**Noname manuscript No.**
(will be inserted by the editor)

# Fast Convex Decomposition for Algorithmic Mechanism Design

**Martin Bichler · Salman Fadaei · Dennis Kraft**

Received: date / Accepted: date

**Abstract** Algorithms for the convex decomposition of fractional linear programming solutions are at the core of a class of market protocols proposed by Lavi and Swamy (2011). Due to their generality, these protocols can approximately maximize social welfare in many hard real-world allocation problems while encouraging truthful participation at the same time. Until recently, the only polynomial-time decomposition technique relied heavily on the notoriously inefficient ellipsoid method, limiting the practical applications of Lavi and Swamy's work significantly. To address this issue, we present a much simpler and faster decomposition technique based on a simple geometric idea. After a formal comparison with other decomposition techniques, we conduct an extensive experimental evaluation to show its advantages in practice.

**Keywords** algorithmic game theory · approximation algorithms · convex decomposition · linear programming · mechanism design

## 1 Introduction

In many markets resources are allocated to agents whose valuation of the market's outcome is private information. If the agents are self-interested, they might be misrepresented their valuation for personal benefit. To avoid strategic manipulation and determine a desirable allocation, *algorithmic mechanism design* studies market protocols that encourage truthful participation. See (Nisan et al 2007) for an introduction. The celebrated Vickrey (1961), Clarke (1971) and Groves (1973) (VCG) mechanism achieves this through a pricing scheme ensuring that all agents maximize their utility by revealing their true valuation. For a VCG mechanism to be truthful, the ability to compute outcomes that maximize social welfare with

Dennis Kraft
Department of Informatics
Technical University of Munich
Boltzmannstr. 3, 85748 Garching
Tel.: +49-89-289-17534
E-mail: kraftd@in.tum.de

respect to the reported valuations is crucial. Unfortunately, this so called *winner determination problem* (WDP) is often NP-hard and cannot be solved optimally for many real-world problem sizes. A prominent example is the broadcast incentive auction which was recently conducted by the US Federal Communications Commission.[1] Computationally hard allocation problems are also wide-spread in logistics and supply chain management (Bichler et al 2006). For our experiments we draw on an allocation problem from retail logistics.

## 1.1 Related Work

To preserve truthfulness for approximate solutions of the WDP, a mechanism with a VCG pricing scheme must be *maximal-in-range* (Nisan and Ronen 2000). This means it must return an optimal outcome with respect to a predefined subset of outcomes. The maximal-in-range principle can be readily generalized to randomized mechanisms. Such mechanisms are called *maximal-in-distribution-range* and draw an outcome randomly according to a distribution optimizing expectation among a predefined subset of distributions. In combination with a VCG pricing scheme, maximal-in-distribution-range mechanisms are *truthful in expectation*, i.e. all agents maximize their expected utility by revealing their true valuation (Dobzinski and Dughmi 2009). Assuming the WDP can be expressed as an *integer linear program* (ILP), Lavi and Swamy (2011) propose a scheme to construct maximal-in-distribution-range mechanisms based on ordinary approximation algorithms for the ILP. Due to its remarkable generality, their approach applies to wide range of real-world resource allocation problems.

A formal description of Lavi and Swamy's mechanism design scheme can be found in Section 3. At this point we present a high level overview: Let $\alpha$ be an upper bound on the *integrality gap* of the ILP, i.e. the largest ratio between a solution of the ILP and a solution of the *linear programming relaxation* (LP-relaxation). Scaling the polytope of the LP relaxation by a factor of $\alpha$ yields a set of fractional solutions completely contained in the convex hull spanned by the solutions of the ILP. Any solution from this set therefore corresponds to the expected outcome of some probability distribution over outcomes. Furthermore, it is easy to optimize over this set using standard linear programming techniques such as the simplex method. To obtain a maximal-in-distribution-range mechanism, all that remains is to compute a distribution over outcomes, i.e. a convex combination of integral solutions, matching the relaxed solution in expectation.

For this task, Lavi and Swamy resort to a decomposition technique by Carr and Vempala (2000), which models the decomposition as a linear program (LP). However, since the LP has exponentially many variables, one for each outcome, it is not solved directly. Instead, Carr and Vempala consider the dual to reduce the number of variables. The dual, which has exponentially many constraints, is solved via the ellipsoid method. The use of the ellipsoid method is crucial as is does not need a complete enumeration of the constraints provided that a suitable separation oracle exists (Bland et al 1981). As separation oracle, the approximation algorithm of the WDP can be used. The resulting algorithm yields a polynomially sized convex combination within a polynomial number of calls to the approxima-

[1] `https://www.fcc.gov/about-fcc/fcc-initiatives/incentive-auctions`

tion algorithm. However, considering the notoriously poor practical performance of the ellipsoid method, Carr and Vempala's approach is mostly of theoretical relevance. Since an efficient convex decomposition is crucial for the applicability of Lavi and Swamy's work to real-world problems, two algorithms to replace the ellipsoid method have been developed in parallel. In an earlier conference version of this work, Kraft et al (2014) propose a geometric decomposition technique, while Elbassioni et al's (2016) approach is based on the *multiplicative weights update method* (MWU method).

### 1.2 Contribution

We first introduce a decomposition technique based on a simple geometric principle that avoids the impractical ellipsoid method.[2] Instead of computing an exact decomposition, we settle for arbitrarily precise approximations. Given a precision parameter $\varepsilon > 0$, our approach requires at most $\lceil n^2\varepsilon^{-2}\rceil$ calls to the approximation algorithm, where $n$ denote the number of positive variables in the solution of the LP relaxation. Furthermore, the resulting convex combination consists of at most $\lceil n^2\varepsilon^{-2}\rceil + n + 1$ outcomes. The gain in performance comes at the price of a reduction in social welfare by a factor of $(1 + \varepsilon)$.

Next, we formally compare our algorithm to the decomposition technique proposed by Elbassioni et al (2016). The technique is based on an approximation of Carr and Vempala's LP via Khandekar's (2004) algorithm. The resulting convex combination consist of at most $n\lceil \log(n)\varepsilon^{-2}\rceil$ outcomes and is constructed within at most the same number of calls to the approximation algorithm.

The formal analysis suggests a better performance of Elbassioni et al's decomposition technique in the worst case. To compare performance in practice, we conduct an extensive study with more than 4000 individual experiments, which are based on a real-world resource allocation problem in retail logistics proposed by Karänke et al (2015). To our knowledge, this is the first application of Lavi and Swamy's mechanism design scheme to a realistic resource allocation problem. As the baseline of our experiments we use a slightly modified version of Carr and Vempala's decomposition technique that does not run the ellipsoid method to the end, but terminate as soon as a convex decomposition is possible. This generally requires less calls to the approximation algorithm and allows for a more meaningful comparison to the other decomposition techniques. For low levels of precision, we find that the difference in performance among all three methods is small. However, as the level of precision rises, our algorithm is substantially faster than Elbassioni et al's, which in turn is substantially faster than Carr and Vempala's. In particular, our approach requires the fewest calls to the approximation algorithm. The evaluation of our data suggests that the advantage in practical performance is due to the greedy nature of our algorithm.

## 2 Setting

Many real-world resource allocation problems can be solved naturally by means of *0-1 integer programming.* Consider for instance the following scenario from retail

[2] The key algorithmic idea was presented at WINE Kraft et al (2014).

logistics by Karänke et al (2015) in which $I$ carriers $i \in \{1, 2, \ldots, I\}$ have to transport goods between $K$ warehouses $k \in \{1, 2, \ldots, K\}$. Each carrier owns one truck and each warehouse $k$ can service at most $c_{t,k}$ trucks in time slot $t \in \{1, 2, \ldots, T\}$. To reduce waiting time at the loading docks, carriers can reserve time slots via a binary *reservation matrix* $r \in \{0, 1\}^{T \times K}$. Setting $r_{t,k} = 1$ corresponds to a reservation for warehouse $k$ at time $t$. Otherwise, if $r_{t,k} = 0$, no reservation is made. Let $R = \{0, 1\}^{T \times K}$ be the set of all reservation matrices. Carrier $i$'s valuation for a particular reservation matrix $r$ is denoted by $v_{i,r}$. We assume that all valuations are *normalized* and *monotone*, i.e. $v_{i,r} = 0$ if $r$ contains no reservations and $v_{i,r} \leq v_{i,r'}$ if all reservations of $r$ are included in $r'$.

Suppose reservations are allocated to carriers in a *multi-unit combinatorial auction*. Valuation $v_{i,r}$ corresponds to carrier $i$'s bid for the reservation matrix $r$. The outcome of the auction is captured by binary variables $x_{i,r}$. If carrier $i$ receives her reservations according to $r$, then $x_{i,r} = 1$; otherwise $x_{i,r} = 0$. Assuming the auctioneer aims to maximize social welfare, we can formulate the WDP as the following 0-1 ILP:

$$
\begin{aligned}
\text{maximize:} \quad & \sum_{i=1}^{I} \sum_{r \in R} v_{i,r} x_{i,r} \\
\text{subject to:} \quad & \sum_{i=1}^{I} \sum_{r \in R} r_{t,k} x_{i,r} \leq c_{t,k} \qquad \forall t \in T, k \in K \\
& \sum_{r \in R} x_{i,r} \leq 1 \qquad \forall i \in I \\
& x_{i,r} \in \{0,1\} \qquad \forall i \in I, r \in R
\end{aligned}
\tag{Q}
$$

The first set of constraints ensures that no warehouses exceeds its capacity at any point in time. The second set of constraints captures the XOR relation of bids, i.e. each carrier is granted at most one reservation matrix.

Note that Q corresponds to a *multidimensional multiple choice knapsack problem*. Solving the WDP is therefore NP-hard and the direct implementation of a VCG auction computationally intractable. However, Lavi and Swamy's (2011) mechanism design scheme admits the construction of a randomized auction that approximates social welfare and is truthful in expectation at the same time. We will review their work in Section 3.

### 2.1 Prerequisites

In the following we identify the prerequisites for Lavi and Swamy's mechanism design scheme. For this purpose, we introduce a general framework, which we will use throughout Section 3 and 4. Suppose the WDP of a given resource allocation problem is formulated as a 0-1 ILP $\max\{\sum_{k=1}^{n} v_k x_k \mid x \in X \cap \mathbb{Z}^n\}$ over some $n$-dimensional polytope $X \subseteq [0, 1]^n$. For convenience we define $\mathbb{Z}(X) = X \cap \mathbb{Z}^n$. The requirements the ILP must satisfy are:

1. The components of an outcome $x \in X$ must be *separable* into disjoint sets $N_i$ such that $\sum_{k \in N_i} v_k x_k$ corresponds to agent $i$'s valuation of $x$.

2. Polytope $X$ must satisfy the *packing property*. This means if $x \in [0,1]^n$ is *dominated* by an outcome $x' \in X$, then $x \in X$ must also hold true.
3. The LP relaxation $\max\{\sum_{k=1}^n v_k x_k \mid x \in X\}$ must be efficiently solvable.
4. The integrality gap of $X$ must be bounded by an efficiently *verifiable* value $\alpha \geq 1$. More precisely, a polynomial-time algorithm $\mathcal{A} : \mathbb{R}^n_{\geq 0} \to \mathbb{Z}(X)$ should exists such that $\alpha \sum_{k=1}^n v_k \mathcal{A}(v)_k \geq \max\{\sum_{k=1}^n v_k x_k \mid x \in X\}$ for all $v \in \mathbb{R}^n_{\geq 0}$.

The first requirement, i.e. the separability of outcomes, is necessary for the computation of VCG prices. Clearly, the retail logistics problem satisfies this. As carrier $i$'s valuation of outcome $x \in [0,1]^{I \times R}$ corresponds to $\sum_{r \in R} v_{i,r} x_{i,r}$, it suffices to set $N_i = \{(i,r) \mid r \in R\}$. The remaining three requirements are necessary for the decomposition of LP relaxation's solution. In the remainder of this section, we argue that all three requirements are satisfied by the retail logistics problem.

We begin with the packing property. Let $x, x' \in [0,1]^{I \times R}$ be two, possibly fractional, allocations. We say that $x'$ dominates $x'$ if $x_{i,r} \leq x'_{i,r}$ for all carriers $i$ and reservation matrices $r$. Note that $x$ can be created from $x'$ by partially retaining reservation matrices. Since this neither violates the warehouse capacities nor the XOR relation of bids, Q satisfies the packing property naturally.

Solving the LP relaxation of Q is more intricate. The reason is the exponential number of variables $x_{i,r}$ with respect to the warehouses and time slots. Clearly, it is impractical to solve the LP relaxation directly. Even gathering an exhaustive list of the carriers' valuations is already intractable; a common issue in combinatorial auctions. However, not all valuations must be known explicitly. Under the reasonable assumption that each carrier is only interested in a small number of reservation matrices, bids can be expressed concisely via *bidding languages* (Nisan 2006). Otherwise, *demand queries* can be used to identify relevant valuations (Blumrosen and Nisan 2005). Both approaches yield optimal solutions of the LP relaxation. Furthermore, we may assume these solutions to be *sparse*, i.e. their number of positive components is polynomial in $T$ and $K$.

Finally, we address the integrality gap. Remember that Q represents the WDP of a multi-unit combinatorial auction. As such, it is common knowledge that Q can be a approximated within a ratio of $\mathcal{O}((TK)^{1/(1+c)})$, where $c$ denotes the minimum capacity among all warehouses and time slots. In a stronger result, Briest et al (2005) present a polynomial-time approximation algorithm that verifies an integrality gap of $3e(TK)^{1/(1+c)}$. It is important to note that this integrality gap verfier works for all vectors $v \in \mathbb{R}^{I \times R}_{\geq 0}$ and is not restricted to normalized and monotone valuations.

## 3 The Mechanism Design Scheme

In this section, we review Lavi and Swamy's (2011) mechanisms design scheme. Suppose the WDP is formulated as a 0-1 ILP $\max\{\sum_{k=1}^n v_k x_k \mid x \in \mathbb{Z}(X)\}$ over some $n$-dimensional polytope $X$. Provided that the ILP satisfies the prerequisites of Section 2, the mechanism design scheme yields a randomized market protocol that is truthful in expectation and approximates the optimal social welfare within a ratio of $\alpha$. Their approach consists of three basic steps:

1. Compute a solution $x^* \in X$ of the LP relaxation $\max\{\sum_{k=1}^n v_k x_k \mid x \in X\}$ and determine the corresponding VCG prices $p_i$ for each agent $i$. This yields a truthful mechanism for the fractional problem.

2. Use the $\alpha$-integrality gap verifier $\mathcal{A}$ to decompose $x^*/\alpha$ into a convex combination $\lambda \in [0,1]^{\mathbb{Z}(X)}$ of integral outcomes $x \in \mathbb{Z}(X)$. The resulting convex combination should satisfy $\sum_{x\in\mathbb{Z}(X)} \lambda_x x = x^*/\alpha$.
3. Choose outcome $x \in \mathbb{Z}(X)$ at random with probability $\lambda_x$ and charge each agent $i$ a price of $p_i(\sum_{k\in N_i} v_k x_k)/(\sum_{k\in N_i} v_k x_k^*)$ if $\sum_{k\in N_i} v_k x_k^* \neq 0$. Otherwise do not charge agent $i$ anything. This yields an $\alpha$-approximation mechanism that is truthful in expatiation.

The first step can be implemented via standard linear programming algorithms, such as the simplex method. The implementation of the third step is also straight forward. Our main focus is therefore on the decomposition of $x^*/\alpha$.

For a convenient decomposition we slightly modify Lavi and Swamy's mechanism design scheme in the following way: First, we do not require $\lambda$ to be an exact decomposition of $x^*/\alpha$. Instead, $\lambda$ only needs to weakly dominate $x^*/\alpha$, i.e. $\sum_{x\in\mathbb{Z}(X)} \lambda_x x_k \geq x_k^*/\alpha$ for all $k$. At the end of this section we show how to adapt the third step of the mechanism design scheme to account for such a relaxed $\lambda$. It is also possible to convert $\lambda$ directly into an exact decomposition of $x^*/\alpha$ (Kraft et al 2014). Secondly, we assume that all components of $x^*$ are positive. This assumption is justified as all $\lambda \in [0,1]^{\mathbb{Z}(X)}$ trivially dominate the components $x_k^*/\alpha$ that are 0. Consequently, these components are irrelevant to the decomposition and can be ignored. As mentioned in Section 2, this is particularly important for combinatorial auctions, where the number of variables is exponential, but the fractional solution is sparse.

## 3.1 The Ellipsoid Method

We proceed with the decomposition of $x^*/\alpha$. For this task Lavi and Swamy use a technique proposed by Carr and Vempala (2000), which is based on the ellipsoide method and yields a convex combination $\lambda$ weakly dominating $x^*/\alpha$. Leveraging the packing property of $X$, Lavi and Swamy slightly adapt this technique to obtain an exact decomposition. However, since a dominating $\lambda$ is sufficient for our purposes, we present the original version of Carr and Vempala's technique. Their approach is based on the following LP:

$$\begin{aligned}
\text{minimize:} \quad & \sum_{x\in\mathbb{Z}(X)} \lambda_x \\
\text{subject to:} \quad & \sum_{x\in\mathbb{Z}(X)} x_k \lambda_x \geq x_k^*/\alpha \quad && \text{for all } 1 \leq k \leq n \\
& \sum_{x\in\mathbb{Z}(X)} \lambda_x \geq 1 \\
& \lambda_x \geq 0 && \text{for all } x \in \mathbb{Z}(X)
\end{aligned} \tag{P}$$

The variables $\lambda_x$ represent the coefficients of a linear combination. The first set of constraints ensures that the linear combination weakly dominates $x^*/\alpha$. The second constraint enforces the coefficients to sum up to a value of at least 1. Observe that a convex combination dominating $x^*/\alpha$ exists if and only if P evaluates to 1. To prove that P indeed evaluates to 1, we consider the dual:

$$\begin{aligned}
\text{maximize:} \quad & \Big(\sum_{k=1}^{n} (x^*_k/\alpha)\mu_k\Big) + \nu \\
\text{subject to:} \quad & \Big(\sum_{k=1}^{n} x_k\mu_k\Big) + \nu \le 1 \quad \text{for all } x \in \mathbb{Z}(X) \\
& \mu_k \ge 0 \quad \text{for all } 1 \le k \le n \\
& \nu \ge 0
\end{aligned} \tag{D}$$

The variables $\mu_k$ and $\nu$ correspond to the constraints of P. To gain intuition, it is helpful to think of $\mu$ as a valuation function for the outcomes of the mechanism. According to this notion, the objective function of P captures the social welfare of $x^*/\alpha$ with respect to $\mu$. Similarly, the left hand side of the first set of constraints corresponds to the social welfare of the integral outcomes $x \in \mathbb{Z}(X)$. As the integrality gap of $X$ is bounded by $\alpha$, it is easy to see that D evaluates to 1. Refer to Lemma 1 for a proof. By strong duality P also evaluates to 1, establishes existence of a convex combination dominating $x^*/\alpha$.

**Lemma 1** *D has an optimal solution of value* 1.

*Proof* We first show that D has a solution of value 1. For this purpose, we set $\nu = 1$ and $\mu = 0$. Clearly, this solution does not violate any constraints and has an objective value of 1. It remains to show that no feasible solution has a better objective value. For the sake of contradiction assume the existence of some $\mu'$ and $\nu'$ with an objective value greater than 1, i.e. $(\sum_{k=1}^{n}(x^*_k/\alpha)\mu'_k) + \nu' > 1$. Recall that the integrality gap of $X$ is at most $\alpha$. Consequently, there must exist an $x \in \mathbb{Z}(X)$ for which

$$\Big(\sum_{k=1}^{n} x_k\mu'_k\Big) + \nu' \ge \max\Big\{\sum_{k=1}^{n} \frac{x'_k}{\alpha}\mu'_k \;\Big|\; x' \in X\Big\} + \nu' \ge \Big(\sum_{k=1}^{n} \frac{x^*_k}{\alpha}\mu'_k\Big) + \nu' > 1.$$

However, this contradicts the feasibility of $\mu'$ and $\nu'$ as they violate the constraint of D associated with $x$. □

It remains to show how to compute an optimal solution of P. As each variable $\lambda_x$ corresponds to a point $x \in \mathbb{Z}(X)$, of which there might be exponentially many, it is not practical to solve P directly. Instead, Carr and Vempala consider D first. Since P only has $n$ variables, it can be optimized in polynomial-time via the ellipsoid method provided that a suitable separation oracle exists. Conveniently, $\mathcal{A}$ can be used to construct such a separation oracle, see the proof of Theorem 1 for more details. Let $\tilde{X} \subseteq \mathbb{Z}(X)$ be the set of all integral points in $X$ associated with a constraint the ellipsoid method considered while optimizing D. Clearly, $\tilde{X}$ is of polynomial size. Furthermore, the constraints associated with the points $x \in \tilde{X}$ must be sufficient to verify optimality for the solution computed by the ellipsoid method. By strong duality, the corresponding variables $\lambda_x$ are sufficient to find an optimal solution of P. Let $\tilde{\text{P}}$ be a condensed version of P only containing variables $\lambda_x$ for which $x \in \tilde{X}$. All other variables are 0. As $\tilde{\text{P}}$ is of polynomial size, it can be solve efficiently by standard linear programming algorithms. Algorithm 1 gives a short summary of Carr and Vempala's decomposition technique.

**Algorithm 1:** EllipsoidDecomposition

```
Input: Polytope $X \subseteq [0,1]^n$, fractional solution $x^* \in X$, $\alpha$-integrality gap verifier $\mathcal{A}$
Output: Convex combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$
1 optimize D via the ellipsoid method using $\mathcal{A}$ as separation oracle;
2 let $\tilde{X}$ contain all points $x \in \mathbb{Z}(X)$ considered by the ellipsoid method;
3 construct $\tilde{\mathrm{P}}$ by removing all variables $\lambda_x$ for which $x \notin \tilde{X}$ from P;
4 return an optimal solution of $\tilde{\mathrm{P}}$;
```

**Theorem 1** *Algorithm 1 returns a convex combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$ dominating $x^*/\alpha$. The run time of the algorithm is polynomial in that of $\mathcal{A}$. The number of positive coefficients in $\lambda$ is also polynomial.*

*Proof* Recall that every feasible solution of $\tilde{\mathrm{P}}$ is a linear combination of points in $\tilde{X}$ dominating $x^*/\alpha$. Since $\tilde{X} \subseteq \mathbb{Z}(X)$, proving the following three statements immediately establishes the theorem: (a) Every optimal solution of $\tilde{\mathrm{P}}$ is a convex combination, i.e. $\tilde{\mathrm{P}}$ evaluates to 1. (b) $\tilde{X}$ is of polynomial size. (c) $\tilde{X}$ can be computed in polynomial-time with respect to the run time of $\mathcal{A}$.

As our first step, we consider the execution of the ellipsoid method on D. Provided that we have a suitable separation oracle, the ellipsoid method yields an optimal solution of D in polynomial-time with respect to the run time of the separation oracle. Recall that a separation oracle identifies hyperplanes separating infeasible solutions from the feasible region. According to Lemma 1, we can add $(\sum_{k=1}^{n}(x_k^*/\alpha)\mu_k) + \nu \leq 1$ to the constraints of D without losing optimal solutions. To construct a suitable separation oracle, we distinguish three cases: First, if $(\sum_{k=1}^{n}(x_k^*/\alpha)\mu_k) + \nu < 1$, we return the newly added constraint as separator. Secondly, if $\mu < 0$, we return the violated non-negativity constraint as separator. Finally, if the previous two cases do not apply, we use $\mathcal{A}$ to compute an integral point $x = \mathcal{A}(\mu)$. Because $\mathcal{A}$ verifies an integrality gap of $\alpha$, it holds true that

$$\Big(\sum_{k=1}^{n} x_k \mu_k\Big) + \nu \geq \max\Big\{\sum_{k=1}^{n} \frac{x'_k}{\alpha}\mu_k \;\Big|\; x' \in X\Big\} + \nu \geq \Big(\sum_{k=1}^{n} \frac{x_k^*}{\alpha}\mu_k\Big) + \nu \geq 1.$$

Consequently, using the constraint associated with $x$ as separator yields the desired separation oracle.

We now direct our attention to the set $\tilde{X}$. Remember that $\tilde{X}$ contains all integral points of $X$ corresponding to constraints the ellipsoid method considered while optimizing D. Because the ellipsoid method only requires a polynomial number of iterations, each of which calls the separation oracle, we know that $\tilde{X}$ only contains a polynomial number of points and the run time is polynomial in $\mathcal{A}$. This proves (b) and (c). Let $\tilde{\mathrm{D}}$ be an LP identical to D where all constraints associated with a point not in $\tilde{X}$ are removed. Since the ellipsoid method never uses these constraints, $\tilde{\mathrm{D}}$ and D must have the same optimal value, namely 1. Furthermore, $\tilde{\mathrm{D}}$ is the dual of $\tilde{\mathrm{P}}$. By strong duality, $\tilde{\mathrm{P}}$ also evaluates to 1. This establishes (a) and concludes the proof. □

### 3.2 Dominating Convex Combinations

We end this section with a brief remark on how to adopt Lavi and Swamy's mechanism design scheme to account for a convex combination $\lambda$ that dominates $x^*/\alpha$. Remember that $\lambda$ models a probability distribution over all feasible outcomes according to which a single outcome $x \in \mathbb{Z}(X)$ is drawn at random in the final step of the scheme. The mechanism is truthful in expectation if the expected value $\mathbb{E}(x)$ of $x$ is equal to $x^*/\alpha$. If $\lambda$ dominates $x^*/\alpha$, then $\mathbb{E}(x)$ also dominates $x^*/\alpha$. To compensate for the surplus in expectation, all components $x_k$ for which $\mathbb{E}(x_k) > 0$ can be set to 0 with probability $1 - (x_k^*/\alpha)/\mathbb{E}(x_k)$. This modification is admissible because $X$ satisfies the packing property. Let $x'$ denote the resulting outcome. Basic probability theory implies that

$$\mathbb{E}(x'_k) = \Pr(x'_k = 1) = \Pr(x_k = 1)\frac{x_k^*/\alpha}{\mathbb{E}(x_k)} = \Pr(x_k = 1)\frac{x_k^*/\alpha}{\Pr(x_k = 1)} = \frac{x_k^*}{\alpha}.$$

Consequently, $x'$ matches $x^*/\alpha$ in expectation and can be used instead of $x$ in the third step of Lavi and Swamy's mechanism design scheme.

## 4 Alternative Decomposition Techniques

In this section we present two alternative decomposition techniques avoiding the notoriously impractical ellipsoid method in Lavi and Swamy's (2011) mechanism design scheme. The gain in efficiency comes with a slight loss in social welfare. Instead of a convex combination $\lambda \in [0,1]^{\mathbb{Z}(X)}$ that dominates $x^*/\alpha$, both decomposition techniques yield a $\lambda$ dominating $(1+\varepsilon)^{-1}x^*/\alpha$ where $\varepsilon > 0$ is an arbitrary accuracy parameter.

### 4.1 The Closest Point Method

We begin with our decomposition technique, which we call *closest point method* (CP method). Starting with a trivial convex combination $\lambda(t) \in [0,1]^{\mathbb{Z}(X)}$, the method repeatedly updates $\lambda(t)$ until it is within close distance to $x^*/\alpha$. In each iteration the new convex combination $\lambda(t+1)$ is chosen to match the closest point to $x^*/\alpha$ on the line segment between $\sum_{x\in\mathbb{Z}(X)} \lambda(t)_x x$ and a sampled integral point $x \in \mathbb{Z}(X)$. After at most $n^2\varepsilon^{-2}$ iterations $\lambda(t)$ is sufficiently close to $x^*/\alpha$ to be converted into a dominating convex combination. For this purpose, the coefficients of $\lambda(t)$ are carefully raised, resulting in a linear combination $\lambda$ that dominates $x^*/\alpha$ and whose coefficients sum up to $\sum_{x\in\mathbb{Z}(X)} \lambda_x \le 1+\varepsilon$. Scaling $\lambda$ down by the sum of its coefficients then yields a convex combination that dominates $(1+\varepsilon)^{-1}x^*/\alpha$.

Refer to Algorithm 2 for a formal implementation of the CP method. In the following we present an intuitive description of the algorithm. The initialization phase constructs a convex combination $\lambda(0)$ consisting of the single point 0, i.e. the origin. Since $X$ satisfies the packing property, we know that 0 is contained in $X$. Let $t$ denote the current iteration. Furthermore, let $\tilde{x}(t) = \sum_{x\in\mathbb{Z}(X)} \lambda(t)_x$ denote the point corresponding to the convex combination $\lambda(t)$. The shortest $\mathrm{L}_1$ distance between $\tilde{x}(t)$ and a point dominating $x^*/\alpha$ is $\sum_{k=1}^{n} \max\{x_k^*/\alpha - \tilde{x}(t)_k, 0\}$. While

**Algorithm 2:** ClosestPointDecomposition

```
Input: Polytope $X \subseteq [0,1]^n$, fractional solution $x^* \in X$, $\alpha$-integrality gap verifier $\mathcal{A}$,
       accuracy parameter $\varepsilon > 0$
Output: Linear combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$
1  $t \leftarrow 0$; $\lambda(t) \leftarrow 0$; $\lambda_0 \leftarrow 1$; $\tilde{x} \leftarrow 0$;
2  while $\sum_{k=1}^n \max\{x_k^*/\alpha - \tilde{x}(t)_k, 0\} > \varepsilon$ do
3      $\mu(t) \leftarrow x^*/\alpha - \tilde{x}(t)$;
4      $x(t) \leftarrow \mathcal{A}((\max\{\mu(t)_k, 0\})_{k=1}^n)$; forall the $k$ such that $\mu(t)_k < 0$ do $x(t)_k \leftarrow 0$;
5      $\delta \leftarrow \arg\min_{\delta' \in [0,1]} \|x^*/\alpha - (\delta'\tilde{x}(t) + (1-\delta')x(t))\|$;
6      $\lambda(t+1) \leftarrow \delta\lambda(t+1)$; $\lambda(t+1)_{x(t)} \leftarrow \lambda(t+1)_{x(t)} + 1 - \delta$;
7      $\tilde{x}(t+1) \leftarrow \sum_{x \in \mathbb{Z}(X)} \lambda(t+1)_x x$;
8      $t \leftarrow t+1$;
9  $\lambda \leftarrow \lambda(t)$; forall the $k$ such that $x_k^*/\alpha > \tilde{x}(t)_k$ do $\lambda_{\hat{k}} \leftarrow \lambda_{\hat{k}} + (x_k^*/\alpha - \tilde{x}(t)_k)$;
10 return $\lambda$;
```

this distance is still greater than $\varepsilon$, $\lambda(t)$ needs to be updated. For this purpose, the algorithm uses the integrality gap verifier $\mathcal{A}$ to sample a new integral point $x(t) \in \mathbb{Z}(X)$. To close in on $x^*/\alpha$, the vector $\mu(t) = x^*/\alpha - \tilde{x}(t)$ appears to be a sensible sample direction. However, as $\mathcal{A}$ can only handle non-negative inputs, the algorithm uses $(\max\{\mu(t)_k, 0\})_{k=1}^n$ instead and sets all components $x(t)_k$ for which $\mu(t)_k < 0$ to 0 afterwards. Since $X$ satisfies the packing property, the resulting $x(t)$ is still contained in $\mathbb{Z}(X)$. The algorithm is now ready to construct a new convex combination $\lambda(t+1)$. Note that every point $x'$ on the line segment between $\tilde{x}(t)$ and $x(t)$ can be expressed as a convex combination $x' = \delta\tilde{x}(t) + (1-\delta)x(t)$. Choosing the $\delta$ that minimizes $\|x^*/\alpha - x'\|$ yields the $x'$ closest to $x^*/\alpha$. The corresponding convex combination $\lambda(t+1)$ is obtained by scaling $\lambda(t)$ with $\delta$ and adding $1-\delta$ to coefficient of $x(t)$. Iteration $t$ ends with an update of $\tilde{x}(t)$.

Once $\tilde{x}(t)$ is close enough to the set of points dominating $x^*/\alpha$, the algorithm begins to construct a linear combination $\lambda$ that dominates $x^*/\alpha$. For this purpose, let $\hat{k}$ be an $n$-dimensional point whose components are all 0 except for component $\hat{k}_k$, which is 1. Since $X$ has a finite integrality gap and satisfies the packing property, we know that $\hat{k}$ is contained in $\mathbb{Z}(X)$. To obtain $\lambda$ , it is sufficient to add $x_k^*/\alpha - \tilde{x}(t)_k$ to the coefficient $\lambda(t)_{\hat{k}}$ whenever $x_k^*/\alpha > \tilde{x}(t)_k$. Clearly, the coefficients of $\lambda$ sum up $1 + \sum_{k=1}^n \max\{x_k^*/\alpha - \tilde{x}(t)_k, 0\} \leq 1 + \varepsilon$, yielding the desired linear combination.

**Theorem 2** *Algorithm 2 computes a linear combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$ that dominates $x^*/\alpha$ and satisfies $\sum_{x \in \mathbb{Z}(X)} \lambda_x \leq 1 + \varepsilon$. The algorithm terminates after at most $\lceil n^2\varepsilon^{-2} \rceil$ iterations of the while loop and $\lambda$ has at most $\lceil n^2\varepsilon^{-2} \rceil + n + 1$ positive components.*

*Proof* From the discretion of Algorithm 2, it should be clear that each $\lambda(t)$ is a valid convex combination. The initial convex combination $\lambda(0)$ consists of a single point and each subsequent iteration adds at most one new point. Once the while loop has terminated, the algorithm constructs a linear combination $\lambda$ that dominates $x^*/\alpha$ by adding at most $n$ more points, i.e. the points $\hat{1}, \hat{2}, \ldots, \hat{n}$. This process raises the sum over the coefficients by an additional quantity of at most $\varepsilon$. To establish the theorem, it is therefore sufficient to proof that the algorithm terminates after at most $\lceil n^2\varepsilon^{-2} \rceil$ iterations.

To estimate the number of iterations, we consider how the length of $\mu(t)$, i.e. the distance between $x^*/\alpha$ and $\tilde{x}(t)$, decreases over time. In particular, our goal is to prove that $\|\mu(\lceil n^2\varepsilon^{-2}\rceil)\| \leq \varepsilon/\sqrt{n}$. Using Hölder's inequality to bound the L$_1$ norm of $\mu(\lceil n^2\varepsilon^{-2}\rceil)$, this implies that

$$\sum_{k=1}^{n} \max\Big\{\frac{x_k^*}{\alpha} - \tilde{x}(\lceil n^2\varepsilon^{-2}\rceil)_k, 0\Big\} \leq \big\|\mu(\lceil n^2\varepsilon^{-2}\rceil)\big\|_1 \leq \sqrt{n}\big\|\mu(\lceil n^2\varepsilon^{-2}\rceil)\big\| \leq \varepsilon.$$

Therefore, if $\|\mu(\lceil n^2\varepsilon^{-2}\rceil)\| \leq \varepsilon/\sqrt{n}$ indeed holds true, then the while loop terminates after iteration $t = \lceil n^2\varepsilon^{-2}\rceil$ or before.

Let $t$ be an iteration before termination. To show $\|\mu(\lceil n^2\varepsilon^{-2}\rceil)\| \leq \varepsilon/\sqrt{n}$, we first argue that $\tilde{x}(t)$ and $x(t)$ are separated by a hyperplane $H$ through $x^*/\alpha$ perpendicular to $\mu(t)$. We say that $H$ separates $\tilde{x}(t)$ from $x(t)$ if

$$\sum_{k=1}^{n} x(t)_k\mu(t)_k \geq \sum_{k=1}^{n} \frac{x_k^*}{\alpha}\mu(t)_k > \sum_{k=1}^{n} \tilde{x}(t)_k\mu(t)_k. \tag{1}$$

To prove the first inequality of (1), remember that the algorithm sets all components $x(t)_k = 0$ for which $\mu_k < 0$. Consequently, we get

$$\begin{aligned}\sum_{k=1}^{n} x(t)_k\mu(t)_k &= \sum_{k=1}^{n} x(t)_k \max\{\mu(t)_k, 0\}\\ &= \sum_{k=1}^{n} \mathcal{A}\big((\max\{\mu(t)_k, 0\})_{k=1}^n\big)_k \max\{\mu(t)_k, 0\}.\end{aligned}$$

Since $\mathcal{A}$ returns a point that is an $\alpha$-approximation in the direction given as its argument, we conclude further that

$$\begin{aligned}\sum_{k=1}^{n} x(t)_k\mu(t)_k &\geq \max\Big\{\sum_{k=1}^{n} \frac{x_k'}{\alpha}\max\{\mu(t)_k, 0\} \,\Big|\, x' \in X\Big\}\\ &\geq \sum_{k=1}^{n} \frac{x_k^*}{\alpha}\max\{\mu(t)_k, 0\} \geq \sum_{k=1}^{n} \frac{x_k^*}{\alpha}\mu(t)_k.\end{aligned}$$

This proves the first inequality of (1). We continue with the second inequality. Because the algorithm has not terminated yet, it holds true that

$$0 < \sum_{k=1}^{n}\Big(\frac{x_k^*}{\alpha} - \tilde{x}(t)_k\Big)^2 = \sum_{k=1}^{n}\Big(\Big(\frac{x_k^*}{\alpha}\Big)^2 - 2\frac{x_k^*}{\alpha}\tilde{x}(t)_k + \tilde{x}(t)_k^2\Big).$$

Rearranging this term yields

$$\sum_{k=1}^{n}\Big(\Big(\frac{x_k^*}{\alpha} - \tilde{x}(t)_k\Big)\tilde{x}(t)_k\Big) < \sum_{k=1}^{n}\Big(\Big(\frac{x_k^*}{\alpha} - \tilde{x}(t)_k\Big)\frac{x_k^*}{\alpha}\Big).$$

As $\mu(t) = x^*/\alpha - \tilde{x}(t)$, this establishes the second inequality of (1) and proves that $\tilde{x}(t)$ and $x(t)$ are separated by $H$.

Next, let $z(t)$ be the point at the intersection of $H$ with the line segment between $\tilde{x}(t)$ and $x(t)$. This intersection exists because $H$ separates $\tilde{x}(t)$ from

$x(t)$. Consider the triangle formed by the points $\tilde{x}(t)$, $x^*/\alpha$ and $z(t)$. Since $H$ is perpendicular to $\mu(t)$, i.e. the vector $x^*/\alpha - \tilde{x}(t)$, there is a right angle at $x^*/\alpha$. Furthermore, remember that $\tilde{x}(t+1)$ is chosen as the point on the line segment between $\tilde{x}(t)$ and $x(t)$ that is closest to $x^*/\alpha$. Consequently, $\tilde{x}(t+1)$ is the perpendicular foot on the leg opposing $x^*/\alpha$. The corresponding altitude is $\|x^*/\alpha - \tilde{x}(t+1)\| = \|\mu(t+1)\|$. Using basic geometry, we can express the squared altitude of such a right triangle as

$$\|\mu(t+1)\|^2 = \frac{\|x^*/\alpha - \tilde{x}(t)\|^2 \|x^*/\alpha - z(t)\|^2}{\|x^*/\alpha - \tilde{x}(t)\|^2 + \|x^*/\alpha - z(t)\|^2} = \frac{\|\mu(t)\|^2 \|x^*/\alpha - z(t)\|^2}{\|\mu(t)\|^2 + \|x^*/\alpha - z(t)\|^2}.$$

Although we do not know the exact distance between $x^*/\alpha$ and $z(t)$ we can bound the distance by

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

as both points are contained in the standard hypercube $[0,1]^n$. For $\|\mu(t+1)\|^2$ this yields the recurrence relation $\|\mu(t+1)\|^2 \leq (\|\mu(t)\|^2 n)/(\|\mu(t)\|^2 + n)$. Rearranging this inequality to $n/\|\mu(t+1)\|^2 \geq 1 + n/\|\mu(t)\|^2$, enables us to easily derive a bound on its solution. The result is $n/\|\mu(t+1)\|^2 \geq (t+1) + n/\|\mu(0)\|^2$. Rearranging once more yields

$$\|\mu(t+1)\|^2 \leq \frac{\|\mu(0)\|^2 n}{\|\mu(0)\|^2 (t+1) + n} \leq \frac{n^2}{n(t+1) + n} = \frac{n}{t+2}.$$

The second inequality holds true because $\mu(0)$ is contained in the standard hypercube $[0,1]^n$, implying that $\|\mu(0)\|^2 \leq n$. Setting $t = \lceil n^2 \varepsilon^{-2} \rceil$, we get

$$\|\mu(\lceil n^2 + \varepsilon^{-2} \rceil)\| \leq \sqrt{\frac{n}{\lceil n^2 \varepsilon^{-2} \rceil + 1}} \leq \sqrt{\frac{n}{n^2 \varepsilon^{-2}}} = \frac{\varepsilon}{\sqrt{n}},$$

which concludes the proof. $\square$

## 4.2 The Multiplicative Weights Update Method

Next, we compare the CP method to a decomposition technique proposed by Elbassioni et al (2015). Their decomposition technique, which is based on the MWU method, is an adaptation of Khandekar's (2004) fully polynomial-time approximation scheme for covering LPs. Applying this algorithm to the linear program Q introduced in Section 2 yields a linear combination $\lambda$ that dominates $x^*/\alpha$ and whose coefficients sum up to at most $1+\varepsilon$. To highlight similarities and differences between the two decomposition techniques, we present a slightly modified version of Elbassioni et al.'s work. In particular, we apply Khandekar's algorithm directly to the decomposition LP, using similar notation to that of the previous subsection whenever appropriate. See Algorithm 3 for a formal implementation. Simplifying the algorithm also allows us to slightly improve the approximation bounds from $1 + 4\varepsilon$ to $1 + \varepsilon$.

We proceed with an intuitive description of Algorithm 3. Starting with the initially empty linear combination $\lambda(0)$, the algorithm repeatedly adds integral

**Algorithm 3:** MultiplicativeWeightsUpdateDecomposition

```
Input: Polytope $X \subseteq [0,1]^n$, fractional solution $x^* \in X$, $\alpha$-integrality gap verifier $\mathcal{A}$,
       accuracy parameter $0 < \varepsilon \leq 1/2$
Output: Linear combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$
1  $t \leftarrow 0$; $\lambda(t) \leftarrow 0$; $\tilde{x}(t) \leftarrow 0$; $N(t) \leftarrow \{1, 2, \ldots, n\}$;
2  while $N(t) \neq \emptyset$ do
3      $\mu(t) \leftarrow 0$; forall the $k \in N(t)$ do $\mu(t)_k \leftarrow (1-\varepsilon)^{(\alpha\tilde{x}(t)_k)/x_k^*}$;
4      $x(t) \leftarrow \mathcal{A}((\mu(t)_k/x_k^*)_{k=1}^n)$;
5      $\lambda(t+1) \leftarrow \lambda(t)$; $\lambda(t+1)_{x(t)} \leftarrow \lambda(t+1)_{x(t)} + \alpha^{-1}\min\{x_k^*/x(t)_k \mid k \in N(t)\}$;
6      $\tilde{x}(t+1) \leftarrow \sum_{x \in \mathbb{Z}(X)} \lambda(t+1)_x x$;
7      $N(t+1) \leftarrow \{k \mid \tilde{x}(t+1)_k < \log(n)\varepsilon^{-2}(x_k^*/\alpha)\}$;
8      $t \leftarrow t+1$;
9  $\lambda \leftarrow (\varepsilon^2/\log(n))\lambda(t)$;
10 return $\lambda$;
```

points $x(t) \in \mathbb{Z}(X)$ to $\lambda(t)$ until all dimensions $k$ are *inactive*. We say that $k$ is inactive if $\tilde{x}(k)_t \geq \log(n)\varepsilon^{-2}(x_k^*/\alpha)$. For convenience, we denote the set of active dimensions at time $t$ by $N(t) = \{k \mid \tilde{x}(t)_k < \log(n)\varepsilon^{-2}(x_k^*/\alpha)\}$. As $x^*$ is positive, all dimensions are initially active, i.e. $N(0) = \{1, 2, \ldots, n\}$.

To update $\lambda(t)$, the algorithm samples a new integral point $x(t)$ via the integrality gap verifier $\mathcal{A}$. The sample direction is determined as follows: First, the algorithm constructs the vector $\mu(t)$ by setting $\mu(t)_k = (1-\varepsilon)^{(\alpha\tilde{x}(t)_k)/x_k^*}$ if $k \in N(t)$, and $\mu(t)_k = 0$ otherwise. This way the contribution of an active dimension $k$ is proportional to the ratio $\tilde{x}(t)_k/x_k^*$ while inactive dimensions do not contribute at all. Note that the exponent of $(1-\varepsilon)$ is finite as $x^*$ is positive. Secondly, the algorithm scales $\mu(t)$ by $x^*$ to obtain the sampling vector $(\mu(t)_k/x_k^*)_{k=1}^n$. Again all components of this vector are finite as $x^*$ is positive. After $x(t)$ has been computed, the corresponding coefficient $\lambda(t)_{x(t)}$ is raised to obtain a new convex combination $\lambda(t+1)$. For this purpose, the algorithm identifies the minimum ratio $x_k^*/x(t)_k$ among all active dimensions $k \in N(t)$, multiplies it by $\alpha^{-1}$ and adds it to $\lambda(t)_{x(t)}$. The iteration ends with an update of $\tilde{x}(t)$ and $N(t)$.

Once all components are inactive, the while loop terminates. At this point $\tilde{x}(t)$ weakly dominates $\log(n)\varepsilon^{-2}(x^*/\alpha)$. Scaling $\lambda(t)$ by a factor of $\varepsilon^2/\log(n)$ eventually yields a linear combination $\lambda$ dominating $x^*/\alpha$. Furthermore, the coefficients of $\lambda$ sum up to at most $1+\varepsilon$ as the following theorem states:

**Theorem 3** *Algorithm 3 computes a linear combination $\lambda \in \mathbb{R}_{\geq 0}^{\mathbb{Z}(X)}$ that dominates $x^*/\alpha$ and satisfies $\sum_{x \in \mathbb{Z}(X)} \lambda_x \leq 1+\varepsilon$. The algorithm terminates after at most $n\lceil\log(n)\varepsilon^{-2}\rceil$ iterations of the while loop and $\lambda$ has at most $n\lceil\log(n)\varepsilon^{-2}\rceil$ positive components.*

*Proof* From the description of Algorithm 3, it should be clear that the final linear combination $\lambda$ returned by the algorithm dominates $x^*/\alpha$. Furthermore, each iteration of the while loop adds at most one new integral point $x(t) \in \mathbb{Z}(X)$ to $\lambda(t)$. To establish the theorem, it is therefore sufficient to prove the following two statements: (a) The algorithm terminates after at most $n\lceil\log(n)\varepsilon^{-2}\rceil$ iterations. (b) The coefficients of $\lambda(t)$ sum up to at most $\log(n)\varepsilon^{-2}(1+\varepsilon)$ upon termination of the while loop.

We begin with (a). Our goal is to show that each iteration increases some component $\tilde{x}(t)_k$ with $k \in N(t)$ by $x^*_k/\alpha$. We also show that no component of $\tilde{x}(t)$ ever decreases. Because dimension $k$ becomes inactive once $\tilde{x}(t)_k \geq \log(n)\varepsilon^{-2}(x^*_k/\alpha)$, this immediately yields the desired bound on the number of iterations.

Let $t$ be an arbitrary iteration before termination and let $k$ be an active dimension that maximizes the ratio $x(t)_{k'}/x^*_{k'}$ among all $k' \in N(t)$. As our first step we argue that $x(t)_k = 1$. For the sake of contradiction, assume that $x(t)_k \neq 1$. Because all components of $x(t)$ are either 0 or 1, this implies $x(t)_k = 0$. Considering that $k$ maximizes the ratio $x(t)_{k'}/x^*_{k'}$, we conclude that $\alpha \sum_{k'=1}^n (\mu(t)_{k'}/x^*_{k'})x(t)_{k'} = 0$. Remember that the vector $(\mu(t)_{k'}/x^*_{k'})_{k'=1}^n$ corresponds to the sample direction of $x(t)$. Since $\mathcal{A}$ verifies an integrality gap of $\alpha$, it holds true that

$$\begin{aligned}\alpha \sum_{k'=1}^{n} \frac{\mu(t)_{k'}}{x^*_{k'}} x(t)_{k'} &\geq \max\Big\{ \sum_{k'=1}^{n} \frac{\mu(t)_{k'}}{x^*_{k'}} x'_{k'} \;\Big|\; x' \in X \Big\} \\ &\geq \sum_{k'=1}^{n} \frac{\mu(t)_{k'}}{x^*_{k'}} x^*_{k'} = \|\mu(t)\|_1 \geq \mu(t)_k.\end{aligned} \tag{2}$$

Being an active dimension, $k$ satisfies $\mu(t)_k = (1-\varepsilon)^{(\alpha \tilde{x}(t)_k)/x^*_k} > 0$; a contradiction to the observation that $\alpha \sum_{k'=1}^n (\mu(t)_{k'}/x^*_{k'})x(t)_{k'} = 0$.

Now that $x(t)_k = 1$ has been established, we investigate how $\tilde{x}(t)_k$ increases in iteration $t$. By choice of $k$, we have $x^*_k/x(t)_k = \min\{x^*_{k'}/x(t)_{k'} \mid k' \in N(t)\}$. Remember that $\alpha^{-1} \min\{x^*_{k'}/x(t)_{k'} \mid k' \in N(t)\}$ is the weight assigned to $x(t)$ when added to $\lambda(t)$. Consequently, it holds true that $\tilde{x}(t+1) = \tilde{x}(t) + (\alpha^{-1} x^*_k/x(t)_k)x(t)$. Clearly, no component of $\tilde{x}(t+1)$ is decreased in this process. Furthermore, $\tilde{x}(t+1)_k$ is increased by $x^*_k/\alpha$ as $\tilde{x}(t+1)_k = \tilde{x}(t)_k + (\alpha^{-1}x^*_k/x(t)_k)x(t)_k = \tilde{x}(t)_k + (x^*_k/\alpha)$. This proves (a).

We continue with (b). Let $q$ be the value of $t$ upon termination. Our goal is to show that $\sum_{x \in \mathbb{Z}(X)} \lambda(q)_x \leq (1+\varepsilon)\varepsilon^{-2}\log(n)$ is valid. As our first step, we analyze the decrease in the $\mathrm{L}_1$ norm of $\mu(t)$ over time. For this purpose, we assume that $1 \leq t < q$ and define $\delta(t) = \min\{x^*_k/x(t)_k \mid k \in N(t)\}$. Remember that $\tilde{x}(t) = \tilde{x}(t-1) + \alpha^{-1}\delta(t-1)x(t-1)$. In the process of proving (a), we have argued that the components of $\tilde{x}(t)$ monotonically increase over time. As a result, $N(t)$ monotonically decreases, i.e. $N(t) \subseteq N(t-1)$. Applying the definition of $\mu(t)$ we conclude that

$$\begin{aligned}\|\mu(t)\|_1 &= \sum_{k \in N(t)} (1-\varepsilon)^{(\alpha \tilde{x}(t)_k)/x^*_k} \leq \sum_{k \in N(t-1)} (1-\varepsilon)^{(\alpha \tilde{x}(t)_k)/x^*_k} \\ &= \sum_{k \in N(t-1)} (1-\varepsilon)^{(\alpha \tilde{x}(t-1)_k + \delta(t-1)x(t-1)_k)/x^*_k} \\ &= \sum_{k \in N(t-1)} \mu(t-1)_k (1-\varepsilon)^{(\delta(t-1)x(t-1)_k)/x^*_k}.\end{aligned} \tag{3}$$

We will now take a closer look at $(1-\varepsilon)^{(\delta(t-1)x(t-1)_k)/x^*_k}$. By definition of $\delta(t-1)$, we can upper bound the exponent by

$$\delta(t-1)\frac{x(t-1)_k}{x^*_k} = \min\Big\{ \frac{x^*_{k'}}{x(t-1)_{k'}} \;\Big|\; k' \in N(t-1) \Big\} \frac{x(t-1)_k}{x^*_k} \leq 1$$

for all $k \in N^{t-1}$. Consequently, it holds true that

$$(1-\varepsilon)^{(\delta(t-1)x(t-1)_k)/x^*_k} + \varepsilon\Big(\delta(t-1)\frac{x(t-1)_k}{x^*_k}\Big) \le (1-\varepsilon) + \varepsilon = 1,$$

or simply $(1-\varepsilon)^{(\delta(t-1)x(t-1)_k)/x^*_k} \le 1 - \varepsilon(\delta(t-1)x(t-1)_k)/x^*_k$. Applying this inequality to the bound on $\|\mu(t)\|_1$ established earlier yields

$$\begin{aligned}\|\mu(t)\|_1 &\le \sum_{k\in N(t-1)} \mu(t-1)_k\Big(1-\varepsilon\delta(t-1)\frac{x(t-1)_k}{x^*_k}\Big)\\ &= \|\mu(t-1)\|_1\Big(1-\frac{\varepsilon\delta(t-1)}{\|\mu(t-1)\|_1}\sum_{k\in N(t-1)}\frac{\mu(t-1)_k}{x^*_k}x(t-1)_k\Big).\end{aligned}$$

According to inequality (2), the term $\sum_{k\in N(t-1)}(\mu(t-1)_k/x^*_k)x(t-1)_k$ is lower bounded by $\|\mu(t-1)\|/\alpha$, implying that

$$\|\mu(t)\|_1 \le \|\mu(t-1)\|_1\Big(1-\frac{\varepsilon\delta(t-1)}{\|\mu(t-1)\|_1}\frac{\|\mu(t-1)\|_1}{\alpha}\Big) = \|\mu(t-1)\|_1\Big(1-\frac{\varepsilon\delta(t-1)}{\alpha}\Big).$$

Finally, using the inequality $1-y \le e^{-y}$, which holds true for all $y \in \mathbb{R}$, we obtain the following recurrence relation

$$\|\mu(t)\|_1 \le \|\mu(t-1)\|_1 \exp\Big(-\frac{\varepsilon\delta(t-1)}{\alpha}\Big). \tag{4}$$

As our next step, we bound $(1-\varepsilon)^{(\alpha\tilde{x}(q)_k)/x^*_k}$, where $k \in N(q-1)$ is an active component of the last iteration $q-1$ before termination. Remember that the sum $\sum_{k'\in N(q-1)}(1-\varepsilon)^{(\alpha\tilde{x}(q)_{k'})/x(q)^*_{k'}}$ arose as an intermediate term in inequality (3) while establishing recurrence relation (4). Therefore, this sum is less or equal to $\|\mu(q-1)\|_1\exp(-\alpha^{-1}\varepsilon\delta(q-1))$, implying that

$$(1-\varepsilon)^{(\alpha\tilde{x}(q)_k)/x^*_k} \le \sum_{k'\in N(q-1)}(1-\varepsilon)^{(\alpha\tilde{x}(q)_{k'})/x(q)^*_{k'}} \le \|\mu(q-1)\|_1\exp\Big(-\frac{\varepsilon\delta(q-1)}{\alpha}\Big).$$

Applying recurrence relation (4) $q-1$ times yields

$$\begin{aligned}(1-\varepsilon)^{(\alpha\tilde{x}(q)_k)/x^*_k} &\le \|\mu(0)\|_1\exp\Big(-\frac{\varepsilon\delta(q-1)}{\alpha}\Big)\prod_{t=0}^{q-2}\exp\Big(-\frac{\varepsilon\delta(t)}{\alpha}\Big)\\ &= n\exp\Big(-\frac{\varepsilon}{\alpha}\sum_{t=0}^{q-1}\delta(t)\Big).\end{aligned} \tag{5}$$

The last equality uses the observation that $\|\mu(0)\|_1 = n$ as all components of $\mu(0)$ are initially 1.

Finally, we consider the coefficients of $\lambda(q)$. In each iteration $t$, the algorithm raises one coefficient by an additional value of $\alpha^{-1}\delta(t)$. Since all coefficients of $\lambda(0)$ are initially 0, $\sum_{x\in\mathbb{Z}(X)}\lambda(q)_x$ is equal to $\alpha^{-1}\sum_{t=0}^{q-1}\delta(t)$. We may therefore substitute one term with the other. In case of inequality (5), this results in

$(1-\varepsilon)^{(\alpha\tilde{x}(q)_k)/x_k^*} \leq n\exp(-\varepsilon\sum_{x\in\mathbb{Z}(X)}\lambda(q)_x)$ for any $k \in N(q-1)$. After taking logarithms and rearranging we obtain

$$\sum_{x\in\mathbb{Z}(X)}\lambda(q)_x \leq \frac{1}{\varepsilon}\Big(\log(n) - \frac{\alpha\tilde{x}(q)_k}{x_k^*}\log(1-\varepsilon)\Big).$$

Although $k$ is an active dimension in iteration $q-1$ it cannot be active upon termination anymore, i.e. $\tilde{x}(q)_k \geq \log(n)\varepsilon^{-2}(x_k^*/\alpha)$. Consequently, we get

$$\sum_{x\in\mathbb{Z}(X)}\lambda(q)_x \leq \frac{1}{\varepsilon}\Big(\log(n) - \log(n)\varepsilon^{-2}\log(1-\varepsilon)\Big) = \big(1-\varepsilon^2\log(1-\varepsilon)\big)\frac{\log(n)}{\varepsilon}.$$

If we choose $\varepsilon$ from the interval $(0,1/2]$, where $1/2$ provides a lower bound on the root of $\log(1-y)y^3+1$, then $1-\varepsilon^2\log(1-\varepsilon) \leq (1+\varepsilon)/\varepsilon$ is satisfied. As a result, $\sum_{x\in\mathbb{Z}(X)}\lambda(q)_x \leq (1+\varepsilon)\varepsilon^{-2}\log(n)$ holds true, which concludes the proof. $\square$

## 5 Experimental Evaluation

For the experimental comparison between our decomposition technique and Elbassioni et al's (2015), we come back to the retail logistics scenario from Section 2. In this scenario $I$ carriers must plan their daily tour between $K$ warehouses. As each warehouse has a limited capacity at the loading docks, carriers may experience long waiting times for loading and unloading their cargo if they do not coordinate. To reduce waiting times, Karänke et al (2015) suggest that carriers reserve loading docs in advance via a multi-unit combinatorial auction. Due to the complexity of the corresponding WDP, this retail logistic scenario poses an appealing setting for Lavi and Swamy's (2011) mechanism design scheme. In what follows, we briefly describe the transportation network we are using, the bid generation, and the main treatment variables of the experiments. We then conclude with an evaluation of the experimental results.

### 5.1 Experimental Setup

We draw on data from the distribution network of a German retailer, whose transportation network provides the distances and respective travel times between 65 locations of warehouses. In each simulation, a random subset of locations is chosen. From this set, carriers are assigned a distinct depot, marking the start and end point of their tour, and a random set of warehouses they need to visit. Let $S$ denote number of warehouses per carrier, i.e. the number of stops on a carrier's tour excluding the start and endpoint. The loading docs of the warehouses have a capacity of 2 for all time slots of the day. Each time slot is 10 minutes, resulting in a total of $T = 90$ time slots. To consider different problem sizes, we simulate problem instances from 12 up to 30 carriers and 10 upto 30 warehouses in step sizes of 2. Furthermore, the number of warehouses each carrier must visit ranges from 3 to 5. Overall, we run a full factorial experiment with the treatment variables described in Table 1 leading to 3267 experiments.

In each simulation, carriers compute all possible routes to visit their warehouses and submit bids on routes that are not taking longer than 10 percent of the optimal

| Variable | Values |
|---|---|
| Number of carriers ($I$) | $\{10, \ldots, 30\}$ |
| Number of warehouses ($K$) | $\{10, \ldots, 30\}$ |
| Warehouses per carrier ($S$) | $\{3, 4, 5\}$ |
| Precision ($\varepsilon$) | $\{0.001, 0.01, 0.05\}$ |
| Decomposition technique | $\{CP, MWU, ellipsoid\}$ |

**Table 1** Treatment variables

round trip time. The problem sizes of this traveling salesman problem is small enough that we can compute exact solutions. After the routes for each carrier have been determined, we compute a bid price for each route, which is a package bid in the auction. The bid price is proportional to the time saved per day. All bid data generated is available upon request for replication studies.

The simulation is implemented in the Java programming language. The commercial mathematical programming solver Gurobi Optimizer v5.6.3 is used for solving the optimization problems. Experiments were executed on a computer with an Intel Core i7-3612 QM CPU (4 cores, 2.1GHz) and 8GB RAM. We do not run the ellipsoid method to the end, but terminate as soon as a decomposition is possible.

## 5.2 Results

*We find that the CP method is substantially faster in terms of runtime and uses less iterations (or terms in the convex decomposition) than the MWU method for high precision values of* $\varepsilon \leq 0.01$*. For low precision values of* $\varepsilon = 0.05$ *the differences between MWU and CP are not significant. The CP algorithm and the MWU algorithms are both substantially faster than the ellipsoid method.*

Figure 1 illustrates the average number of iterations with respect to the precision parameter $\varepsilon$ across all combinations of $I$, $K$ and $S$. The figure shows clearly that the average case performance of the CP method is much less sensitive to the value of $\varepsilon$ than the MWU method. Furthermore, as indicated by the dashed line, CP and MWU both need much fewer iterations than the ellipsoid method. Table 2 lists the average number of iterations and the average runtime for CP and MWU not only with respect to $\varepsilon$, but also with respect to the problem size, in particular the number of carriers $I$. The average is taken across all combinations of $K$ and $S$. We also provide the average number of iterations and the average runtime of the ellipsoid method.

For our statistical evaluation, we use a linear regression to analyze the differences among CP, MWU, and the ellipsoid method and to control for the impact of $I$, $K$, and $S$ in the different problem instances. The results across different levels of precision vary significantly such that we report them separately.

For $\varepsilon = 0.001$ the MWU method and the ellipsoid method need 81.51 and 141.89 iterations more than the CP method on average. If we use the number of iterations as dependent variable and set the CP method as baseline, then the difference in the number of iterations is significant at $p < 0.0001$. Similarly, the CP method is on average 13.95 seconds faster than the MWU method and 6.09 seconds faster than the ellipsoid method, which is significant at $p < 0.0001$. With

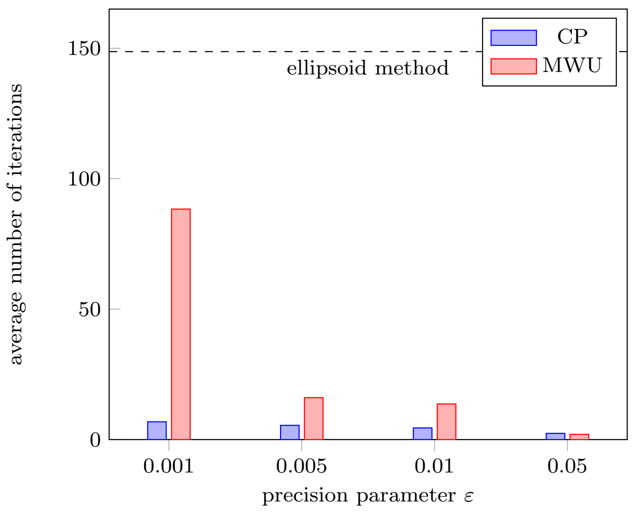

**Fig. 1** Number of iterations with respect to $\varepsilon$ averaging over all $I$, $K$ and $S$

decreasing precision the performance differences between CP and MWU become smaller. For instance, choosing $\varepsilon = 0.01$ results in an average of 9.16 iterations more for the MWU when compared to the CP method. The difference is not significant. Considering the runtime, the CP method is 1.46 seconds faster than the MWU method, a difference that is only significant at $p < 0.01$. The performance difference becomes even less significant for $\varepsilon = 0.05$. In this case, the MWU requires 0.37 fewer iterations on average than the CP method and its average runtime is 0.29 seconds faster than the CP method. Both values are not significant. For a complete list of the regression coefficients and their significance refer to Table 3.

To better understand the performance differences between the CP method and MWU the method, it is interesting to consider the convergence of both techniques. Figure 2 shows how the distance between the current convex combination and the scaled fractional solution decreases over time for a sample instance with $I = 20$, $k = 10$, $S = 5$ and $\varepsilon = 0.01$. It is striking how the progress of the MWU method is almost linear whereas the CP method makes rapid progress at first and slows down towards the end resulting in a hyperbolic plot. The reason might be that the MWU method updates the convex combination much more carefully than the CP method. An interesting question for future research would be to analytically analyze the average case performance of the MWU method.

## References

Bichler M, Davenport A, Hohner G, Kalagnanam J (2006) Industrial procurement auctions. In: Cramton P, Shoham Y, Steinberg R (eds) Combinatorial Auctions, MIT Press

Bland RG, Goldfarb D, Todd MJ (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: ACM Conference on Electronic Commerce (EC 05), Vancouver, Canada

| | | Number of iterations per $\varepsilon$ | | | | Runtime in seconds per $\varepsilon$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| $I$ | Technique | 0.001 | 0.005 | 0.01 | 0.05 | 0.001 | 0.005 | 0.01 | 0.05 |
| 10 | CP | 3.91 | 3.24 | 2.06 | 1.12 | 0.06 | 0.05 | 0.04 | 0.03 |
| 10 | Ellipsoid | 13.58 | 13.58 | 13.58 | 13.58 | 0.09 | 0.08 | 0.09 | 0.09 |
| 10 | MWU | 63.33 | 12.18 | 9.79 | 2.00 | 1.45 | 0.21 | 0.19 | 0.02 |
| 12 | CP | 3.88 | 3.48 | 2.42 | 1.15 | 0.10 | 0.08 | 0.08 | 0.05 |
| 12 | Ellipsoid | 16.09 | 16.09 | 16.09 | 16.09 | 0.14 | 0.13 | 0.14 | 0.14 |
| 12 | MWU | 67.00 | 12.73 | 10.61 | 2.00 | 2.44 | 0.34 | 0.34 | 0.04 |
| 14 | CP | 4.79 | 3.88 | 2.85 | 1.45 | 0.18 | 0.13 | 0.13 | 0.08 |
| 14 | Ellipsoid | 35.48 | 35.48 | 35.48 | 35.48 | 0.44 | 0.40 | 0.44 | 0.43 |
| 14 | MWU | 72.33 | 13.58 | 11.52 | 2.00 | 3.87 | 0.53 | 0.55 | 0.05 |
| 16 | CP | 5.42 | 4.39 | 3.36 | 1.64 | 0.29 | 0.21 | 0.22 | 0.12 |
| 16 | Ellipsoid | 38.27 | 38.27 | 38.27 | 38.27 | 0.74 | 0.49 | 0.55 | 0.53 |
| 16 | MWU | 79.15 | 14.24 | 12.12 | 2.00 | 5.85 | 0.79 | 0.83 | 0.07 |
| 18 | CP | 6.15 | 4.97 | 4.21 | 1.91 | 0.42 | 0.32 | 0.35 | 0.18 |
| 18 | Ellipsoid | 84.52 | 84.52 | 84.52 | 84.52 | 1.88 | 1.74 | 1.90 | 1.85 |
| 18 | MWU | 83.94 | 15.39 | 13.09 | 2.00 | 8.37 | 1.15 | 1.20 | 0.10 |
| 20 | CP | 6.67 | 5.33 | 4.52 | 2.06 | 0.60 | 0.45 | 0.50 | 0.25 |
| 20 | Ellipsoid | 94.12 | 94.12 | 94.12 | 94.12 | 2.38 | 2.19 | 2.37 | 2.33 |
| 20 | MWU | 87.67 | 16.06 | 13.91 | 2.00 | 11.40 | 1.55 | 1.66 | 0.13 |
| 22 | CP | 7.03 | 5.73 | 4.85 | 2.52 | 0.78 | 0.60 | 0.66 | 0.38 |
| 22 | Ellipsoid | 121.33 | 121.33 | 121.33 | 121.33 | 3.86 | 3.54 | 3.86 | 3.81 |
| 22 | MWU | 95.21 | 16.94 | 14.48 | 2.00 | 15.46 | 2.08 | 2.19 | 0.16 |
| 24 | CP | 7.94 | 6.24 | 5.45 | 2.94 | 1.12 | 0.82 | 0.91 | 0.55 |
| 24 | Ellipsoid | 190.79 | 190.79 | 190.79 | 190.79 | 7.62 | 6.89 | 7.61 | 7.57 |
| 24 | MWU | 99.00 | 17.76 | 15.15 | 2.00 | 20.18 | 2.75 | 2.87 | 0.19 |
| 26 | CP | 9.00 | 6.88 | 5.85 | 3.36 | 1.53 | 1.12 | 1.20 | 0.77 |
| 26 | Ellipsoid | 212.76 | 212.76 | 212.76 | 212.76 | 9.22 | 8.48 | 9.32 | 9.25 |
| 26 | MWU | 102.15 | 18.61 | 16.03 | 2.00 | 25.36 | 3.55 | 3.71 | 0.23 |
| 28 | CP | 9.97 | 7.76 | 6.73 | 3.79 | 2.03 | 1.49 | 1.65 | 1.00 |
| 28 | Ellipsoid | 371.03 | 371.03 | 371.03 | 371.03 | 20.63 | 18.64 | 20.76 | 20.54 |
| 28 | MWU | 108.27 | 19.33 | 16.42 | 2.00 | 32.42 | 4.42 | 4.60 | 0.27 |
| 30 | CP | 10.27 | 8.00 | 6.97 | 4.18 | 2.53 | 1.85 | 2.03 | 1.33 |
| 30 | Ellipsoid | 457.82 | 457.82 | 457.82 | 457.82 | 31.17 | 28.19 | 31.35 | 30.90 |
| 30 | MWU | 113.64 | 19.97 | 16.94 | 2.03 | 40.06 | 5.49 | 5.68 | 0.32 |

**Table 2** Number of iterations and runtime with respect to $I$ and $\varepsilon$, averaging over all $K$ and $S$

| | | Precision values $\varepsilon$ | | | |
|---|---|---|---|---|---|
| | Technique | 0.001 | 0.005 | 0.01 | 0.05 |
| Runtime | MWU | 13.95(*) | 1.43(*) | 1.46(.) | -0.29(-) |
| | Ellipsoid | 6.95(*) | 5.79(*) | 6.40(*) | 6.61(*) |
| Iterations | MWU | 81.51(*) | 10.63(-) | 9.16(-) | -0.37(-) |
| | Ellipsoid | 141.89(*) | 143.26(*) | 144.23(*) | 146.33(*) |

**Table 3** Regression coefficients describing estimated differences to the CP technique. (*): significant at $p < 0.0001$, (.): significant at $p < 0.01$, (-): not significant

Briest P, Krysta P, Vöcking B (2005) Approximation techniques for utilitarian mechanism design. In: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing, ACM, pp 39–48

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

Clarke E (1971) Multipart pricing of public goods. Public Choice XI:17–33

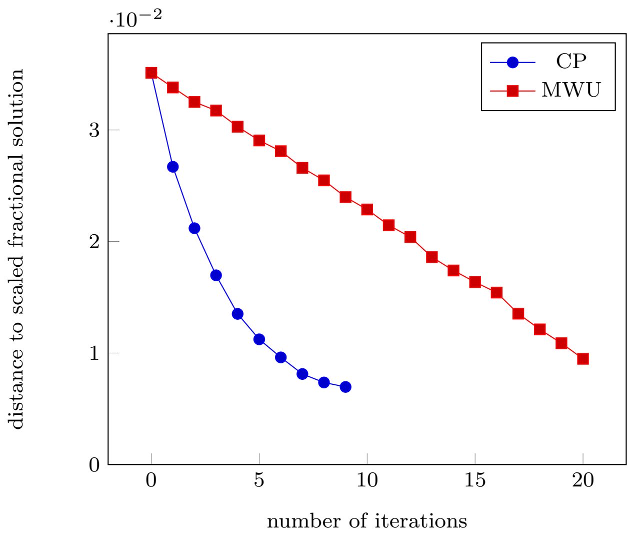

**Fig. 2** Convergence plot for a sample instance with $I = 20$, $k = 10$, $S = 5$ and $\varepsilon = 0.01$

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

Elbassioni K, Mehlhorn K, Ramezani F (2015) Algorithmic Game Theory: 8th International Symposium, SAGT 2015, Saarbrücken, Germany, September 28-30, 2015. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, chap Towards More Practical Linear Programming-Based Techniques for Algorithmic Mechanism Design, pp 98–109

Elbassioni K, Mehlhorn K, Ramezani F (2016) Towards more practical linear programming-based techniques for algorithmic mechanism design. Theory of Computing Systems 59(4):641–663

Groves T (1973) Incentives in teams. Econometrica 41:617–631

Karänke P, Bichler M, Minner S (2015) Retail warehouse loading dock coordination by core-selecting package auctions.

Khandekar R (2004) Lagrangian relaxation based algorithms for convex programming problems. PhD thesis, Indian Institute of Technology Delhi

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

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

Nisan N (2006) Bidding languages. In: Cramton P, Shoham Y, Steinberg R (eds) Combinatorial Auctions, MIT Press, Cambridge, MA

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

Nisan N, Roughgarden T, Tardos E, Vazirani VV (2007) Algorithmic Game Theory. Cambridge University Press

Vickrey W (1961) Counterspeculation, auctions, and competitive sealed tenders. Journal of Finance (3):8–37