A Truthful-in-expectation Mechanism for the Generalized Assignment Problem

Salman Fadaei and Martin Bichler

Department of Informatics, TU München, Munich, Germany
salman.fadaei@in.tum.de, bichler@in.tum.de

Abstract. We propose a truthful-in-expectation, $(1 - \frac{1}{e})$-approximation mechanism for the generalized assignment auction. In such an auction, each bidder has a knapsack valuation function and bidders' values for items are private. We present a novel convex optimization program for the problem which makes a maximal-in-distributional-range (MIDR) allocation rule. The presented convex program contains at least $(1 - \frac{1}{e})$ ratio of the optimal social welfare. We show how to implement the convex program in polynomial time using a fractional greedy algorithm which approximates the optimal solution within an arbitrarily small error. This leads to an approximately MIDR allocation rule which in turn transforms to an approximately truthful-in-expectation mechanism. From an algorithmic point of view, our contribution has importance, as well; it outperforms the existing optimization algorithms for the GAP in terms of runtime while the approximation ratio is comparable to the best given approximation.

Keywords: Generalized assignment problem, truthful-in-expectation, mechanism design, convex optimization

1 Introduction

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

The well-known *Vickrey-Clarke-Groves* (VCG) technique provides truthfulness as well as social welfare maximization. The VCG technique however is applicable when the optimal social welfare can be computed efficiently. Yet, in many cases including our problem optimizing social welfare is computationally intractable and this makes the VCG technique inapplicable. Usually, when faced with computational intractability, computer scientists turn to approximations or heuristics. Unfortunately, the VCG technique cannot be applied to approximate solutions [1].

The best for a mechanism designer is to devise a computationally efficient and truthful mechanism with an approximation factor that (very closely) matches

the best one known for the problem in which the underlying data is publicly known. In many cases it has been shown that it is impossible to achieve the same approximation factor in incentive-compatible mechanisms [2, 3, 4, 5, 6].

From an algorithmic point of view, the generalized assignment problem (henceforth GAP) has been studied extensively in the literature. Chekuri and Khanna [7] made it explicit that the algorithm of Shmoys and Tardos [8] can be adapted to give a 2-approximation. Later Fleischer et al. [9] improved the factor to $1 - \frac{1}{e}$. Using a reduction to submodular maximization subject to a matroid constraint, Calinescu et al. [10] achieved a ratio of $1 - \frac{1}{e} - o(1)$ without using the ellipsoid method which was pivotal in [9]. An algorithm due to Feige and Vondrák [11] yields an approximation factor of $1 - \frac{1}{e} + \rho$, $\rho \leq 10^{-5}$ which is the best given approximation ratio for the GAP. Chakrabarty and Goel [12] gave the best known hardness result, showing that it is NP-hard to approximate GAP to any factor better than $\frac{10}{11}$.

We observe that all aforementioned algorithms consist of two algorithms, a *relaxation algorithm* and a *rounding algorithm*. This type of algorithm usually cannot constitute a truthful mechanism, since the rounding component is not predictable. For instance, suppose x is a fractional feasible solution with social welfare more than y, i.e. $\sum_i v_i(x) > \sum_i v_i(y)$. However, the relation might be different when the rounding procedure r is applied to the fractional solutions: $\sum_i v_i(X) < \sum_i v_i(Y)$, where $X \sim r(x)$ and $Y \sim r(y)$. This violates the MIDR property.

In order to devise truthful mechanisms, Dughmi et al. [13] propose an approach which optimizes directly on the outcome of the rounding algorithm, rather than on the outcome of the relaxation algorithm. Since the rounding procedure is embedded into the objective function, this approach is not always computationally tractable. Yet, assuming that the optimization problem can be solved efficiently, this approach always leads to an MIDR algorithm. MIDR or maximal-in-distributional-range 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 [14].

Lavi and Swamy [15] proposed a general method for deriving MIDR mechanisms from linear programming relaxations. They solve the relaxed problem in the first step and then they use a very special rounding method (convex decomposition) to obtain the randomized integral allocation. Although they are also using 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 to the rounding algorithm, component-wise, and this interestingly guarantee truthfulness-in-expectation. In contrast to the approach of [13], it is straightforward to design a truthful mechanism using this framework, however it is not obvious how to apply the framework to the settings where bidders have private structured valuations such as submodular functions.

1.1 Our Results and Techniques

Despite all the impossibility results in the field of algorithmic mechanism design, in this paper we present a truthful-in-expectation randomized mechanism for the GAP.

In order to achieve a MIDR, we directly optimize over the outcome of the rounding procedure, rather than on the outcome of the relaxation algorithm. To this end, we formulate the GAP as a convex optimization problem where the objective function equals the expected value of the rounding procedure. This is similar to the technique used in [13] for finding a truthful-in-expectation mechanism for players whose valuations are of a special type of submodular functions. We notice that our technique allows to guarantee *non-negativity of payments* and *individual rationality ex post*, while in [13], these important properties are provided only *ex ante*.

Although, our formulation for the GAP is convex, unfortunately we are not able to solve the convex optimization program exactly, yet we are able to approximate it within an arbitrarily small error, in the sense of an FPTAS. This in fact leads to an approximate MIDR as mentioned in the following.

Theorem 1. *There is a* $(1-\epsilon)$*-MIDR allocation rule that achieves a* $(1-\frac{1}{e}-\epsilon)$*-approximation to the social welfare in the generalized assignment problem, for any* $\epsilon > 0$*.*

It has been shown in [16] how to transform an approximately MIDR allocation rule to an approximately truthful-in-expectation mechanism. Taking into account this black box transformation, we immediately conclude that

Theorem 2. *There is a* $(1-\epsilon)$*-truthful-in-expectation mechanism that achieves a* $(1-\frac{1}{e}-\epsilon)$*-approximation to the social welfare in the generalized assignment problem, for any* $\epsilon > 0$*.*

We remark that our result has algorithmic importance, as well. It has advantages over the existing optimization algorithms in terms of runtime. Our algorithm does not employ the ellipsoid method as in [9]. Moreover, the algorithm improves over that of [10] since in the algorithm of [10], in each iteration, a random sampling is required to compute the gradient of the function at the current point which increases runtime. Actually in [10], the gradient of the function is calculated by taking the average of $(mn)^5$ independent samples, where m and n are the number of items and bidders, respectively. However, we use a novel objective function which is specified exactly, rather than by random sampling, therefore it is possible to calculate the gradient of the objective function explicitly, thus greatly improving the runtime of the algorithm.

2 Preliminaries

In the generalized assignment problem (GAP), there are n bidders and m items. Let I and J denote the set of bidders and items, respectively. Let v_{ij} denote the

value of bidder i for item j. Each bidder 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 bidder i , i.e. $\forall S \in \mathcal{F}_i : \sum_{j \in S} w_{ij} \leq C_i$. Every item can be assigned to only one bidder.

We assume bidders' valuations for items are private while weights and capacities are publicly known.

An allocation $(S_1, \ldots, S_n)$, where $S_i \subseteq J$ denotes the subset assigned to bidder i, is feasible if $\forall i \in I : S_i \in \mathcal{F}_i$ and $\{S_i\}_{i \in I}$ are mutually disjoint. The knapsack valuation is defined as $g_i : 2^J \to \mathbb{R}_{\geq 0}$ such that $g_i(S) = \max_{T \subseteq S, T \in \mathcal{F}_i} \sum_{j \in T} v_{ij}$. Notice, $\forall S \in \mathcal{F}_i : g_i(S) = \sum_{j \in S} v_{ij}$. 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 $\sum_{i \in I} g_i(S_i)$.

Due to the revelation principle, we limit ourselves 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 1 (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 that is 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 $D \in \mathcal{R}$.*

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

Definition 2 ($(1-\epsilon)$-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 that is 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 $D \in \mathcal{R}$.*

Definition 3 ($(1-\epsilon)$ truthful-in-expectation). *A mechanism is $(1-\epsilon)$-approximately truthful-in-expectation for the GAP problem 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,*

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

The expectation in (3) is taken over the coin flips of the mechanism.

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

3 MIDR Allocation Rule for the GAP

We optimize directly over the expected value of the allocation produced by the 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 player i.

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

In effect, in $\mathcal{R}$ one randomized feasible set is assigned to each bidder i. The sets assigned to different players may overlap, however in the rounding step, as we explain next, each item is assigned only once. We wish to maximize the expected value of the rounded allocation over range $\mathcal{R}$. This leads to an MIDR allocation, since we maximize over a range which is independent of the players' private information. Let call the rounding algorithm as r_{greedy}. Algorithm 1 presents the desired MIDR algorithm.

Algorithm 1: 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 $(S_1, \ldots, S_n) \sim r_{\text{greedy}}(x^*)$.

Proposition 1. *Algorithm 1 is an MIDR allocation rule.*

As we show in the following, interestingly, this optimization problem is tractable. We attempt to explain step by step how to implement Algorithm 1 and how good the outcome of the algorithm is. We first start explaining the rounding procedure.

3.1 Greedy Rounding

We choose a rounding algorithm which preserves a good ratio of the fractional solution while it 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 called greedy rounding has two steps. In the first step, given a point $x \in \mathcal{R}$ it 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 bidder i with probability $x'_{i,S}$ while resolving conflicts as explained in the algorithm.

To do the first step, we propose Algorithm 2. Algorithm 2 takes a point $x \in \mathcal{R}$ and the 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 2: An oblivious method for finding a dominated point in $\mathcal{R}$.

Data: $x \in \mathcal{R}$, $y' \in [0,1]^{I \times J}$ 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 *bidder* i **do**
  foreach *item* j **do**
    1. **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'.

The following lemma confirms that Algorithm 2 returns the desired outcome.

Lemma 1. *Suppose $x \in \mathcal{R}$ with polynomially-many $x_{i,S} > 0$, $y' \in [0,1]^{I \times J}$, and $y' \preceq \phi(x)$. If we call Algorithm 2 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 it for one bidder and one item and since the number of items and bidders is polynomial, we obtain the desired conclusion.

Fix bidder i and item j. We consider one iteration in which $x'_{i,S}$ with $j \in S$ is chosen. Two cases can happen. 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: $x'_{i,S\setminus\{j\}}$. Moreover, this case can happen as many times as the number of $x_{i,S} > 0$, where $j \in S$ which are only polynomially-many by assumption.

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

Thus, in total for bidder 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. □

For the first step of the rounding algorithm, thus we call Algorithm 2 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 $y' \preceq y$.

Now, we are ready to present the greedy rounding algorithm, r_{greedy}.

Algorithm 3: 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 (2) with x and y' as the inputs and let x' be the result.
2. Independently for each bidder i, assign set S to i with probability $x'_{i,S}$. If some item j is assigned to more than a bidder, assign it to the bidder among these bidders with the maximum value v_{ij}. Let S_i be the set assigned to bidder i.
return $(S_1, \ldots, S_n)$.

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

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

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

Proof. Assume $x \in \mathcal{R}$. Let x' be the outcome of Step 1 of Algorithm 3. 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 $\sigma_j(i) = i$. That means, bidders 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 bidder 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 bidder 1 does not contain the item and the set assigned to bidder 2 contains it and therefore item j is assigned to bidder 2. This case leads to a returned value of $(1 - y'_{1j})y'_{2j}v_{2j}$.

Continuing similarly for other bidders, 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 to $\sum_{i=1}^{n}(v_{ij} - v_{i+1,j})(1 - \prod_{k=1}^{i}(1 - y'_{kj}))$. The equality of the two terms can be observed by simply extending the latter. Taking into account that $y'_{ij} = 1 - e^{-y_{ij}}$, and 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

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

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

3.2 The Approximation Ratio

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

The configuration LP of the 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 $F(y)$, first we introduce a new variable into the program and then we 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:

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

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

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

We observe that solving GAP-CLP is equivalent to finding $\max_{y\in\mathcal{P}'} \sum_{i\in I, j\in J} v_{ij}y_{ij}$.

Now, we are 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 returns 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, and summing them up, we obtain the desired conclusion. □

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

$$\text{maximize} \sum_{j=1}^{m}\sum_{i=1}^{n}\big(v_{\sigma_j(i),j}-v_{\sigma_j(i+1),j}\big)\big(1-\exp(-\sum_{k=1}^{i} y_{\sigma_j(k),j})\big)$$

$$\forall i\in I, \forall j\in J: \sum_{S\in\mathcal{F}_i: j\in S} x_{i,S}=y_{ij},$$

$$\forall i\in I: \sum_{S\in\mathcal{F}_i} x_{i,S}\leq 1,$$

$$\forall i\in I, \forall S\in\mathcal{F}_i: x_{i,S}\geq 0.$$

First, we show that GAP-CONVEX 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 which is shown by the following theorem.

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

Proof. The function is concave, since it is weighted sums of functions which are compositions of the concave function $1-e^{-x}$ with affine function $x\rightarrow\sum_{k=1}^{i} y_{\sigma_j(k),j}$. □

It is also possible to see the concavity of the function by observing that the second partial derivatives of the function are not positive. We calculate second partial derivatives in Lemma 7.

We also need to show that $\mathcal{P}$ is a *packing polytope* which is to prove for every $y,y'\in[0,1]^{I\times J}$ if $y\in\mathcal{P}$ and $y'\prec y$, then $y'\in\mathcal{P}$.

Lemma 5. *Polytope $\mathcal{P}$ is a packing polytope.*

Proof. Suppose y is in $\mathcal{P}$. This implies that there exists $x\in\mathcal{R}$ such that $y=\phi(x)$. Assume $y'\in[0,1]^{I\times J}$ and $y'\prec y$. To show that $y'\in\mathcal{P}$, we must prove that there exists $x'\in\mathcal{R}$ such that $y'=\phi(x')$.

We start from point y^1 which is equal to y in all components except for one component and in that component, y^1 is equal to the corresponding component in y'. In other words, assuming w.l.o.g. that $y'_{k1}<y_{k1}$, we look at point y^1 with $y^1_{ij}=y_{ij}\ \forall(i,j)\neq(k,1)$ and $y^1_{k1}=y'_{k1}$. We show that $y^1\in\mathcal{P}$. By induction, subsequently, we can replace the role of y and y^1 and repeat this argument by looking at another point which is equal to y' in one more component until we reach to point y' and show that $y'\in\mathcal{P}$.

To show that $y^1\in\mathcal{P}$, we must prove that there exists $x^1\in\mathcal{R}$ such that $y^1=\phi(x^1)$. Let $\mathcal{H}$ be a set of subsets of items with the following properties. $\mathcal{H}\subseteq\mathcal{F}_k$, $\forall S\in\mathcal{H}: 1\in S$, $\sum_{S\in\mathcal{H}} x_{k,S}\geq y^1_{k1}$ and there exists $T\in\mathcal{H}$ such that $\sum_{S\in\mathcal{H}\setminus\{T\}} x_{k,S}<y^1_{k1}$. It is easy to observe that such set $\mathcal{H}$ exists.

We can now choose point x^1 which is equal to x in all components but with the following differences: $x^1_{k,T}=x_{k,T}-d$ and $x^1_{k,T\setminus\{1\}}=x_{k,T\setminus\{1\}}+d$ for $T\setminus\{1\}\neq\emptyset$,

where $d = \sum_{S\in\mathcal{H}} x_{k,S} - y^1_{k1}$. Moreover, for every $S \in \mathcal{F}_k \setminus \mathcal{H}$, where $1 \in S$, we have $x^1_{k,S} = 0$ and $x^1_{k,S\setminus\{1\}} = x_{k,S\setminus\{1\}} + x_{k,S}$ for $S \setminus \{1\} \neq \emptyset$. We show that x^1 is the required point: $x^1 \in \mathcal{R}$ and $y^1 = \phi(x^1)$.

$x^1 \in \mathcal{R}$ because $\forall S \in \mathcal{F}_k$ we have $S \setminus \{1\} \in \mathcal{F}_k$ and therefore if $x^1_{k,S} > 0$ then $S \in \mathcal{F}_k$. Moreover, it is easy to observe that $\sum_{S\in\mathcal{F}_k} x^1_{k,S} = \sum_{S\in\mathcal{F}_k} x_{k,S}$. Furthermore, every component of x^1 is non-negative. Actually, we only need to show that $x^1_{k,T}$ is non-negative. By definition of $\mathcal{H}$: $\sum_{S\in\mathcal{H}\setminus\{T\}} x_{k,S} - y^1_{k1} < 0$, adding $x_{k,T}$ to both sides we obtain $d < x_{k,T}$, equivalently $0 < x^1_{k,T}$.

We also have $y^1 = \phi(x^1)$ since $\sum_{S\in\mathcal{F}_k;1\in S} x^1_{k,S} = (\sum_{S\in\mathcal{H}} x_{k,S}) - d = y^1_{k1}$ and all other components in y^1 remain equal to y. This completes the proof. □

In order to solve the convex optimization problem, we present a fractional greedy algorithm. Our algorithm gets arbitrarily close to the optimal solution. Unfortunately, at this point we do not know how to solve the convex optimization problem exactly. This is mostly because of the difficulty that arises from the exponential number of variables in the convex program. Thus, we are able to implement a $(1-\epsilon)$-MIDR allocation rule, for any $\epsilon > 0$.

Our fractional greedy algorithm looks like the local search algorithm presented in [16]. However, since we are maximizing a different objective function over a different polytope, we explain the algorithm completely. In every iteration of the algorithm, we need to find $y^* \in \mathcal{P}$ which maximizes $y \cdot \nabla F(y)$ [1] over all $y \in \mathcal{P}$. Proposition 2 tells us that 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 bidder i which maximizes $\sum_{j\in S^*_i} v_{ij}$.

Proposition 2. $\max_{y\in\mathcal{P}} \sum_{i\in I, j\in J} v_{ij}y_{ij} = \sum_{i\in I} \max\{\sum_{j\in S} v_{ij} : S \in \mathcal{F}_i\}$.

Proof.

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

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 bidder i.

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

[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.

Algorithm 4: Fractional greedy algorithm

Data: $v = (v_{ij})_{i\in I, j\in J}$, $\epsilon > 0$.
Result: $x \in \mathcal{R}$ such that $F(\phi(x)) \geq (1-\epsilon)\max\{F(y)|y \in \mathcal{P}\}$.
0. Initialize $x := \mathbf{0}$; $y := \mathbf{0}$. [a] Let $\delta = \frac{\epsilon}{8m^2n^2}$.
1. Let $u := \nabla F(y)$; $z := \mathbf{0}$. [b]
2. **foreach** *bidder* i **do**
   Find a set $S_i^* \in \mathcal{F}_i$ such that $\sum_{j\in S_i^*} u_{ij} > (1-\epsilon)\max\{\sum_{j\in S} u_{ij} : S \in \mathcal{F}_i\}$ using the FPTAS for the knapsack problem; set $z_{ij} := 1$ if $j \in S_i^*$, otherwise $z_{ij} := 0$. Keep S_i^* for a possible update in the next step.
3. **if** $z \cdot \nabla F(y) > \frac{1}{4}\epsilon M$ **then**
   update $y := y + \delta z$ and for all $i \in I$ update $x_{i,S_i^*} := x_{i,S_i^*} + \delta$; go back to Step 1.
return x.

[a] $x \in [0,1]^{I\times 2^J}$, $y \in [0,1]^{I\times J}$.
[b] $u, z \in [0,1]^{I\times J}$.

Lemma 6. *Algorithm 4 returns $x \in \mathcal{R}$ such that $F(\phi(x)) \geq (1-\epsilon)\max\{F(y)|y \in \mathcal{P}\}$.*

Proof. Assume $x \in \mathcal{R}$ is the outcome of the algorithm. Let $y = \phi(x)$. Let z be the calculated vector in the last iteration in Step 2 (i.e., $z \cdot \nabla F(y) \leq \frac{1}{4}\epsilon M$). We have $z \cdot \nabla F(y) \geq (1-\epsilon)\max_{w\in\mathcal{P}} w \cdot \nabla F(y)$ according to Proposition 2.

Let $y^* = \arg\max_{y\in\mathcal{P}} F(y)$. We have, $y^* \in \mathcal{P}$, $((y^* - y) \vee \mathbf{0}) \preceq y^*$ [2] and $\mathcal{P}$ is a packing polytope (Lemma 5), thus, $(y^* - y) \vee \mathbf{0} \in \mathcal{P}$. Therefore, $z \cdot \nabla F(y) \geq (1-\epsilon)((y^* - y) \vee \mathbf{0}) \cdot \nabla F(y)$.

In the last iteration of the algorithm we have $z \cdot \nabla F(y) \leq \frac{1}{4}\epsilon M$, thus we obtain

$$\begin{aligned} F(y^*) - F(y) &\leq (y^* - y) \cdot \nabla F(y) \\ &\leq ((y^* - y) \vee \mathbf{0}) \cdot \nabla F(y) \\ &\leq (1-\epsilon)^{-1} z \cdot \nabla F(y) \\ &\leq \tfrac{1}{4}\epsilon M (1-\epsilon)^{-1} \\ &\leq \epsilon \cdot F(y^*). \end{aligned}$$

The first inequality is because of the concavity of F. Therefore, when the algorithm terminates $F(y) \geq (1-\epsilon)F(y^*)$ and this completes the proof. □

Lemma 7. *In each iteration, the value of $F(y)$ increases by at least $\frac{\epsilon^2}{64m^2n^2}M$.*

Proof. The change in gradient has a certain upper bound when y changes by a certain amount:

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

[2] $x \vee y$ denotes the coordinate-wise maximum, $(x \vee y)_i = \max\{x_i, y_i\}$.

$$\left|\frac{\partial^2 F}{\partial {y_{ij}}^2}\right| = \sum_{l=i}^{n}(v_{lj} - v_{l+1,j})\exp(-\sum_{k=1}^{l} y_{kj}) \leq M.$$

For $j \neq j'$:

$$\frac{\partial^2 F}{\partial y_{i'j'}\partial y_{ij}} = 0.$$

Finally, for $i \neq i'$:

$$\left|\frac{\partial^2 F}{\partial y_{i'j}\partial y_{ij}}\right| = \sum_{l=\max(i,i')}^{n}(v_{lj} - v_{l+1,j})\exp(-\sum_{k=1}^{l} y_{kj}) \leq M.$$

As $\frac{\partial F}{\partial y_{ij}}$ has continuous derivatives, for any y' such that $||y' - y||_\infty \leq \delta$, we obtain

$$\left.\frac{\partial F}{\partial y_{ij}}\right|_{y'} \geq \left.\frac{\partial F}{\partial y_{ij}}\right|_{y} - \sum_{i,j}|y'_{ij} - y_{ij}| \max\left|\frac{\partial^2 F}{\partial y_{i'j'}\partial y_{ij}}\right| \geq \left.\frac{\partial F}{\partial y_{ij}}\right|_{y} - \delta mnM. \quad (1)$$

As long as the algorithm continues we have $z \cdot \nabla F(y) > \frac{1}{4}\epsilon M$. Thus,

$$\begin{aligned} F(y + \delta z) &\geq F(y) + \delta z \cdot \nabla F(y + \delta z) \\ &\geq F(y) + \delta z \cdot (\nabla F(y) - \delta mnM\mathbf{1}) \\ &\geq F(y) + \delta z \cdot \nabla F(y) - \delta^2 m^2 n^2 M \\ &\geq F(y) + \delta \cdot \tfrac{1}{4}\epsilon M - \delta^2 m^2 n^2 M \end{aligned}$$

where, $\mathbf{1}$ denotes the vector of all ones in $[0,1]^{I\times J}$. The first inequality is because of the concavity of F. The second inequality holds because of inequality (1), above. The third inequality is because $z \cdot \mathbf{1} \leq mn$.

Now, using $\delta = \frac{\epsilon}{8m^2n^2}$, we obtain

$$F(y + \delta z) \geq F(y) + \frac{\epsilon^2}{64m^2n^2}M.$$

□

Lemma 8. *After at most $\frac{64m^3n^2}{\epsilon^2}$ iterations, Algorithm 4 terminates.*

Proof. Since M denotes $\max\{v_{ij} : i \in I; j \in J\}$, mM is an upper bound for OPT. Moreover, based on Lemma 7, in each iteration the growth in value is at least $\frac{\epsilon^2}{64m^2n^2}M$, the algorithm in $\frac{64m^3n^2}{\epsilon^2}$ iterations, reaches the value of mM, which is an upper bound on the best solution. This concludes the proof. □

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

Remark. In order to use this algorithm as an optimization algorithm, one can employ a simpler rounding algorithm. The simpler rounding requires only Step 2

of Algorithm 3. For an optimization purpose, there is no need to also execute Step 1 of the greedy rounding algorithm. Thus, after finding a fractional solution x by invoking Algorithm 4, we assign set S to each bidder i with probability $x_{i,S}$ and resolve conflicts similar to the technique in the greedy rounding algorithm. This improves the runtime for the optimization purpose.

4 Computing Payments

Supplementing the MIDR allocation rule of Section 3 with VCG payments yields a truthful-in-expectation mechanism. We compute payments in order to also enforce non-negative payments and individual rationality, *ex post*.

To compute the VCG fractional payment p_i^F 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 solving GAP-CONVEX by also adding constraint $x_{i,S} = 0$ for all $S \in J$. 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 4. We notice that function $F(y)$ is explicitly known to us and we can set v_{ij} to 0 in it. Finally, $p_i^F = h_i(v_{-i}) - F_{-i}(y^*)$.

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 $w_i^F = F(y^*) - F_{-i}(y^*)$. 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^R, satisfying individual rationality and non-negativity of payments as follows.

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

Bibliography

[1] Nisan, N., Ronen, A.: Computationally feasible vcg mechanisms. In: Electronic Commerce: Proceedings of the 2 nd ACM conference on Electronic commerce. Volume 17. (2000) 242–252

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

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

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

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

[6] Dughmi, S., Vondrák, J.: Limitations of randomized mechanisms for combinatorial auctions. Games and Economic Behavior (2014)

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

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

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

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

[11] Feige, U., Vondrak, J.: 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 (2006) 667–676

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

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

[14] Dobzinski, S., Dughmi, S.: On the power of randomization in algorithmic mechanism design. In: Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on, IEEE (2009) 505–514
[15] Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. Journal of the ACM (JACM) **58**(6) (2011) 25
[16] Dughmi, S., Roughgarden, T., Vondrák, J., Yan, Q.: An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries. arXiv preprint arXiv:1109.1053 (2011)