points of the original polytope. Considering that the social welfare of many combinatorial mechanism design problems can be expressed naturally as an integer program, this scaled polytope corresponds to a set of distributions over the outcomes of the mechanism. Thus, by decomposing a scaled solution of the relaxed linear program into a convex combination of integer solutions, Lavi and Swamy obtain an α-approximation mechanism which is MIDR.

Algorithmically, the work of Lavi and Swamy builds on a decomposition technique by Carr and Vempala (2000), which uses a linear program to decompose the scaled relaxed solution. However, since this linear program might have an exponential number of variables, one for every outcome of the mechanism, it can not be solved directly. Instead, Carr and Vempala use the ellipsoid method in combination with an integrality gap verifier to identify a more practical, but still sufficient, subset of outcomes for the decomposition. This method is termed *meta-randomized rounding*. Although this approach only requires a polynomial number of calls to the integrality gap verifier in theory, the ellipsoid method is notoriously inefficient in practice (Bland et al., 1981).

In this work, we propose an alternative decomposition technique which does not rely on the ellipsoid method and is general enough to substitute Carr and Vempala's decomposition technique. The main component of our decomposition technique is an algorithm which is based on a simple geometric idea and computes a convex combination within an arbitrarily small distance ϵ to the scaled relaxed solution. However, since an exact decomposition is necessary to guar-

antee truthfulness, we slightly increase the scaling factor of the relaxed solution and apply a post-processing step to match the convex combination with the relaxed solution. Assuming that ϵ is positive and fixed, our technique yields an $\alpha(1+\epsilon)$ approximation of the optimal social welfare but uses only a quadratic number of calls to the integrality gap verifier, with respect to the number of positive components in the relaxed solution vector.

It turns out that our method has interesting connections to an old algorithm of Von Neumann reproduced by Dantzig (1992) [1]. At first sight, similarities in the sampling and geometric techniques used in both algorithms can be observed. However, Von Neumann's algorithm may sample fractional points whereas our setting requires integral points. Due to these more involved constraints, a direct usage of Von Neumann's technique in our setting is impossible.

5.2 SETTING

Integer programming is a powerful tool in combinatorial optimization. Using binary variables to indicate whether certain goods are allocated to a player, the outcomes of various NP-hard mechanism design problems, such as combinatorial auctions or generalized assignment problems (Lavi and Swamy, 2011; Dughmi and Ghosh, 2010), can be modeled as integer points of an n-dimensional packing polytope $X \subseteq [0,1]^n$.

Definition 11. ***(Packing Polytope)*** *Polytope X satisfies the packing property if all points y which are dominated by some point x from X are also contained in X*

$$\forall x, y \in \mathbb{R}^n_{\geq 0} : \; x \in X \text{ and } x \geq y \Rightarrow y \in X.$$

Together with a vector $\mu \in \mathbb{R}^n_{\geq 0}$ which denotes the accumulated valuations of the players, it is possible to express the social welfare as an integer program of the form $\max_{x \in \mathbb{Z}(X)} \sum_{k=1}^{n} \mu_k x_k$, where $\mathbb{Z}(X)$ denotes the set of integer points in X. Using the simplex method, or other standard linear programming techniques, an optimal solution $x^* \in X$ of the relaxed linear program $\max_{x \in X} \sum_{k=1}^{n} \mu_k x_k$ can be computed efficiently for most mechanism design problems. Note that for combinatorial auctions, where the dimension of X grows exponentially with the number of available goods, special attention is necessary to preserve computational feasibility. One possible approach is the use of demand queries which yields an optimal solution in polynomial time and with a polynomial number of positive components (Blumrosen and Nisan, 2005).

The maximum ratio between the original program and its relaxation is called the integrality gap of X. Assuming this gap is at most

1 We thank the anonymous referee of WINE 2014 who pointed us to this paper.

$\alpha \in \mathbb{R}_{\geq 1}$, Lavi and Swamy (2011) observe that the scaled fractional solution $\frac{x^*}{\alpha}$ can be decomposed into a convex combination of integer solutions. More formally, there exists a convex combination λ from the set $\Lambda = \{\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)} \mid \sum_{x \in \mathbb{Z}(X)} \lambda_x = 1\}$ such that the point $\sigma(\lambda)$, which is defined as $\sigma(\lambda) = \sum_{x \in \mathbb{Z}(X)} \lambda_x x$, is equal to $\frac{x^*}{\alpha}$. Regarding λ as a probability distribution over the feasible integer solutions, the MIDR principle allows for the construction of a randomized α-approximation mechanism which is truthful in expectation.

From an algorithmic point of view, the main challenge in decomposing $\frac{x^*}{\alpha}$ is the computation of suitable integer points. Since the size of $\mathbb{Z}(X)$ is typically exponential in n, it is intractable to consider the entire polytope. Instead, Carr and Vempala (2000) propose the use of an approximation algorithm $\mathcal{A} : \mathbb{R}_{\geq 0}^n \to \mathbb{Z}(X)$ which verifies an integrality gap of α to sample a more practical, but still sufficient, subset of integer points.

Definition 12. ***(Integrality Gap Verifier)*** *Approximation algorithm $\mathcal{A}$ verifies an integrality gap of α if the integer solution which is computed by $\mathcal{A}$ is at least α times the optimal relaxed solution for all non-negative vectors μ*

$$\forall \mu \in \mathbb{R}_{\geq 0}^n : \ \alpha \sum_{k=1}^{n} \mu_k \mathcal{A}(\mu)_k \geq \max_{x \in X} \sum_{k=1}^{n} \mu_k x_k.$$

As it turns out, Carr and Vempala's approach only requires a polynomial number of calls to to $\mathcal{A}$ with respect to n. In particular, this implies that the number of integer points in λ, which is defined as $\psi(\lambda) = |\{x \in \mathbb{Z}(X) \mid \lambda_x > 0\}|$, is polynomial as well. Observe that for sparse x^*, which are common in the case of combinatorial auctions, it is only necessary to consider the subspace of positive components in x^*. This is possible since no point in $\mathbb{Z}(X)$ which has a positive component k can contribute to λ if x_k^* is 0. In either case, the fact that Carr and Vempala strongly rely on the ellipsoid method indicates that their results are more of theoretical importance than of practical use.

5.3 DECOMPOSITION WITH EPSILON PRECISION

The first part of our decomposition technique is to construct a convex combination λ such that the point $\sigma(\lambda)$ is within an arbitrarily small distance $\epsilon \in \mathbb{R}_{>0}$ to the scaled relaxed solution $\frac{x^*}{\alpha}$. Similar to the approach proposed by Carr and Vempala, our technique requires an approximation algorithm $\mathcal{A}' : \mathbb{R}^n \to \mathbb{Z}(X)$ to sample integer points from X. It is important to note that $\mathcal{A}'$ must verify an integrality gap of α for arbitrary vectors $\mu \in \mathbb{R}^n$ whereas $\mathcal{A}$, only accepts non-negative vectors. However, since X satisfies the packing property, it is

easy to extend the domain of $\mathcal{A}$ while preserving an approximation ratio of α.

Lemma 16. *Approximation algorithm $\mathcal{A}$ can be extended to a new approximation algorithm $\mathcal{A}'$ which verifies an integrality gap of α for arbitrary vectors μ.*

Proof. The basic idea of $\mathcal{A}'$ is to replace all negative components of μ by 0 and run the original integrality gap verifier $\mathcal{A}$ on the resulting non-negative vector, which is defined as $\xi(\mu)_k = \max(\{\mu_k, 0\})$. Exploiting the fact that X is a packing polytope, the output of $\mathcal{A}$ is then set to 0 for all negative components of μ. More formally, $\mathcal{A}'$ is defined as

$$\mathcal{A}'(\mu)_k = \begin{cases} \mathcal{A}(\xi(\mu))_k & \text{if } \mu_k \geq 0 \\ 0 & \text{if } \mu_k < 0. \end{cases}$$

Since $\mathcal{A}'(\mu)_k$ is equal to 0 if μ_k is negative and otherwise corresponds to $\mathcal{A}(\xi(\mu))_k$, it holds that

$$\sum_{k=1}^{n} \mu_k \mathcal{A}'(\mu)_k = \sum_{k=1}^{n} \xi(\mu)_k \mathcal{A}'(\mu)_k = \sum_{k=1}^{n} \xi(\mu)_k \mathcal{A}(\xi(\mu))_k.$$

Furthermore, since X only contains non-negative points, $\max_{x \in X} \sum_{k=1}^{n} \xi(\mu)_k x_k$ must be greater or equal to $\max_{x \in X} \sum_{k=1}^{n} \mu_k x_k$. Together with the fact that $\mathcal{A}$ verifies an integrality gap of α for $\xi(\mu)$ this proves that $\mathcal{A}'$ verifies the same integrality gap for μ

$$\alpha \sum_{k=1}^{n} \mu_k \mathcal{A}'(\mu)_k = \alpha \sum_{k=1}^{n} \xi(\mu)_k \mathcal{A}(\xi(\mu))_k \geq \max_{x \in X} \sum_{k=1}^{n} \xi(\mu)_k x_k \geq \max_{x \in X} \sum_{k=1}^{n} \mu_k x_k.$$

□

Once $\mathcal{A}'$ is specified, Algorithm 10 is used to decompose $\frac{x^*}{\alpha}$. Starting at the origin, which can be expressed trivially as a convex combination from Λ due to the packing property of X, the algorithm gradually improves $\sigma(\lambda^i)$ until it is sufficiently close to $\frac{x^*}{\alpha}$. For each iteration of the algorithm, μ^i denotes the vector which points from $\sigma(\lambda^i)$ to $\frac{x^*}{\alpha}$. If the length of μ^i is less or equal to ϵ, then $\sigma(\lambda^i)$ must be within an ϵ-distance to $\frac{x^*}{\alpha}$ and the algorithm terminates. Otherwise, $\mathcal{A}'$ samples a new integer point x^{i+1} based on the direction of μ^i. It is important to observe that all points on the line segment between $\sigma(\lambda^i)$ and x^{i+1} can be expressed as a convex combination of the form $\delta\lambda^i + (1-\delta)\tau(x^{i+1})$, where δ is a value between 0 and 1 and $\tau(x^{i+1})$ denotes a convex combination such that the coefficient $\tau(x^{i+1})_{x^{i+1}}$ is equal to 1 while all other coefficients are 0. Thus, by choosing λ^{i+1} as the convex combination which minimizes the distance between the line segment and $\frac{x^*}{\alpha}$, an improvement over the current convex combination is possible. As theorem 1 shows, at most $\lceil n\epsilon^{-2} \rceil - 1$ iterations are necessary to obtain the desired ϵ-precision.

Algorithm 10: Decomposition with Epsilon Precision

Data: an optimal relaxed solution x^*, an approximation algorithm $\mathcal{A}'$, a precision ϵ
Result: a convex combination λ which is within an ϵ-distance to $\frac{x^*}{\alpha}$

$x^0 \leftarrow 0,\ \lambda^0 \leftarrow \tau(x^0),\ \mu^0 \leftarrow \frac{x^*}{\alpha} - \sigma(\lambda^0),\ i \leftarrow 0$
while $\|\mu^i\|_2 > \epsilon$ **do**
  $x^{i+1} \leftarrow \mathcal{A}'(\mu^i)$
  $\delta \leftarrow \arg\min_{\delta \in [0,1]} \|\frac{x^*}{\alpha} - (\delta\sigma(\lambda^i) + (1-\delta)x^{i+1})\|_2$
  $\lambda^{i+1} \leftarrow \delta\lambda^i + (1-\delta)\tau(x^{i+1});\ \mu^{i+1} \leftarrow \frac{x^*}{\alpha} - \sigma(\lambda^{i+1});\ i \leftarrow i+1$
return λ^i

Theorem 9. *Algorithm 10 returns a convex combination within an ϵ-distance to the scaled relaxed solution $\frac{x^*}{\alpha}$ after at most $\lceil n\epsilon^{-2} \rceil - 1$ iterations.*

Proof. Clearly, Algorithm 10 terminates if and only if the distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ becomes less or equal to ϵ. Thus, suppose the length of vector μ^i is still greater than ϵ. Consequently, approximation algorithm $\mathcal{A}'$ is deployed to sample a new integer point x^{i+1}. Keeping in mind that $\mathcal{A}'$ verifies an integrality gap of α, the value of x^{i+1} must be greater than or equal to the value of $\frac{x^*}{\alpha}$ with respect to vector μ^i

$$\sum_{k=1}^{n} \mu_k^i x_k^{i+1} = \sum_{k=1}^{n} \mu_k^i \mathcal{A}'(\mu^i)_k \geq \max_{x \in X} \sum_{k=1}^{n} \mu_k^i \frac{x_k}{\alpha} \geq \sum_{k=1}^{n} \mu_k^i \frac{x^*}{\alpha}.$$

Conversely, since the squared distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ is greater than ϵ^2, and therefore also greater than 0, it holds that the value of $\sigma(\mu^i)$ is less than the value of $\frac{x^*}{\alpha}$ with respect to vector μ^i

$$\begin{aligned}
& 0 < \sum_{k=1}^{n} \left(\frac{x_k^*}{\alpha} - \sigma(\lambda^i)_k\right)^2 \\
\iff \quad & 0 < \sum_{k=1}^{n} \left(\left(\frac{x_k^*}{\alpha}\right)^2 - 2\frac{x_k^*}{\alpha}\sigma(\lambda^i)_k + \sigma(\lambda^i)_k^2\right) \\
\iff \quad & \sum_{k=1}^{n} \left(\frac{x_k^*}{\alpha}\sigma(\lambda^i)_k - \sigma(\lambda^i)_k^2\right) < \sum_{k=1}^{n} \left(\left(\frac{x_k^*}{\alpha}\right)^2 - \frac{x_k^*}{\alpha}\sigma(\lambda^i)_k\right) \\
\iff \quad & \sum_{k=1}^{n} \mu_k^i \sigma(\lambda^i)_k < \sum_{k=1}^{n} \mu_k^i \frac{x_k^*}{\alpha}.
\end{aligned}$$

As a result, the hyperplane $\{x \in \mathbb{R}^n \mid \sum_{k=1}^{n} \mu_k^i x_k = \sum_{k=1}^{n} \mu_k^i \frac{x_k^*}{\alpha}\}$ separates $\sigma(\lambda^i)$ from x^{i+1}, which in turn implies that the line segment $\text{conv}(\{\sigma(\lambda^i), x^{i+1}\})$ intersects the hyperplane at a unique point z^{i+1}.

Since the hyperplane is orthogonal to μ^i, the points $\frac{x^*}{\alpha}$, $\sigma(\lambda^i)$ and z^{i+1} form a right triangle, as Figure 6 illustrates. Furthermore, the altitude of this triangle minimizes the distance from the line segment $\text{conv}(\{\sigma(\lambda^i), x^{i+1}\})$ to $\frac{x^*}{\alpha}$ and therefore corresponds to the length of new vector μ^{i+1}. According to the basic relations between the sides in a right triangle, the length of μ^{i+1} can be expressed as

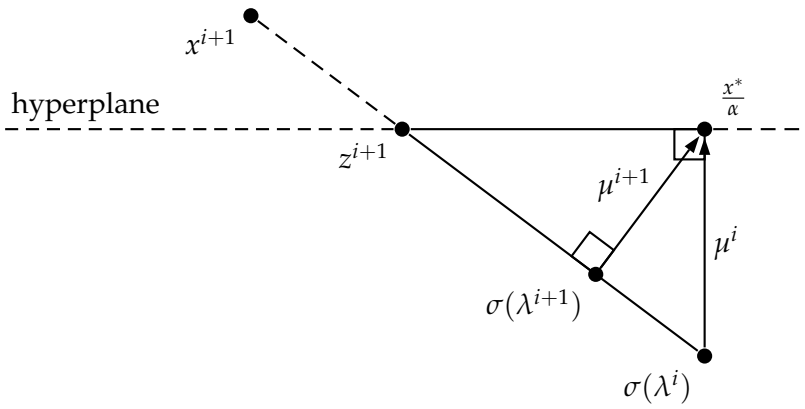


Figure 6: Right triangle between the points $\frac{x^*}{\alpha}$, $\sigma(\lambda^i)$ and z^{i+1}

$$\left\|\mu^{i+1}\right\|_2 = \sqrt{\frac{\left\|\mu^i\right\|_2^2 \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}{\left\|\mu^i\right\|_2^2 + \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}}.$$

Unfortunately, the exact position of z^{i+1}, depends on the implementation $\mathcal{A}'$. To obtain an upper bound on the length μ^{i+1} which does not rely on z^{i+1}, it is helpful to observe that the altitude of the triangle grows as the distance between z^{i+1} and $\frac{x^*}{\alpha}$ increases. However, since both points are contained in the standard hypercube $[0,1]^n$, the square of this distance is at most n

$$\left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2 = \sum_{k=1}^{n} \left(\frac{x_k^*}{\alpha} - z_k^{i+1}\right)^2 \leq \sum_{k=1}^{n} 1 = n,$$

which means that the maximum length of μ^{i+1} is given by

$$\begin{aligned}
& \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2 \leq n \\
\iff \quad & \frac{\left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}{\left\|\mu^i\right\|_2^2 + \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2} \leq \frac{n}{\left\|\mu^i\right\|_2^2 + n} \\
\iff \quad & \frac{\left\|\mu^i\right\|_2^2 \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}{\left\|\mu^i\right\|_2^2 + \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2} \leq \frac{\left\|\mu^i\right\|_2^2 n}{\left\|\mu^i\right\|_2^2 + n} \\
\iff \quad & \sqrt{\frac{\left\|\mu^i\right\|_2^2 \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}{\left\|\mu^i\right\|_2^2 + \left\|\frac{x^*}{\alpha} - z^{i+1}\right\|_2^2}} \leq \sqrt{\frac{\left\|\mu^i\right\|_2^2 n}{\left\|\mu^i\right\|_2^2 + n}}.
\end{aligned}$$

It is important to note that this upper bound on the length of μ^{i+1}, only depends on the previous vector μ^i and the dimension n. Solving the recurrence inequality yields yet another upper bound which is based on the initial vector μ^0 and the number of iterations i

$$
\begin{aligned}
& & \left\|\mu^i\right\|_2^2 &\leq \frac{\left\|\mu^{i-1}\right\|_2^2 n}{\left\|\mu^{i-1}\right\|_2^2 + n} \\
\iff & & \frac{\left\|\mu^i\right\|_2^2}{n} &\leq \frac{\left\|\mu^{i-1}\right\|_2^2}{\left\|\mu^{i-1}\right\|_2^2 + n} \\
\iff & & \frac{n}{\left\|\mu^i\right\|_2^2} &\geq \frac{n}{\left\|\mu^{i-1}\right\|_2^2} + 1 \ldots \\
\implies & & \frac{n}{\left\|\mu^i\right\|_2^2} &\geq \frac{n}{\left\|\mu^0\right\|_2^2} + i \\
\iff & & \frac{\left\|\mu^i\right\|_2^2}{n} &\leq \frac{\left\|\mu^0\right\|_2^2}{\left\|\mu^0\right\|_2^2 i + n} \\
\iff & & \left\|\mu^i\right\|_2 &\leq \sqrt{\frac{\left\|\mu^0\right\|_2^2 n}{\left\|\mu^0\right\|_2^2 i + n}}.
\end{aligned}
$$

Considering that the squared length of vector μ^0, which corresponds to the distance between $\frac{x^*}{\alpha}$ and the origin, is at most n

$$\left\|\mu^0\right\|_2^2 = \sum_{k=1}^{n} \left(\frac{x_k^*}{\alpha}\right)^2 \leq \sum_{k=1}^{n} 1 = n,$$

it follows that

$$\left\|\mu^i\right\|_2 \leq \sqrt{\frac{\left\|\mu^0\right\|_2^2 n}{\left\|\mu^0\right\|_2^2 i + n}} \leq \sqrt{\frac{n^2}{ni + n}} = \sqrt{\frac{n}{i+1}}.$$

Finally, this proves that the distance between $\sigma(\lambda^i)$ and $\frac{x^*}{\alpha}$ must be less or equal to ϵ after not more than $\lceil n\epsilon^{-2} \rceil - 1$ iterations, at which point the algorithm terminates

$$\left\|\mu^{\lceil n\epsilon^{-2} \rceil - 1}\right\|_2 \leq \sqrt{\frac{n}{1 + (\lceil n\epsilon^{-2} \rceil - 1)}} \leq \epsilon.$$

This completes the proof. □

At this point, it should be mentioned that the upper bound on the number of iterations given in Theorem 1 can be further refined with a simple modification of $\mathcal{A}'$. Due to the packing property of X it is possible to set all components of $\mathcal{A}'(\mu^i)$ which correspond to a component of value 0 in x^* to 0 as well. Given that μ^0 is equal to $\frac{x^*}{\alpha}$ and all other μ^{i+1} are defined recursively as the difference between $\frac{x^*}{\alpha}$ and a convex combination of $\frac{x^*}{\alpha} - \mu^i$ and $\mathcal{A}'(\mu^i)$, every vector μ^{i+1} must share the 0 components of x^*. As a result, the new $\mathcal{A}'$ preserves the approximation ratio and Algorithm 10 still works as expected.

Furthermore, only integer points from the subspace of positive components in x^* are considered, which means that the convergence of Algorithm 10 depends on the number of positive components in x^* rather than n.

5.4 EXACT DECOMPOSITION

Although the convex combination λ which is returned by Algorithm 10 is within an ϵ-distance to $\frac{x^*}{\alpha}$, an exact decomposition of the relaxed solution is necessary to guarantee truthfulness. Assuming that an additional scaling factor of $\sqrt{n}\epsilon$ is admissible, the second part of our decomposition technique shows how to convert λ into a new convex combination λ'' such that $\sigma(\lambda'')$ is equal to $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$. Note that this additional scaling factor depends on ϵ, which means that it can still be made arbitrarily small. In particular, running Algorithm 10 with a precision of $\frac{\epsilon}{\sqrt{n}}$, instead of ϵ, reduces the factor to ϵ and yields a decomposition which is equal to $\frac{x^*}{\alpha(1+\epsilon)}$. However, since this new precision is not independent of n anymore, the maximum number of iterations is increased to $\lceil n(\frac{\epsilon}{\sqrt{n}})^{-2}\rceil - 1$, which is quadratic in n. It is helpful to observe that the techniques which are introduced in this section can be adapted easily to the subspace of positive components in x^*. Hence, all complexity results carry over directly from n to the number of positive components in x^*.

To adjust $\sigma(\lambda)$ component-wisely, it is helpful to consider the integer points $e^k \in \{0,1\}^n$. For every dimension k, the kth component of e^k is defined to be 1 while all other components are 0. Since X has a finite integrality gap and also satisfies the packing property, all points e^k must be contained in X.

Lemma 17. *The polytope X contains all points e^k.*

Proof. For the sake of contradiction, assume there exists a dimension k for which e^k is not contained in X. Since X satisfies the packing property, this implies that there exists no point in X whose kth component is 1, in particular no integer point. As a result, the optimal solution for the integer program with respect to the vector e^k must be 0

$$\max_{x \in \mathbb{Z}(X)} \sum_{l}^{n} e_l^k x_l = \max_{x \in \mathbb{Z}(X)} x_k = 0.$$

Keeping in mind that X has an integrality gap of at most α, it immediately follows that the optimal solution for the relaxed linear program with respect to e^k must also be 0

$$\max_{x \in X} \sum_{l}^{n} e_l^k x_l = \max_{x \in X} x_k = 0.$$

However, this implies that the kth component of every point in X is 0, which contradicts the fact that X is n-dimensional. □

Applying Theorem 10, our decomposition technique uses the points e^k to construct an intermediate convex combination λ' such that $\sigma(\lambda')$ dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$.

Theorem 10. *Convex combination λ can be converted into a new convex combination λ' which dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$.*

Proof. According to Lemma 17, the points e^k are contained in $\mathbb{Z}(X)$. Thus, they can be added to λ to construct a positive combination $\lambda + \sum_{k=1}^{n} |\frac{x_k^*}{\alpha} - \sigma(\lambda)_k| \tau(e^k)$ which dominates $\frac{x^*}{\alpha}$

$$\begin{aligned}
\sigma\Big(\lambda + \sum_{k=1}^{n} \Big|\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big|\tau(e^k)\Big) &= \sigma(\lambda) + \Big(\sum_{k=1}^{n} \Big|\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big|e^k\Big) \\
&\geq \sigma(\lambda) + \Big(\sum_{k=1}^{n} \Big(\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big)e^k\Big) \\
&= \sigma(\lambda) + \frac{x^*}{\alpha} - \sigma(\lambda) \\
&= \frac{x^*}{\alpha}.
\end{aligned}$$

Since the sum over the additional coefficients $\sum_{k=1}^{n} |\frac{x_k^*}{\alpha} - \sigma(\lambda)_k|$ is equivalent to the L1 distance between $\sigma(\lambda)$ and $\frac{x^*}{\alpha}$, it is bounded by the Hölder inequality

$$\sum_{k=1}^{n} \Big|\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big| = \Big\|\frac{x^*}{\alpha} - \sigma(\lambda)\Big\|_1 \leq \Big\|1\Big\|_2 \Big\|\frac{x^*}{\alpha} - \sigma(\lambda)\Big\|_2 \leq \sqrt{n}\epsilon.$$

As a result, scaling down the positive combination by a factor of $1 + \sqrt{n}\epsilon$ yields a new positive combination which dominates $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$ and whose coefficients sum up to a value less or equal to 1. To ensure that this sum becomes exactly 1, the coefficients must be increased by an additional value of $\sqrt{n}\epsilon - \sum_{k=1}^{n} |\frac{x_k^*}{\alpha} - \sigma(\lambda)_k|$. An easy way to achieve this is by adding the origin, which is trivially contained in $\mathbb{Z}(X)$ due to the packing property of X, to the positive combination. Thus, the desired convex combination λ' corresponds to

$$\frac{\lambda + \sum_{k=1}^{n} \Big|\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big|\tau(e^k) + \Big(\sqrt{n}\epsilon - \sum_{k=1}^{n} \Big|\frac{x_k^*}{\alpha} - \sigma(\lambda)_k\Big|\Big)\tau(0)}{1 + \sqrt{n}\epsilon}.$$

□

In the final step, our decomposition technique exploits the packing property of X to convert λ' into an exact decomposition of $\frac{x^*}{\alpha(1+\sqrt{n}\epsilon)}$.

A simple but general approach to this problem is provided by Algorithm 11. Given a point $x \in X$ which is dominated by $\sigma(\lambda')$, the basic idea of the algorithm is to iteratively weaken the integer points which comprise λ' until the desired convex combination λ'' is reached. As Theorem 11 shows, this computation requires at most $|\psi(\lambda)|n + \frac{n^2+n}{2}$ iterations.

Algorithm 11: From a Dominating to an Exact Decomposition

Data: a convex combination λ', a point x which is dominated by $\sigma(\lambda')$
Result: a convex combination λ'' which is an exact decomposition of x

$\lambda^0 \leftarrow \lambda'$, $i \leftarrow 0$
foreach $1 \leq k \leq n$ **do**
  while $\sigma(\lambda^i)_k > x_k$ **do**
    $y \leftarrow$ pick some y from $\mathbb{Z}(X)$ such that $\lambda^i_y > 0$ and $y_k = 1$
    if $\lambda^i_y \geq \sigma(\lambda^i)_k - x_k$ **then**
      $\lambda^{i+1} \leftarrow \lambda^i - (\sigma(\lambda^i)_k - x_k)\tau(y) + (\sigma(\lambda^i)_k - x_k)\tau(y - e^k)$
    else
      $\lambda^{i+1} \leftarrow \lambda^i - \lambda^i_y \tau(y) + \lambda^i_y \tau(y - e^k)$
    $i \leftarrow i + 1$
return λ^i

Theorem 11. *Assuming that $\sigma(\lambda')$ dominates the point x, Algorithm 11 converts λ' into a new convex combination λ'' such that $\sigma(\lambda'')$ is equal to x. Furthermore, the required number of iterations is at most $|\psi(\lambda')|n + \frac{n^2+n}{2}$.*

Proof. In order to match $\sigma(\lambda')$ with x, Algorithm 11 considers each dimension k separately. Clearly, while $\sigma(\lambda^i)_k$ is still greater than x_k, there must exist at least one point y in λ^i which has a value of 1 in component k. If λ^i_y is greater or equal to the difference between $\sigma(\lambda^i)_k$ and x_k, it is reduced by the value of this difference. To compensate for this operation, the coefficient of the point $y - e^k$, which is trivially contained in X due to its packing property, is increased by the same value. Thus, the value of $\sigma(\lambda^{i+1})_k$ is equal to x_k

$$\begin{aligned}
\sigma(\lambda^{i+1})_k &= \sigma(\lambda^i)_k - (\sigma(\lambda^i)_k - x_k)\tau(y)_k + (\sigma(\lambda^i)_k - x_k)\tau(y - e^k)_k \\
&= \sigma(\lambda^i)_k - (\sigma(\lambda^i)_k - x_k) \\
&= x_k,
\end{aligned}$$

which means that the algorithm succeeded at computing a matching convex combination for x at component k. It should be noted that the other components of λ^{i+1} are unaffected by this update.

Conversely, if λ_y^i is less than the remaining difference between $\sigma(\lambda^i)_k$ and x_k, the point y can be replaced completely by $y - e^k$. In this case the value of $\sigma(\lambda^{i+1})_k$ remains greater than x_k

$$\sigma(\lambda^{i+1})_k = \sigma(\lambda^i)_k - \lambda_y^i \tau(y)_k + \lambda_y^i \tau(y - e^k)_k = \sigma(\lambda^i)_k - \lambda_y^i > x_k$$

Furthermore, the number of points in λ^{i+1} which have a value of 1 at component k is reduced by one with respect to λ^i. Considering that the number of points in λ^i is finite, this implies that the algorithm must eventually compute a convex combination λ'' which matches x at component k.

To determine an upper bound on the number of iterations, it is helpful to observe that the size of the convex combination can only increase by 1 for every iteration of the for loop, namely if λ_y^i is greater than the difference between $\sigma(\lambda^i)_k$ and x_k. As a result, the number of points which comprise a convex combination during the kth iteration of the for loop is at most $\psi(\lambda') + k$. Since this number also gives an upper bound on the number of iterations performed by the while loop, the total number of iterations is at most

$$\sum_{k=1}^{n} (|\psi(\lambda')| + k) = n|\psi(\lambda')| + \sum_{k=1}^{n} k = n|\psi(\lambda')| + \frac{n^2 + n}{2}.$$

□

5.4.1 *Simpler Exact Decomposition*

Our final goal of representing the fractional point as a convex decomposition of integer points is to sample a random integer solution according to the respective probability in the convex decomposition. Exploiting this fact, we can replace Algorithm 11 with a simple idea. Given the dominating point λ', we treat the convex decomposition of λ' as a probability distribution, and sample an integer point from it. Let $x \in \mathbb{Z}(X)$ be the resulting integer solution. Then, independently, for each component $x_k = 1$, we update x_k to 0 with probability $p_k = \frac{\frac{x_k^*}{\alpha(1+\epsilon)}}{\sigma(\lambda')_k}$. Note that for every k, $\sigma(\lambda')_k \geq \frac{x_k^*}{\alpha(1+\epsilon)}$, thus probability p_k is correctly defined. Let x' denote the new solution. From the packing property of X, the new solution x' belongs to $\mathbb{Z}(X)$ and is thus a feasible solution. For each k, we have $Pr[x'_k = 1] = \sigma(\lambda')_k \cdot \frac{\frac{x_k^*}{\alpha(1+\epsilon)}}{\sigma(\lambda')_k} = \frac{x_k^*}{\alpha(1+\epsilon)}$. In other words, x' is an integer solution sampled from the probability distribution that corresponds to the convex decomposition of $\frac{x^*}{\alpha(1+\epsilon)}$, the desired outcome.

6

MECHANISM DESIGN VIA DANTZIG-WOLFE DECOMPOSITION

In random allocation rules, typically first an optimal fractional point is calculated via solving a linear program. The calculated point represents a fractional assignment of objects or more generally packages of objects to agents. In order to implement an expected assignment, the mechanism designer must decompose the point into integer solutions, each satisfying underlying constraints. The resulting convex combination can then be viewed as a probability distribution over feasible assignments out of which a random assignment can be sampled. This approach has been successfully employed in combinatorial optimization as well as mechanism design with or without money.

In this chapter, we show that both finding the optimal fractional point as well as its decomposition into integer solutions can be done at once. We propose an appropriate linear program which provides the desired solution. We show that the linear program can be solved via Dantzig-Wolfe decomposition. Dantzig-Wolfe decomposition is a direct implementation of the revised simplex method which is well known to be highly efficient in practice. We also show how to use the Benders decomposition as an alternative method to solve the problem. The proposed method can also find a decomposition into integer solutions when the fractional point is readily present perhaps as an outcome of other algorithms rather than linear programming. The resulting convex decomposition in this case is tight in terms of the number of integer points according to the Carathéodory's theorem.

The result presented in this chapter has advantages over the result in Chapter 5. First, the size of the convex decomposition (number of integer solutions) is strictly smaller than the size of the convex decomposition produced by the method in Chapter 5. Second, the decomposition is exact and does not suffer from an $\epsilon > 0$ compromise in the solution. However, we provide no theoretical upper bound on the number of iterations, and the proposed method relies on the performance of Dantzig-Wolfe decomposition in practice.

6.1 INTRODUCTION

The technique of finding a fractional solution and decomposing it into polynomially-many integer points has been successfully employed in many problems. For a usage of the technique in combinatorial optimization, for instance, see Carr and Vempala (2000). In mechanism design with bidders who have quasi-linear valuations, the framework presented by Lavi and Swamy for designing truthful and approximate mechanisms strongly relies on this technique (Lavi and Swamy, 2011, 2005). Perhaps the best connection between linear programming and algorithmic mechanism design has been established by this framework. Finally, for applications in mechanism design without money see e.g. Chapter 4, Budish et al. (2013) and Nguyen et al. (2015).

Typically in such applications, first a fractional optimal point is calculated, and in a second step, the point is represented as a convex combination of integer points usually using the ellipsoid method. A subroutine or an approximation algorithm which returns an integer point with respect to a cost vector is employed to construct the separation oracle for the ellipsoid method. A separation oracle, in the ellipsoid method, states if a given point is feasible, or in case it is not feasible, the oracle returns a violated constraint. A natural question is to ask if the two steps, optimization as well as convex decomposition, can be done at once without using the ellipsoid method?

6.1.1 *Results and Techniques*

We propose an appropriate linear program for finding an optimal fractional point and its decomposition into integer points. We show how to use the Dantzig-Wolfe decomposition which is based on the revised simplex to solve the linear program. More specifically, we show that finding a convex combination of integer points whose value is maximum is indeed equivalent to solving a linear program using the Dantzig-Wolfe decomposition. The proposed method will improve the connection between linear programming and algorithmic mechanism design.

Dantzig-Wolfe (DW) decomposition comprises a master problem and a subproblem. DW decomposition proceeds by solving the two problems in each iteration until the subproblem is not able to find any point which can contribute to the objective value of the master problem (Bazaraa et al., 2011). Since we are interested in integer points, we run a subroutine or an approximation algorithm which returns integer solutions as the subproblem. DW decomposition has been previously used for optimizing over a discrete set using branch and cut to obtain integer solutions (Desrosiers and Lübbecke, 2005). However, here we assume the existence of a subroutine which returns an

approximate integer solution of good quality and prove that the algorithm ends in optimality. To the best of our knowledge, this usage of DW decomposition in mechanism design has not been introduced before.

Dantzig-Wolfe decomposition is a variant of the revised simplex algorithm. A computational evaluation of the Dantzig-Wolfe decomposition has been done in Tebboth (2001). The study shows DW decomposition has a high performance, especially when a reasonable block structure can be found.

The resulting convex combination is tight in terms of the number of integer solutions in the convex decomposition according to the Carathéodory's theorem provided that the number of constraints is less than the dimension of the polytope.

Theorem 12 (Carathéodory)**.** *Given a polytope in* $\mathbb{R}^n$*, any point in the polytope is a convex combination of at most* $n+1$ *vertices of the polytope.*

By standard polyhedra theory, the number of non-zero variables in an extreme point is upper bounded by the number of constraints in the underlying linear program (see e.g. Chvátal (1983)). The proposed algorithm produces a convex combination of at most $m+1$ integer points, where m is the number of constraints, and thus the solution is tight in this sense. It is very common that the number of constraints is less than the number of variables. For example, in the relaxations of combinatorial auctions, this is usually the case because the bidders may obtain any package of items (for which one decision variable is needed) and there are exponentially many packages of items. Thus, given that $m < n$, the number of integer solutions will be at most $n+1$ which is tight according to the Carathéodory's theorem.

Sometimes, a fractional point - not necessarily an optimal - is calculated via other methods rather than linear programming. For example, a greedy algorithm may be used to find the fractional point since truthfulness can be guaranteed via the greedy algorithm but not with directly solving the linear program. See for example Chapter 4, and Dughmi and Ghosh (2010). Our method can find a decomposition into integer solutions for such readily present points.

Aside from the solution presented in Chapter 5, Elbassioni et al. (2015) present an alternative method for finding a convex decomposition of a given fractional point. Their method relies on the multiplicative weights update method which is a general technique for solving packing and covering problems (Arora et al., 2012; Khandekar, 2004). While the algorithm presented in Elbassioni et al. (2015) has a theoretical upper bound on the number of iterations, the algorithm is inferior to the presented method here in two aspects. First, their convex decomposition might have a size (the number of integer solutions) of $s(\lceil \epsilon^{-2} \ln s \rceil + 1)$, s being the number of non-zero components of the fractional point, and $\epsilon > 0$. Our solution will have a size of at

most $s+1$. Second, their convex decomposition can be as precise as $\frac{1}{1+4\epsilon}$ times the fractional solution at the expense of increasing runtime, while our convex decomposition is exact.

We also show how to apply the Benders decomposition to the problem. Benders decomposition is known to be the dual of the Dantzig-Wolfe decomposition technique (Bazaraa et al., 2011). We observe that sometimes working with the Benders decomposition has advantages over the DW decomposition.

6.1.2 *Structure*

In Section 6.2, we formally introduce the setting of the problem. In Section 6.3, we provide a short summary of DW decomposition technique. In Section 6.4, we establish our main result and show how the DW principle can be applied to our setting. Section 6.5 is devoted to the Benders decomposition applied to our setting. Section 6.6 discusses two applications of the adapted DW principle. Finally, in Section 6.7, we provide a numerical example for the adapted DW technique.

6.2 SETTING

Consider a finite set of integer points in $\mathbb{Z}^n_+$. Let Q denote the *convex hull* of all these points. That is Q defines a polytope with integral extreme points. Let $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \ \& \ \boldsymbol{x} \geq \boldsymbol{0}\}$ denote a polytope, where $\boldsymbol{A}$ is an m by n matrix, and $\boldsymbol{b}$ an m-dimensional column vector. A subroutine $\mathcal{A}$ for any cost function $\boldsymbol{c} \in \mathbb{R}^n$ returns an integer point $\mathbf{x} \in Q$ such that $\boldsymbol{c}\mathbf{x} \geq \boldsymbol{c}\boldsymbol{x}^*$, where $\boldsymbol{x}^* = \arg\max\{\boldsymbol{c}\boldsymbol{x} \mid \boldsymbol{x} \in P\}$. Equivalently, we say subroutine $\mathcal{A}$ will return for any cost vector $\boldsymbol{c}$ an integer point $\mathbf{x} \in Q$ such that $\boldsymbol{c}\mathbf{x} \geq \boldsymbol{c}\boldsymbol{x}$ for any $\boldsymbol{x}$ in P. Let $\mathcal{I}$ denote the index set for integer points in Q. The set of integer points in Q is therefore $\{\mathbf{x}_j\}_{j\in\mathcal{I}}$.

Usually, subroutine $\mathcal{A}$ only accepts non-negative cost vectors. Examples are approximation algorithms for NP-hard optimization problems. For instance, the approximation algorithm provided for the knapsack problem works with non-negative profits of items. However, in our setting, we expect $\mathcal{A}$ to work with any arbitrary cost vector. In such cases, an assumption that Q is a packing polytope is required: if $x \in Q$ and $y \leq x$ then $y \in Q$. See Lemma 16 in Chapter 5 for more information.

In this chapter, we address the following problem. Given a cost vector $\boldsymbol{c} \geq \boldsymbol{0}$, find values $\{\lambda^*_j \geq 0\}_{j\in\mathcal{I}}$ such that *i*) $\sum_{j\in\mathcal{I}} \lambda^*_j = 1$, and *ii*) $|\{\lambda^*_j \mid j \in \mathcal{I},\ \lambda^*_j > 0\}|$ is polynomial in m and n, and *iii*) $\sum_{j\in\mathcal{I}} \lambda^*_j \mathbf{x}_j = \boldsymbol{x}^*$, where $\boldsymbol{x}^* = \arg\max\{\boldsymbol{c}\boldsymbol{x} \mid \boldsymbol{x} \in P\}$.

Using the ellipsoid method, it can be shown that every point in P can be written as a convex combination of the extreme points in Q (Carr and Vempala, 2000; Lavi and Swamy, 2011). In this work, aside from answering the question above, we give an alternative proof for this fact.

6.3 SUMMARY OF DANTZIG-WOLFE DECOMPOSITION

Dantzig-Wolfe decomposition belongs to *column generation* techniques. We shall here briefly go over the Dantzig-Wolfe Decomposition. For a detailed explanation of the method we refer the reader to Bazaraa et al. (2011). Consider the following linear program.

$$\begin{aligned} \text{Minimize} \quad & \boldsymbol{cx} \\ \text{subject to} \quad & \boldsymbol{Ax} = \boldsymbol{b} \\ & \boldsymbol{x} \in X \end{aligned}$$

Where X is a bounded polyhedral of special structure, $\boldsymbol{A}$ is a $m \times n$ matrix, $\boldsymbol{c}$ is a n-dimensional vector and $\boldsymbol{b}$ is a m-dimensional vector.

Since X is a bounded polyhedra, then any point $\boldsymbol{x} \in X$ can be represented as a convex combination of a finite number of extreme points of X. Let us denote these points by $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_l$, and substitute $\boldsymbol{x}$ with its convex combination of extreme points, then the aforementioned LP can be transformed into the following program in which the variables are $\lambda_1, \lambda_2, \ldots, \lambda_l$.

$$\text{Minimize} \quad \sum_{j=1}^{l} (\boldsymbol{c}\boldsymbol{x}_j)\lambda_j \tag{9}$$

$$\text{subject to} \quad \sum_{j=1}^{l} (\boldsymbol{A}\boldsymbol{x}_j)\lambda_j = \boldsymbol{b} \tag{10}$$

$$\sum_{j=1}^{l} \lambda_j = 1, \tag{11}$$

$$\lambda_j \geq 0 \quad j = 1, 2, \ldots, l \tag{12}$$

The linear program $(9) - (12)$ is called the *master problem* and the program which finds an appropriate $\boldsymbol{x} \in X$ in each iteration is called the *subproblem*. Since the number of extreme points of set X is exponentially many, we follow the idea of *column generation* to find appropriate extreme point in each iteration. The information is passed back and forth between the master problem and the subproblem as follows. In each iteration a different cost coefficient is passed down by the master problem to the subproblem and the subproblem finds an extreme point $\boldsymbol{x}_k$, and sends it to the master problem.

Dantzig-Wolfe decomposition is an implementation of the *revised simplex* method. Let vector $\boldsymbol{w}$ and α denote the dual variables corresponding to equations (10) and (11), respectively. We first need an

BASIS	RHS
$(\boldsymbol{w}, \alpha)$	$\hat{\boldsymbol{c}}_B \bar{\boldsymbol{b}}$
$\boldsymbol{B}^{-1}$	$\bar{\boldsymbol{b}}$

Table 1: Simplex tableau. RHS stands for right-hand side.

initial solution to generate the simplex tableau. Suppose we have a basic feasible solution $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_B, \boldsymbol{\lambda}_N)$ to system $(10) - (12)$, where $\boldsymbol{\lambda}_B$ and $\boldsymbol{\lambda}_N$ denote the basic and non-basic variables, respectively. The initial $(m+1) \times (m+1)$ basis inverse $\mathbf{B}^{-1}$ hence will be known. The cost for each basic variable λ_j is in fact $\hat{c}_j = \boldsymbol{c}\boldsymbol{x}_j$. Therefore, we get $(\boldsymbol{w}, \alpha) = \hat{\boldsymbol{c}}_B \boldsymbol{B}^{-1}$, where $\hat{\boldsymbol{c}}_B$ is the cost vector of the basic variables. Denoting $\bar{\boldsymbol{b}} = \boldsymbol{B}^{-1} \begin{pmatrix} \boldsymbol{b} \\ 1 \end{pmatrix}$, we see the revised simplex tableau in Table 1.

The revised simplex proceeds by improving the current solution via finding an entering and a leaving variable. In other words, the set of basic and nonbasic variables exchange one element. When such an exchange is not possible then the current solution is optimal. The entering variable is in fact a variable λ_k associated with extreme point $\boldsymbol{x}_k$ for which $z_k - \hat{c}_k > 0$, where $z_k = (\boldsymbol{w}, \alpha) \begin{pmatrix} \boldsymbol{A}\boldsymbol{x}_k \\ 1 \end{pmatrix}$ and $\hat{c}_k = \boldsymbol{c}\boldsymbol{x}_k$.

We observe that $z_k - \hat{c}_k = (\boldsymbol{w}\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x}_k + \alpha$ denotes the value of point $\boldsymbol{x}_k$ with respect to current costs $\boldsymbol{w}\boldsymbol{A} - \boldsymbol{c}$ and dual variable α. In order to find such a point, we solve the following subproblem which gives us the required index or tells that the current solution is optimal when the maximum value is zero.

$$\begin{aligned} \text{Maximize} \quad & (\boldsymbol{w}\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x} + \alpha \\ \text{subject to} \quad & \boldsymbol{x} \in X \end{aligned}$$

Notice that the objective function contains a constant and therefore it can be replaced by $(\boldsymbol{w}\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x}$. Assuming that $\boldsymbol{x}_k$ is the optimal solution to the program above, the revised simplex method goes on as follows. If $z_k - \hat{c}_k = 0$ then the algorithm stops and the last solution to the master problem is an optimal of the overall problem.

If $z_k - \hat{c}_k > 0$ the master problem proceeds as follows. Let $\boldsymbol{y}_k = \boldsymbol{B}^{-1} \begin{pmatrix} \boldsymbol{A}\boldsymbol{x}_k \\ 1 \end{pmatrix}$, the entering column then will be $\begin{pmatrix} z_k - \hat{c}_k \\ \boldsymbol{y}_k \end{pmatrix}$. In order to find the leaving column, let index r be determined as follows:

$$\frac{\bar{b}_r}{y_{rk}} = \underset{1 \le i \le m+1}{\text{Minimum}} \left\{ \frac{\bar{b}_i}{y_{ik}} : y_{ik} > 0 \right\}.$$

We pivot at y_{rk} which will update the dual variables, the basis inverse, and the right-hand side. More specifically, *pivoting* on y_{rk} can be stated as follows.

1. Divide row r by y_{rk}.
2. For $i = 1, \ldots m$ and $i \neq r$, update the ith row by adding to it $-y_{ik}$ times the new rth row.
3. Update row zero by adding to it $z_k - \hat{c}_k$ times the new rth row.

After pivoting, the column λ_k is deleted and the algorithm repeats.

An important observation about the DW principle is as follows. The DW principle expects the Subproblem to return any point $\boldsymbol{x} \in X$ where $(\boldsymbol{w}\boldsymbol{A} - \boldsymbol{c})\boldsymbol{x} + \alpha > 0$ and it does not impose any other specific requirement on the selected point. We shall use this observation in our adaptation of the method.

Another observation is that, in each iteration, the master program finds the best solution using known extreme points. This is done in an organized manner as described above.

6.4 APPLYING DANTZIG-WOLFE DECOMPOSITION

A wide range of combinatorial optimization problems can be formulated using the integer program $\max\{\boldsymbol{c}\boldsymbol{x} \mid \boldsymbol{x} \in P \text{ and integer}\}$. Recall, $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \;\; \& \;\; \boldsymbol{x} \geq \boldsymbol{0}\}$. For example, in combinatorial auctions, $\boldsymbol{c}$ denotes the accumulated valuations of the players. The integer program therefore expresses the welfare maximization objective subject to feasibility constraints encoded as P.

Usually, using simplex method or other standard linear programming techniques, first a relaxed linear program of the integer program above is solved:

$$\begin{aligned} \text{Maximize} \quad & \boldsymbol{c}\boldsymbol{x} && (13)\\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} && (14)\\ & \boldsymbol{x} \geq \boldsymbol{0} && (15) \end{aligned}$$

Notice, constraints (14) and (15) together are equivalent to $\boldsymbol{x} \in P$. Next, the solution is rounded to an integer solution at the expense of a value loss or slightly violating the constraints. Given subroutine $\mathcal{A}$, we wish to find a solution to the linear program above as well as a convex decomposition of it into integer points. We wish to achieve both goals at once. Recall, subroutine $\mathcal{A}$ returns for any cost vector $\boldsymbol{c}$ an integer point $\mathbf{x} \in Q$ such that $\boldsymbol{c}\mathbf{x} \geq \boldsymbol{c}\boldsymbol{x}$ for any $\boldsymbol{x}$ in P.

Generally speaking, Dantzig-Wolfe principle uses the fact that if we relax some constraints and obtain a simpler polyhedra then the solution to the original problem can be written as a convex combination of extreme points of the simpler polyhedra. The simplicity refers to the fact that the extreme points of the new polyhedra can be found more easily than those of the original problem. While Dantzig-Wolfe principle is useful when the underlying constraints are decomposable into

simpler regions, we use it in a slightly different manner by looking at Q as the polyhedra over which we can efficiently optimize. Recall, Q denotes the convex hull of a finite set of integer points. To apply the idea of Dantzig-Wolfe decomposition to the problem $(13) - (15)$, we add a new constraint to the program.

$$\boldsymbol{x} \in Q \tag{16}$$

We will show that an optimal solution to problem $(13) - (16)$ is also an optimal solution to $(13) - (15)$, and thus adding the new constraint is harmless. We represent $\boldsymbol{x} \in Q$ as a convex combination of extreme points of Q. Recall, $\mathcal{I}$ denote the index set for integer points in Q, and $\{\mathbf{x}_j\}_{j\in\mathcal{I}}$ is the set of integer points in Q. Substitute $\boldsymbol{x}$ with its convex combination of extreme points of Q, then program $(13) - (16)$ can be transformed into the following program where the variables are $\{\lambda_j\}_{j\in\mathcal{I}}$.

$$\begin{aligned}
\text{Maximize} \quad & \sum_{j\in\mathcal{I}} (\boldsymbol{c}\mathbf{x}_j)\lambda_j && (17)\\
\text{subject to} \quad & \sum_{j\in\mathcal{I}} (\boldsymbol{A}\mathbf{x}_j)\lambda_j + \boldsymbol{s} = \boldsymbol{b} && (18)\\
& \sum_{j\in\mathcal{I}} \lambda_j = 1 && (19)\\
& \lambda_j \geq 0 \quad \forall j \in \mathcal{I} && (20)\\
& \boldsymbol{s} \geq \mathbf{0}. && (21)
\end{aligned}$$

The linear program $(17) - (21)$ is the *master problem*. Notice, we have added slack variables $\boldsymbol{s} \in \mathbb{R}^m_+$ to convert the inequalities into equality as needed by the DW principle (see Section 6.3). The DW *subproblem* is defined below.

$$\max_{j\in\mathcal{I}} (\boldsymbol{c} + \boldsymbol{w}\boldsymbol{A})\mathbf{x}_j \tag{22}$$

While our problem is a maximization problem, the procedure in Section 6.3 is presented for a minimization problem, thus we substitute the $-\boldsymbol{c}$ with $\boldsymbol{c}$ in the objective function of the subproblem. That means, we change the objective function of the master problem from $\max \sum_{j\in\mathcal{I}} (\boldsymbol{c}\mathbf{x}_j)\lambda_j$ to $\min \sum_{j\in\mathcal{I}} (-\boldsymbol{c}\mathbf{x}_j)\lambda_j$, and apply the theory provided in Section 6.3.

We assume $\mathbf{0} \in Q$ thus the initial basic solution is simply defined by letting λ_0, the variable corresponding to point $\mathbf{x} = \mathbf{0}$, equal 1 ($\lambda_0 = 1$), and letting $\boldsymbol{s} = \boldsymbol{b}$. Assuming that program (22) can be efficiently solved for any cost vector, then we are exactly following the DW principle, and therefore, we can successfully solve the overall problem as DW principle does this. However, program (22) is an integer program and solving it may not be computationally tractable.

To address this issue, we propose using subroutine $\mathcal{A}$ to approximate program (22). Let $(\bar{w}, \bar{\alpha})$ be the last dual variables calculated by the master program. In each iteration, we call subroutine $\mathcal{A}$ with current cost vector $\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A}$ to find a point $\mathbf{x}_k \in Q$ to pass to the master problem. This substitution seemingly comes at the expense of stopping at a local optimum, as explained in the following.

As long as the algorithm continues by using the points returned by subroutine $\mathcal{A}$, we are exactly running DW principle. Recall the important observation that in DW principle, the subproblem need not be completely optimized and any point $\mathbf{x}_k$ with $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k + \bar{\alpha} > 0$ suffices to proceed. However, there might be an iteration in which there exists a point $\mathbf{x}' \in Q$ for which we have $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}' + \bar{\alpha} > 0$, but for the integer point $\mathbf{x}_k$ returned by subroutine $\mathcal{A}$, we have $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k + \bar{\alpha} \leq 0$. Therefore, DW stops at a suboptimal point. Nevertheless, below, we argue that this cannot happen. That means as long as DW has not reached the optimum to problem $(13) - (15)$, subroutine $\mathcal{A}$, given the current cost vector $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})$, returns a point $\mathbf{x}_k$ with $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k + \bar{\alpha} > 0$.

Let $\boldsymbol{x}^*$ denote the optimal solution to program $(13) - (15)$. It is instructive to see what the master step would do if the subproblem passes the point $\boldsymbol{x}^*$, rather than an integer point, to the master problem. While we do not know such an optimal point, but we know that such a point exists and this suffices for our reasoning. We argue that the master step will set $\boldsymbol{\lambda}_{x^*} = 1$ and $\boldsymbol{\lambda}_j = 0$ for all other $\boldsymbol{\lambda}_j$'s which are currently in the base. In other words, the master program returns the best possible convex combination which is in fact $\lambda_{x^*} = 1$. This is discussed in the following observation.

Observation 3. *Let $\boldsymbol{x}^*$ denote the optimal solution to program $(13) - (15)$. If supposedly the subproblem in any iteration passes $\boldsymbol{x}^*$ to the master problem, then the master step will set $\boldsymbol{\lambda}_{x^*} = 1$ and $\boldsymbol{\lambda}_j = 0$ for all other $\boldsymbol{\lambda}_j$'s which are currently in the base.*

Proof. First, we observe that if the first subproblem (right after the initialization) passes $\boldsymbol{x}^*$ to the master problem, then the master step will set $\boldsymbol{\lambda}_{x^*} = 1$ and $\boldsymbol{\lambda}_0 = 0$. Clearly, in the solution $\boldsymbol{s}$ needs to be evaluated accordingly. Second, by looking more closely at what simplex does in each iteration, we observe that in any further iteration, if the subproblem passes $\boldsymbol{x}^*$ to the master problem, the master step will set $\boldsymbol{\lambda}_{x^*} = 1$ and $\boldsymbol{\lambda}_j = 0$ for all other $\boldsymbol{\lambda}_j$'s.

The simplex method, in each iteration, performs a set of row operations on the constraints when it pivots (see pivoting steps in Section 6.3). The constraints, in any iteration, are thus the initial constraints after a series of row operations. This will certify that the aforementioned solution ($\boldsymbol{\lambda}_{x^*} = 1$) will be feasible in any further iteration. If the subproblem passes $\boldsymbol{x}^*$ to the master problem, our entering variable will be $\boldsymbol{\lambda}_{x^*}$. The simplex algorithm then increases the entering

variable λ_{x^*} as much as one basic variable gets zero. However, as discussed, the solution $\lambda_{x^*} = 1$ is feasible, and it is possible to increase λ_{x^*} up to 1 and set all other λ_j's to zero. The algorithm will behave as such to produce the highest increase in the objective value, the desired conclusion. □

Theorem 13. *If the subproblem* (22) *calls subroutine* $\mathcal{A}$ *to return an integer point in each iteration, the DW principle never stops until it gets to an optimal solution to problem* (13) − (15).

Proof. Let x^* denote the optimal solution to program (13) − (15). Assume the algorithm stops at a suboptimal point: $\sum_{j\in\mathcal{I}}(cx_j)\lambda_j < cx^*$. Let $(\bar{w}, \bar{\alpha})$ be the last dual variables calculated by the master program. Let x_k be the point returned by subroutine $\mathcal{A}$, given cost vector $(c + \bar{w}A)$, in the last iteration. We must have $(c + \bar{w}A)x_k + \bar{\alpha} \leq 0$ because DW has stopped.

If supposedly the subproblem in the last iteration passes x^* to the master problem, according to Observation 3, the master step will increase λ_{x^*} up to 1 and set all other λ_j's to zero. Since we assumed the algorithm has stopped at a suboptimal point, by setting $\lambda_{x^*} = 1$ the objective value will increase. If entering λ_{x^*} improves the objective value, we must have $(c + \bar{w}A)x^* + \bar{\alpha} > 0$ from the theory of DW principle provided in Section 6.3: if $(c + \bar{w}A)x^* + \bar{\alpha} \leq 0$ then entering variable λ_{x^*} cannot improve the objective value.

By the property of the subroutine, we have $(c + \bar{w}A)x_k \geq (c + \bar{w}A)x^*$. Thus, in the last iteration, we must have $(c + \bar{w}A)x_k + \bar{\alpha} > 0$. This contradicts our assumption that $(c + \bar{w}A)x_k + \bar{\alpha} \leq 0$. Consequently, as long as we have not reached the optimum, the subroutine returns a point x_k with $(c + \bar{w}A)x_k + \bar{\alpha} > 0$. This completes the proof. □

We draw the conclusion that substituting program (22) with subroutine $\mathcal{A}$ is harmless. Therefore, we have shown that finding a convex decomposition of maximum value is indeed equivalent to solving a linear program via DW principle. Let us call the method *integer DW*.

6.5 BENDERS DECOMPOSITION

It is known that the Dantzig-Wolfe Decomposition has an equivalent decomposition technique namely *Benders decomposition* (Bazaraa et al., 2011). Benders decomposition is a *row generation technique* in contrast with the Dantzig-Wolfe column generation procedure. Sometimes, working with Benders decomposition has advantages over Dantzig-Wolfe decomposition. We explain how to apply the Benders algorithm to our problem. Later, we discuss the advantages of the method.

Recall, polytope Q is a bounded polyhedra. Hence, there exist matrix $D \in \mathbb{R}^{m' \times n}$ and $d \in \mathbb{R}^{m'}$ such that $Q = \{x \in \mathbb{R}^n \mid Dx \leq d \ \& \ x \geq$

$\mathbf{0}\}$. We add constraint $\boldsymbol{x} \in Q$ to program $(13) - (15)$ and work with the new program. We will see that adding this constraint has no influence on the region of feasible solutions to program $(13) - (15)$. Following the procedure described in Bazaraa et al. (2011), we can write the Benders decomposition for this new program. The *Benders master problem* will be as follows.

$$\begin{aligned} \text{Maximize} \quad & z && (23)\\ \text{subject to} \quad & z \leq \boldsymbol{wb} - (\boldsymbol{c} + \boldsymbol{wA})\mathbf{x}_j \quad \forall j \in \mathcal{I} && (24)\\ & \boldsymbol{w} \leq \mathbf{0} && (25)\\ & z \quad \text{unrestricted.} && (26) \end{aligned}$$

The variables of the master problem are z and $\boldsymbol{w}$. Variable vector $\boldsymbol{w}$ is the vector of dual variables associated to the constraints (14). The Benders master problem has exponentially many constraints, thus it is inconvenient to solve directly. Hence, we maintain only a few of the constraints (24). Assuming $\mathbf{0} \in Q$, we start with only one constraint: $z \leq \boldsymbol{wb} - (\boldsymbol{c} + \boldsymbol{wA})\mathbf{0} = \boldsymbol{wb}$. Notice, we can use any $\mathbf{x}_j \in Q$ to start with. We solve the master problem and let $(\bar{z}, \bar{\boldsymbol{w}})$ be the solution. The value of $\bar{z}$ is an upper bound on the optimal value to the master problem. If $(\bar{z}, \bar{\boldsymbol{w}})$ satisfies constraints (24) for all $j \in \mathcal{I}$, then $(\bar{z}, \bar{\boldsymbol{w}})$ is optimal for the master problem. We can check constraints (24) by examining if $\bar{z} \leq \bar{\boldsymbol{w}}\boldsymbol{b} - \max_{j\in\mathcal{I}}(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_j$.

Thus, the *Benders subproblem* will be as the following.

$$\max_{j\in\mathcal{I}} (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_j \tag{27}$$

Note that the Benders subproblem is also the subproblem solved by the Dantzig-Wolfe decomposition. Furthermore, the Benders master problem is the dual to the Dantzig-Wolfe master problem $(17) - (21)$. The Benders subproblem is solved by calling subroutine $\mathcal{A}$ with cost vector $\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A}$. If the subproblem returns $\mathbf{x}_k$ that violates the constraints (24): $\bar{z} > \bar{\boldsymbol{w}}\boldsymbol{b} - (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k$, we can generate the new constraint $z \leq \boldsymbol{wb} - (\boldsymbol{c} + \boldsymbol{wA})\mathbf{x}_k$, and add it to the current master program, and reoptimize. We repeat this process until the solution returned by subroutine $\mathcal{A}$ does not violate the constraints. We claim that at this iteration, the value of $\bar{z}$ is the optimal value to the master problem.

The Benders decomposition provides a more concise proof that the decomposition techniques in companion with the subroutine $\mathcal{A}$ work correctly.

Theorem 14. *If the subproblem (27) calls subroutine $\mathcal{A}$ to return an integer point in each iteration, the Benders algorithm never stops until it reaches an optimal solution to problem* $(13) - (15)$.

Proof. Let $\boldsymbol{x}^*$ denote an optimal solution to problem $(13) - (15)$. Let z^* denote an optimal value of the Benders master problem. We have

$z^* = (-\boldsymbol{c})\boldsymbol{x}^*$ by the construction of the Benders master problem, and the duality theorem. Let $\bar{z}$ be the final solution to the master problem and $\mathbf{x}_k$ the solution returned by subroutine $\mathcal{A}$ when the Benders algorithm stops. Since the algorithm stops, we must have $\bar{z} \leq \bar{\boldsymbol{w}}\boldsymbol{b} - (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k$. We always have $z^* \leq \bar{z}$ because $\bar{z}$ is an upper bound on the optimal solution to the master problem. Assume $\bar{z}$ is not optimal: $\bar{z} > z^*$. Remember, by the definition of subroutine $\mathcal{A}$, we have $(\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k \geq (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\boldsymbol{x}^*$. Therefore,

$$
\begin{array}{lll}
 & \boldsymbol{A}\boldsymbol{x}^* \leq \boldsymbol{b} & \text{since } \boldsymbol{x}^* \text{ is a solution to problem } (13)-(15) \\
\Rightarrow & \bar{\boldsymbol{w}}\boldsymbol{A}\boldsymbol{x}^* \geq \bar{\boldsymbol{w}}\boldsymbol{b} & \text{since } \bar{\boldsymbol{w}} \leq \mathbf{0} \\
\Rightarrow & -\boldsymbol{c}\boldsymbol{x}^* \geq \bar{\boldsymbol{w}}\boldsymbol{b} - \boldsymbol{c}\boldsymbol{x}^* - \bar{\boldsymbol{w}}\boldsymbol{A}\boldsymbol{x}^* & \\
\Rightarrow & -\boldsymbol{c}\boldsymbol{x}^* \geq \bar{\boldsymbol{w}}\boldsymbol{b} - (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k & \text{since } (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k \geq (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\boldsymbol{x}^* \\
\Rightarrow & \bar{z} > \bar{\boldsymbol{w}}\boldsymbol{b} - (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k & \text{since } \bar{z} > z^* = -\boldsymbol{c}\boldsymbol{x}^*
\end{array}
$$

But, this contradicts $\bar{z} \leq \bar{\boldsymbol{w}}\boldsymbol{b} - (\boldsymbol{c} + \bar{\boldsymbol{w}}\boldsymbol{A})\mathbf{x}_k$. The contradiction arises from the assumption that $\bar{z}$ is not optimal. Thus, when the algorithm stops, we have the optimal solution, the desired conclusion. □

After solving the Benders master problem, we can use the provided integer solutions and solve the restricted primal to obtain a convex decomposition.

Another advantage of the Benders decomposition arises from the fact that in each iteration, we optimize an LP in the master step. Solving an LP is sometimes more convenient than the implementation of the pivoting steps done in each iteration in the master step of the DW principle.

It is instructive to note that Theorem 14 implies that if the integer solution returned by subroutine $\mathcal{A}$ does not violate constraints (24), then the current master solution is optimal. Exploiting this fact, we can use the ellipsoid method to solve problem $(23)-(26)$ to certify a polynomial runtime which might be of theoretical interest. To use the ellipsoid method, we need to implement a *separation oracle*. Recall that a separation oracle, given a solution, either confirms that it is a feasible solution, or returns the constraint violated by the solution. Using subroutine $\mathcal{A}$ as the separation oracle, as long as we find a violated constraint, we cut the current ellipsoid and continue. When the subroutine $\mathcal{A}$ cannot return a violating constraint, according to Theorem 14, the algorithm has reached the optimum.

6.6 APPLICATION OF THE METHOD IN MECHANISM DESIGN

6.6.1 *The Framework Proposed by Lavi and Swamy*

Let $X = \left\{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \;\&\; \boldsymbol{x} \geq \mathbf{0}\right\}$ denote the underlying polytope of a linear program, and $\boldsymbol{x}^*$ denote an optimal solution to the program

with respect to some cost vector. The maximum ratio between the value of an integer program and its relaxation, with respect to all cost vectors, is called the integrality gap of the relaxation. Assuming that, the integrality gap of X is $\beta \geq 1$, and that a β integrality-gap-verifier is given, Lavi and Swamy propose a method to decompose the scaled-down fractional solution $\frac{x^*}{\beta}$ into a convex combination of integer solutions (Lavi and Swamy, 2011). A β integrality-gap-verifier is an algorithm that, given any cost vector, returns an integer solution whose value is at least $1/\beta$ times the optimal relaxed solution.

This decomposition technique was originally observed by Carr and Vempala (2000), and later adapted by Lavi and Swamy to mechanism design problems provided that the underlying polytope of the relaxation of the problem has the packing property. The approach requires only a polynomial number of calls to the integrality-gap-verifier with respect to the number of positive components in x^*. Yet, the approach strongly relies on the ellipsoid method, and hence it is more of theoretical importance than of practical use. We refer the reader to Chapter 5 for more details about the framework developed by Lavi and Swamy (LS framework).

In order to view the LS framework in our setting, the integrality-gap-verifier is used as subroutine $\mathcal{A}$ and $X/\beta = \{\boldsymbol{x} \mid \beta\boldsymbol{x} \in X\}$ is treated as P in our setting introduced in Section 6.2. This way, the integer DW finds the maximum value in X/β as well as its decomposition into integer points, both in one step. This improves upon other implementations of the LS framework which require two steps to find the convex decomposition (Lavi and Swamy, 2011; Kraft et al., 2014; Elbassioni et al., 2015).

It is instructive to note that solving program $(17) - (21)$, essentially defines a Maximal-In-Distributional-Range (MIDR) allocation rule. An MIDR algorithm fixes a set of distributions over feasible solutions (the distributional range) independently of the valuations reported by the self-interested players, and outputs a random sample from the distribution that maximizes expected (reported) welfare (Dobzinski and Dughmi, 2009). Here, we optimize over a range which is independent of bidder's private information. The range is in fact the feasible region of the program: all probability distributions over integer solutions which satisfy constraints $(18) - (21)$. The range is obviously independent of bidders' valuations.

6.6.2 *Existing Fractional Point*

Sometimes a fractional point $\boldsymbol{x}^* \in Q$ is present, and we wish to find a convex decomposition of $\boldsymbol{x}^*$ into extreme points of Q. This can happen when we use other methods to find a fractional point rather than linear programming. Here, we assume that Q satisfy the packing property.

For this case, we can use the integer DW as follows. Define $P = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \leq \boldsymbol{x}^* \;\&\; \boldsymbol{x} \geq 0\}$ and let $\boldsymbol{c} = \boldsymbol{x}^*$. Now, apply the integer DW. All arguments follow accordingly, assuming that a subroutine $\mathcal{A}$ with the following property is available. Subroutine $\mathcal{A}$ will return for any cost vector $\boldsymbol{c}$ an integer point $\mathbf{x} \in Q$ such that $\boldsymbol{c}\mathbf{x} \geq \boldsymbol{c}\boldsymbol{x}^*$. Because the number of constraints in P is at most n, the resulting convex decomposition in this case is tight in terms of the number of integer points, according to the Carathéodory's theorem.

6.7 NUMERICAL EXAMPLE FOR INTEGER DW

In this section, we focus on applying the integer DW to an instance of multi-unit auctions. In multi-unit auctions, there is a set of m identical items and a set of players. Each player i has a valuation for any number of items denoted by $v_i(j)$ for getting j items where $1 \leq j \leq m$. The goal is to maximize social welfare by distributing items among bidders.

The LP relaxation for this class of problems is as follows. Let x_{ij} denote if j units is assigned to bidder i.

$$\text{Maximize} \quad \sum_{i,j} v_i(j) x_{ij} \tag{MU-P}$$

$$\text{subject to} \quad \sum_{j} x_{ij} \leq 1 \quad \text{for each player } i \tag{28}$$

$$\sum_{i,j} j \cdot x_{ij} \leq m \tag{29}$$

$$0 \leq x_{ij} \leq 1 \quad \text{for each } i, j \tag{30}$$

Lavi and Swamy present a greedy algorithm which returns for any valuation v an integer solution that is at least as good as half of the optimal fractional solution to MU-P with respect to v (Lavi and Swamy, 2011). Thus, we have a 2 integrality-gap-verifier algorithm for MU-P. This greedy algorithm will serve as the subroutine in the integer DW, and is called $\mathcal{A}$.

We give a short example to demonstrate the proposed convex decomposition method. Suppose a simple multi-unit auction with 3 players and 4 identical items. The following valuation vectors $v_i(j)$ are given for each player i and quantity j:

$$\begin{array}{lcccccc} j & & 1 & 2 & 3 & 4 & \\ v_1(j) = (& 6 & 6 & 6 & 6 &) \\ v_2(j) = (& 1 & 4 & 4 & 6 &) \\ v_3(j) = (& 0 & 1 & 1 & 1 &) \end{array}$$

We can reproduce program $(17) - (21)$ for this instance as follows. Let $\mathcal{I}$ denote the index set of integer points which satisfy inequalities

(28) − (30).

Let $\boldsymbol{c} = \begin{bmatrix} 6 & 6 & 6 & 6 & 1 & 4 & 4 & 6 & 0 & 1 & 1 & 1 \end{bmatrix}$, $\boldsymbol{b} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 2 \end{bmatrix}$, and $\boldsymbol{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{bmatrix}$. Notice the integrality gap has been reflected in defining $\boldsymbol{b}$.

Initialization Step

Let the starting basis consist of $\boldsymbol{s}$ and λ_0 where $\mathbf{x}_0 = \vec{0}$ is the starting integer point. Therefore, the first simplex tableau is as the following.

	BASIS INVERSE					RHS
z	0	0	0	0	0	0
s_1	1	0	0	0	0	.5
s_2	0	1	0	0	0	.5
s_3	0	0	1	0	0	.5
s_4	0	0	0	1	0	2
λ_0	0	0	0	0	1	1

Iteration 1

SUBPROBLEM. From the simplex tableau, we have $\boldsymbol{w} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$ and $\alpha = 0$. As a result, $\boldsymbol{wA} + \boldsymbol{c} = \boldsymbol{c}$. The subproblem therefore is $\max_{j \in \mathcal{I}} \boldsymbol{c}\mathbf{x}_j$. Subroutine $\mathcal{A}$ returns $\mathbf{x}$ such that $\mathbf{x}_{11} = \mathbf{x}_{22} = 1$ and all other entries of $\mathbf{x}$ are zero. The objective of the point with respect to current cost is $z - \hat{c} = 10 > 0$. Let us call this point $\mathbf{x}_1$.

MASTER PROBLEM.

$\boldsymbol{A}\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \end{bmatrix}$. Then $\boldsymbol{y}_1 = \boldsymbol{B}^{-1} \begin{bmatrix} \boldsymbol{A}\mathbf{x}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 3 \\ 1 \end{bmatrix}$.

Now, we insert the column into the foregoing tableau and pivot. Variable s_1 leaves the basis and λ_1 enters the basis.

	BASIS INVERSE					RHS	λ_1
z	0	0	0	0	0	0	10
s_1	1	0	0	0	0	.5	$\underline{\mathbf{1}}$
s_2	0	1	0	0	0	.5	1
s_3	0	0	1	0	0	.5	0
s_4	0	0	0	1	0	2	3
λ_0	0	0	0	0	1	1	1

After pivoting we obtain the following tableau.

	BASIS INVERSE					RHS
z	-10	0	0	0	0	-5
λ_1	1	0	0	0	0	.5
s_2	-1	1	0	0	0	0
s_3	0	0	1	0	0	.5
s_4	-3	0	0	1	0	.5
λ_0	-1	0	0	0	1	.5

The best-known feasible solution of the overall problem is given by $\lambda_0 \mathbf{x}_0 + \lambda_1 \mathbf{x}_1 = 0.5\mathbf{x}_0 + 0.5\mathbf{x}_1$. The current objective value is 5.

Iteration 2

SUBPROBLEM. From the simplex tableau, we have $\boldsymbol{w} = \begin{bmatrix}-10 & 0 & 0 & 0\end{bmatrix}$ and $\alpha = 0$. As a result, $\boldsymbol{wA} + \boldsymbol{c} = \begin{bmatrix}-4 & -4 & -4 & -4 & 1 & 4 & 4 & 6 & 0 & 1 & 1 & 1\end{bmatrix}$. The subproblem therefore is $\max_{j \in \mathcal{I}} (\boldsymbol{wA} + \boldsymbol{c})\mathbf{x}_j$. Subroutine $\mathcal{A}$ returns $\mathbf{x}$ such that $\mathbf{x}_{24} = 1$ and all other entries of $\mathbf{x}$ are zero. The objective of the point with respect to current cost is $z - \hat{c} = 6 > 0$. Let us call this point $\mathbf{x}_2$.

MASTER PROBLEM.

$$\boldsymbol{A}\mathbf{x}_2 = \begin{bmatrix}0\\1\\0\\4\end{bmatrix}. \text{ Then } \boldsymbol{y}_2 = \boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{A}\mathbf{x}_2\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\4\\1\end{bmatrix}.$$

Now, we insert the column into the foregoing tableau and pivot. Variable s_2 leaves the basis and λ_2 enters the basis.

	BASIS INVERSE					RHS	λ_2
z	-10	0	0	0	0	-5	6
λ_1	1	0	0	0	0	.5	0
s_2	-1	1	0	0	0	0	$\underline{\mathbf{1}}$
s_3	0	0	1	0	0	.5	0
s_4	-3	0	0	1	0	.5	4
λ_0	-1	0	0	0	1	.5	1

After pivoting we obtain the following tableau.

	BASIS INVERSE					RHS
z	-4	-6	0	0	0	-5
λ_1	1	0	0	0	0	.5
λ_2	-1	1	0	0	0	0
s_3	0	0	1	0	0	.5
s_4	1	-4	0	1	0	.5
λ_0	0	-1	0	0	1	.5

The best-known feasible solution of the overall problem is given by $\lambda_0 \mathbf{x}_0 + \lambda_1 \mathbf{x}_1 = 0.5\mathbf{x}_0 + 0.5\mathbf{x}_1$. The current objective value is 5.

Iteration 3

SUBPROBLEM. From the simplex tableau, we have $\boldsymbol{w} = \begin{bmatrix}-4 & -6 & 0 & 0\end{bmatrix}$ and $\alpha = 0$. As a result, $\boldsymbol{wA} + \boldsymbol{c} = \begin{bmatrix}2 & 2 & 2 & 2 & -5 & -2 & -2 & 0 & 0 & 1 & 1 & 1\end{bmatrix}$. The subproblem therefore is $\max_{j\in\mathcal{I}}(\boldsymbol{wA} + \boldsymbol{c})\mathbf{x}_j$. Subroutine $\mathcal{A}$ returns $\mathbf{x}$ such that $\mathbf{x}_{11} = 1$, $\mathbf{x}_{32} = 1$ and all other entries of $\mathbf{x}$ are zero. The objective of the point with respect to current cost is $z - \hat{c} = 3 > 0$. Let us call this point $\mathbf{x}_3$.

MASTER PROBLEM.

$$\boldsymbol{A}\mathbf{x}_3 = \begin{bmatrix}1\\0\\1\\3\end{bmatrix}. \text{ Then } \boldsymbol{y}_2 = \boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{A}\mathbf{x}_3\\1\end{bmatrix} = \begin{bmatrix}1\\-1\\1\\4\\1\end{bmatrix}.$$

Now, we insert the column into the foregoing tableau and pivot. Variable s_4 leaves the basis and λ_3 enters the basis.

	BASIS INVERSE					RHS	λ_3
z	-4	-6	0	0	0	-5	3
λ_1	1	0	0	0	0	.5	1
λ_2	-1	1	0	0	0	0	-1
s_3	0	0	1	0	0	.5	1
s_4	1	-4	0	1	0	.5	$\underline{\mathbf{4}}$
λ_0	0	-1	0	0	1	.5	1

After pivoting we obtain the following tableau.

	BASIS INVERSE					RHS
z	-4.75	-3	0	$-.75$	0	-5.375
λ_1	.75	1	0	$-.25$	0	.375
λ_2	$-.75$	0	0	.25	0	.125
s_3	$-.25$	1	1	$-.25$	0	.375
λ_3	.25	-1	0	.25	0	.125
λ_0	$-.25$	0	0	$-.25$	1	.375

The best-known feasible solution of the overall problem is given by $\lambda_0\mathbf{x}_0 + \lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \lambda_3\mathbf{x}_3 = 0.375\mathbf{x}_0 + 0.375\mathbf{x}_1 + 0.125\mathbf{x}_2 + 0.125\mathbf{x}_3$. The current objective value is 5.375.

Iteration 4

SUBPROBLEM. From the simplex tableau, we have $\boldsymbol{w} = \begin{bmatrix}-4.75 & -3 & 0 & -.75\end{bmatrix}$ and $\alpha = 0$. As a result,
$\boldsymbol{wA} + \boldsymbol{c} = \begin{bmatrix}.5 & -.25 & -1 & -1.75 & -2.75 & -.5 & -1.25 & 0 & -.75 & -.5 & -1.25 & -2\end{bmatrix}$.
The subproblem therefore is $\max_{j\in\mathcal{I}}(\boldsymbol{wA} + \boldsymbol{c})\mathbf{x}_j$. Subroutine $\mathcal{A}$ returns $\mathbf{x}$ such that $\mathbf{x}_{11} = 1$ and all other entries of $\mathbf{x}$ are zero. The objective of the point with respect to current cost is $z - \hat{c} = 0.5 > 0$. Let us call this point $\mathbf{x}_4$.

MASTER PROBLEM.

$$\boldsymbol{A}\mathbf{x}_4 = \begin{bmatrix}1\\0\\0\\1\end{bmatrix}. \text{ Then } \boldsymbol{y}_2 = \boldsymbol{B}^{-1}\begin{bmatrix}\boldsymbol{A}\mathbf{x}_4\\1\end{bmatrix} = \begin{bmatrix}.5\\-.5\\-.5\\.5\\.5\end{bmatrix}.$$

Now, we insert the column into the foregoing tableau and pivot. Variable λ_3 leaves the basis and λ_4 enters the basis.

	BASIS INVERSE					RHS	λ_4
z	−4.75	−3	0	−.75	0	−5.375	.5
λ_1	.75	1	0	−.25	0	.375	.5
λ_2	−.75	0	0	.25	0	.125	−.5
s_3	−.25	1	1	−.25	0	.375	−.5
λ_3	.25	−1	0	.25	0	.125	**.5**
λ_0	−.25	0	0	−.25	1	.375	.5

After pivoting we obtain the following tableau.

	BASIS INVERSE					RHS
z	−5	−2	0	−1	0	−5.5
λ_1	.5	−2	0	−.5	0	.25
λ_2	−.5	−1	0	.5	0	.25
s_3	0	0	1	0	0	.5
λ_4	.5	−2	0	.5	0	.25
λ_0	−.5	1	0	−.5	1	.25

The best-known feasible solution of the overall problem is given by $\lambda_0\mathbf{x}_0 + \lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \lambda_4\mathbf{x}_4 = 0.25\mathbf{x}_0 + 0.25\mathbf{x}_1 + 0.25\mathbf{x}_2 + 0.25\mathbf{x}_4$. The current objective value is 5.5.

Iteration 5

SUBPROBLEM. From the simplex tableau, we have $\boldsymbol{w} = \begin{bmatrix}-5 & -2 & 0 & -1\end{bmatrix}$ and $\alpha = 0$. As a result,
$\boldsymbol{w}\boldsymbol{A} + \boldsymbol{c} = \begin{bmatrix}-6 & -7 & -8 & -9 & -3 & -4 & -5 & -6 & -1 & -2 & -3 & -4\end{bmatrix}$. The subproblem therefore is $\max_{j\in\mathcal{I}}(\boldsymbol{w}\boldsymbol{A} + \boldsymbol{c})\mathbf{x}_j$. Subroutine $\mathcal{A}$ returns $\mathbf{x} = \vec{0}$. The objective of the point with respect to current cost is $z - \hat{c} = 0$. Therefore, the algorithm terminates. Our final solution is as follows.

$$\boldsymbol{x}^* = \begin{bmatrix}x_{11}\\x_{22}\\x_{24}\end{bmatrix} = 0.25\begin{bmatrix}1\\1\\0\end{bmatrix} + 0.25\begin{bmatrix}0\\0\\1\end{bmatrix} + 0.25\begin{bmatrix}1\\0\\0\end{bmatrix} + 0.25\begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix}0.5\\0.25\\0.25\end{bmatrix}.$$

A simple examination shows that $\boldsymbol{x}^*$ is in fact one half (scaled down by the integrality gap) of the optimal solution to MU-P for our instance.

7

CONCLUSION

Imagine a particular context such as auctions where pieces of data that we need for an optimization problem are owned by strategic agents. In this context, neither sophisticated nor simple optimization algorithms are useful if the agents misreport the pieces of data. Therefore, we have to solve the optimization problem while certifying that the algorithm provides the agents with *incentives* to truthfully report data. In economics, the Vickrey-Clarke-Groves (VCG) mechanism provides both objectives at once. For *NP-hard* problems, however, the VCG mechanism is not directly applicable, thus we have to design dominant strategy incentive-compatible (DSIC) approximation algorithms with polynomial runtime. The field of *algorithmic mechanism design* attempts to provide the compromise between polynomial-time algorithms and incentives.

In Chapter 3, we proposed a truthful approximation algorithm for a strategic variant of the generalized assignment problem in which valuations are private. The generalized assignment problem is a notable problem in combinatorial optimization and operations research. Our truthful algorithm has advantages over existing approximation algorithms for GAP in terms of runtime and simplicity while the presented approximation ratio closely matches the best approximation ratio previously presented for the problem with public valuations.

Application of algorithms in practice stipulates fast, simple and incentive-compatible implementations. Performance and simplicity motivate the redesign of existing algorithms towards more practical implementations. For example, there have been attempts to substitute the usage of the ellipsoid method with practically faster algorithms. See chapters 5 and 6, and Elbassioni et al. (2015). Furthermore, incentive compatibility in practice deserves to be placed under scrutiny. Since many algorithms are not readily incentive-compatible, it will be useful to design a notion of "higher incentive compatibility" to characterize algorithms which are more robust than others against data manipulations.

A general technique to convert approximation algorithms to incentive compatible approximation algorithms is via linear programming relaxation of the optimization problem. This idea has been substantiated in the framework proposed by Lavi and Swamy (LS framework)

(Lavi and Swamy, 2011). The approximations given by the LS framework crucially depend on a geometric property of the underlying polytope of the relaxation, the *integrality gap*. The integrality gap is calculated with respect to the worst-case geometry that a polytope may take. However, in practice (also as the experiments show) the integrality gaps of the polytopes are far less than the worse-case gaps. An analogy is the simplex method with an exponential worst-case runtime but with a polynomial runtime in practice. An interesting research direction is to perform a *smoothed analysis* (Spielman and Teng, 2009) of the integrality gap rather than just a worst-case analysis.

In theoretical computer science, the class of problems that admit incentive-compatible and constant-ratio approximation algorithms is denoted by APX^{IC}. Currently, we know $\text{APX} \neq \text{APX}^{\text{IC}}$, because the submodular welfare maximization problem admits a $(1 - 1/e)$-approximation but no constant-ratio and incentive-compatible algorithm (Dughmi and Vondrák, 2015; Dobzinski and Vondrák, 2013). For many problems in APX, we do not know if they also belong to APX^{IC}. Hence, I believe there are still several open problems in the field of algorithmic mechanism design, and a continuation of research should take place. As an example, consider the problem of combinatorial auctions with budgeted valuations, in which bidders have an overall budget capping their willingness to pay. For this problem, a 3/4-approximation algorithm is already known, however, no incentive-compatible algorithm with a constant-ratio approximation is as of yet known for the problem.

Bayesian incentive compatibility (BIC) according to which truthful bidding is a Bayes-Nash equilibrium, is less stringent than DSIC because we already know how to reduce from BIC mechanism design to algorithm design (Hartline et al., 2015). This possibility primarily stems from the option of reimplementing the distribution of each bidder's valuation in a way that is best for the bidder while keeping changes transparent to the others. An interesting research direction would be to analyze settings where some pieces of data are private while the other pieces of data are drawn from a priori known distribution. These settings inherit properties from both complete and incomplete information games.

A separate research topic is the subject of real-time bidding. In display advertising the most significant concept in recent years is real-time bidding (RTB), or programmatic buying, where advertisers have the ability of making decisions for every impression. Special characteristics of RTB such as being real-time (the responses should take less than 100 milliseconds) and the involvement of multiple layers of players (like demand-side platforms, supply-side platforms, and ad exchanges), provide the possibility of research and development for a wide range of researchers from online algorithm and mechanism designers to web developers (Muthukrishnan, 2009).

In algorithmic mechanism design without money (payment-free environments), the Gibbard-Satterthwaite (G-S) theorem rules out the existence of any truthful, non-dictatorial and unanimous social choice function whose range comprises three or more alternatives (Satterthwaite, 1975; Gibbard, 1973). A dictatorial social choice function is not intended as it does not really aggregate the preferences of agents, but simply the responsibility of choosing the outcome is left solely to a single individual. Recently, the technique of welfare approximation has been shown to be useful in finding non-dictatorial welfare maximizing algorithms for restricted domains where it is possible to avoid the G-S impossibility theorem. See Procaccia and Tennenholtz (2013) and references therein. We applied this technique to a strategic variants of the generalized assignment problem in Chapter 4. A ground-clearing result characterizing the classes of problems which admit non-dictatorial algorithms is tempting; a characterization of a technique – similar to Maximal-In-Distributional-Range allocation rules in mechanism design with quasi-linear utilities – will be of interest.

BIBLIOGRAPHY

Archer, A., Papadimitriou, C., Talwar, K., Tardos, É., 2004. An approximate truthful mechanism for combinatorial auctions with single parameter agents. Internet Mathematics 1 (2), 129–150.

Arora, S., Hazan, E., Kale, S., 2012. The multiplicative weights update method: a meta-algorithm and applications. Theory of Computing 8 (1), 121–164.

Azar, Y., Hoefer, M., Maor, I., Reiffenhäuser, R., Vöcking, B., 2015. Truthful mechanism design via correlated tree rounding. In: Proceedings of the Sixteenth ACM Conference on Economics and Computation. ACM, pp. 415–432.

Barbera, S., Peleg, B., 1990. Strategy-proof voting schemes with continuous preferences. Social choice and welfare 7 (1), 31–38.

Bazaraa, M. S., Jarvis, J. J., Sherali, H. D., 2011. Linear programming and network flows. John Wiley & Sons.

Bland, R. G., Goldfarb, D., Todd, M. J., 1981. The ellipsoid method: a survey. Operations research 29 (6), 1039–1091.

Blumrosen, L., Nisan, N., 2005. On the computational power of iterative auctions. In: Proceedings of the 6th ACM conference on Electronic commerce. ACM, pp. 29–43.

Blumrosen, L., Nisan, N., 2007. Combinatorial auctions (a survey). In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (Eds.), Algorithmic game theory, chapter 11, 267–300.

Boyd, S., Vandenberghe, L., 2009. Convex optimization. Cambridge university press.

Bressoud, T. C., Rastogi, R., Smith, M. A., 2003. Optimal configuration for BGP route selection. In: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies. Vol. 2. IEEE, pp. 916–926.

Budish, E., Che, Y.-K., Kojima, F., Milgrom, P., 2013. Designing random allocation mechanisms: Theory and applications. The American Economic Review 103 (2), 585–623.

Calinescu, G., Chekuri, C., Pál, M., Vondrák, J., 2011. Maximizing a monotone submodular function subject to a matroid constraint. SIAM Journal on Computing 40 (6), 1740–1766.

Carr, R., Vempala, S., 2000. Randomized metarounding. In: Proceedings of the thirty-second annual ACM symposium on Theory of computing. ACM, pp. 58–62.

Cavallo, R., Krishnamurthy, P. a., 2015. On the truthfulness of GSP. Eleventh Workshop on Sponsored Search Auctions.

Chakrabarty, D., Goel, G., 2010. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and GAP. SIAM Journal on Computing 39 (6), 2189–2211.

Chekuri, C., Khanna, S., 2005. A polynomial time approximation scheme for the multiple knapsack problem. SIAM Journal on Computing 35 (3), 713–728.

Chen, N., Gravin, N., Lu, P., 2013. Truthful generalized assignments via stable matching. Mathematics of Operations Research 39 (3), 722–736.

Chvátal, V., 1983. Linear programming. WH Freeman and Company, New York.

Clarke, E., 1971. Multipart pricing of public goods. Public Choice XI, 17–33.

Dantzig, G. B., 1992. An epsilon precise feasible solution to a linear program with a convexity constraint in 1 over epsilon squared iterations independent of problem size.

Desrosiers, J., Lübbecke, M. E., 2005. A primer in column generation. Springer.

Dobson, G., Nambimadom, R. S., 2001. The batch loading and scheduling problem. Operations research 49 (1), 52–65.

Dobzinski, S., Dughmi, S., 2009. On the power of randomization in algorithmic mechanism design. In: Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on. IEEE, pp. 505–514.

Dobzinski, S., Fu, H., Kleinberg, R., 2010. Truthfulness via proxies. CoRR abs/1011.3232.

Dobzinski, S., Vondrák, J., 2012. The computational complexity of truthfulness in combinatorial auctions. In: Proceedings of the 13th ACM Conference on Electronic Commerce. ACM, pp. 405–422.

Dobzinski, S., Vondrák, J., 2013. Communication complexity of combinatorial auctions with submodular valuations. In: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, pp. 1205–1215.

Dughmi, S., Ghosh, A., 2010. Truthful assignment without money. In: Proceedings of the 11th ACM conference on Electronic commerce. ACM, pp. 325–334.

Dughmi, S., Roughgarden, T., Vondrák, J., Yan, Q., 2011a. An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries. arXiv preprint arXiv:1109.1053.

Dughmi, S., Roughgarden, T., Yan, Q., 2011b. From convex optimization to randomized mechanisms: toward optimal combinatorial auctions. In: Proceedings of the 43rd annual ACM symposium on Theory of computing. ACM, pp. 149–158.

Dughmi, S., Vondrák, J., 2015. Limitations of randomized mechanisms for combinatorial auctions. Games and Economic Behavior 92, 370–400.

Ehlers, L., Klaus, B., 2003. Coalitional strategy-proof and resource-monotonic solutions for multiple assignment problems. Social Choice and Welfare 21 (2), 265–280.

Elbassioni, K., Mehlhorn, K., Ramezani, F., 2015. Towards more practical linear programming-based techniques for algorithmic mechanism design. In: Algorithmic Game Theory. Springer, pp. 98–109.

Fadaei, S., 2015. Mechanism design via Dantzig-Wolfe decomposition. arXiv preprint arXiv:1508.04250.

Fadaei, S., Bichler, M., 2014. A truthful-in-expectation mechanism for the generalized assignment problem. In: Web and Internet Economics. Springer, pp. 247–248.

Fadaei, S., Bichler, M., 2016. Generalized assignment problem: Truthful mechanism design without money. TUM working paper.

Feige, U., Vondrak, J., 2006. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In: Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on. IEEE, pp. 667–676.

Ferland, J. A., 1998. Generalized assignment-type problems a powerful modeling scheme. In: Practice and Theory of Automated Timetabling II. Springer, pp. 53–77.

Fisher, M. L., Jaikumar, R., 1981. A generalized assignment heuristic for vehicle routing. Networks 11 (2), 109–124.

Fleischer, L., Goemans, M. X., Mirrokni, V. S., Sviridenko, M., 2006. Tight approximation algorithms for maximum general assignment problems. In: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. ACM, pp. 611–620.

Gibbard, A., 1973. Manipulation of voting schemes: a general result. Econometrica 41, 587–601.

Groves, T., 1973. Incentives in teams. Econometrica 41, 617–631.

Hartline, J. D., Kleinberg, R., Malekian, A., 2015. Bayesian incentive compatibility via matchings. Games and Economic Behavior 92, 401–429.

Hatfield, J., 2009. Strategy-proof, efficient, and nonbossy quota allocations. Social Choice and Welfare 33 (3), 505–515.

Kalagnanam, J. R., Davenport, A. J., Lee, H. S., 2001. Computational aspects of clearing continuous call double auctions with assignment constraints and indivisible demand. Electronic Commerce Research 1 (3), 221–238.

Khandekar, R., 2004. Lagrangian relaxation based algorithms for convex programming problems. Ph.D. thesis, Indian Institute of Technology Delhi.

Klastorin, T., 1979. Note–On the maximal covering location problem and the generalized assignment problem. Management Science 25 (1), 107–112.

Koutsoupias, E., 2014. Scheduling without payments. Theory of Computing Systems 54 (3), 375–387.

Kraft, D., Fadaei, S., Bichler, M., 2014. Fast convex decomposition for truthful social welfare approximation. In: Web and Internet Economics. Springer, pp. 120–132.

Kuhn, H. W., 1955. The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 (1-2), 83–97.

Lavi, R., Mu'alem, A., Nisan, N., 2003. Towards a characterization of truthful combinatorial auctions. In: Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on. pp. 574 – 583.

Lavi, R., Swamy, C., 2005. Truthful and near-optimal mechanism design via linear programming. In: Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on. IEEE, pp. 595–604.

Lavi, R., Swamy, C., 2011. Truthful and near-optimal mechanism design via linear programming. Journal of the ACM (JACM) 58 (6), 25.

Martello, S., Toth, P., 1992. Generalized assignment problems. In: Algorithms and Computation. Springer, pp. 351–369.

Moulin, H., 1980. On strategy-proofness and single peakedness. Public Choice 35 (4), 437–455.

Muthukrishnan, S., 2009. Ad exchanges: Research issues. In: Internet and network economics. Springer, pp. 1–12.

Nguyen, T., Peivandi, A., Vohra, R., 2015. Assignment problems with complementarities. Tech. rep., Mimeo., February.

Nisan, N., Ronen, A., 2001. Algorithmic mechanism design. Games and Economic Behavior 35, 166–196.

Nisan, N., Ronen, A., 2007. Computationally feasible VCG mechanisms. J. Artif. Intell. Res.(JAIR) 29, 19–47.

Öncan, T., 2007. A survey of the generalized assignment problem and its applications. INFOR: Information Systems and Operational Research 45 (3), 123–141.

Papadimitriou, C., Schapira, M., Singer, Y., 2008. On the hardness of being truthful. In: Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on. IEEE, pp. 250–259.

Pápai, S., 2001. Strategyproof and nonbossy multiple assignments. Journal of Public Economic Theory 3 (3), 257–271.

Procaccia, A. D., Tennenholtz, M., 2013. Approximate mechanism design without money. ACM Transactions on Economics and Computation 1 (4), 18.

Ross, G. T., Soland, R. M., 1977. Modeling facility location problems as generalized assignment problems. Management Science 24 (3), 345–357.

Roth, A. E., Sotomayor, M. A. O., 1992. Two-sided matching: A study in game-theoretic modeling and analysis. No. 18. Cambridge University Press.

Ruland, K. S., 1999. A model for aeromedical routing and scheduling. International Transactions in Operational Research 6 (1), 57–73.

Satterthwaite, M. A., 1975. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of economic theory 10 (2), 187–217.

Shmoys, D. B., Tardos, É., 1993. An approximation algorithm for the generalized assignment problem. Mathematical Programming 62 (1-3), 461–474.

Shoham, Y., Leyton-Brown, K., 2008. Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press.

Spielman, D. A., Teng, S.-H., 2009. Smoothed analysis: an attempt to explain the behavior of algorithms in practice. Communications of the ACM 52 (10), 76–84.

Tebboth, J. R., 2001. A computational study of Dantzig-Wolfe decomposition. Ph.D. thesis, University of Buckingham.

Vickrey, W., 1961. Counterspeculation, auctions, and competitive sealed tenders. The Journal of finance 16 (1), 8–37.

Wilkens, C. A., Cavallo, R., Niazadeh, R., 2016. Mechanism design for value maximizers. Cornell Working paper.

Zimokha, V., Rubinstein, M., 1988. R & D planning and the generalized assignment problem. Automation and Remote Control 49 (49), 484–492.