Towards More Practical Linear Programming-Based Techniques for Algorithmic Mechanism Design

Khaled Elbassioni[1], Kurt Mehlhorn[2], and Fahimeh Ramezani[2(✉)]

[1] Masdar Institute of Science and Technology, Abu Dhabi, UAE
kelbassioni@masdar.ac.ae
[2] Max Planck Institute for Informatics, Campus E1 4, 66123 Saarbrucken, Germany
{mehlhorn,ramezani}@mpi-inf.mpg.de

Abstract. R. Lavy and C. Swamy (FOCS 2005, J. ACM 2011) introduced a general method for obtaining truthful-in-expectation mechanisms from linear programming based approximation algorithms. Due to the use of the Ellipsoid method, a direct implementation of the method is unlikely to be efficient in practice. We propose to use the much simpler and usually faster multiplicative weights update method instead. The simplification comes at the cost of slightly weaker approximation and truthfulness guarantees.

1 Introduction

Algorithmic mechanism design studies optimization problems in which part of the input is not directly available to the algorithm; instead, this data is collected from self-interested players. It quests for *polynomial-time* algorithms that *(approximately) optimize* a global objective function (usually called *social welfare*), subject to the *strategic requirement* that the best strategy of the players is to *truthfully* report their part of the input. Such algorithms are called *truthful mechanisms*.

If the underlying optimization problem can be efficiently solved to optimality, the celebrated VCG mechanism (see, e.g., [17]) achieves truthfulness, social welfare optimization, and polynomial running time.

In general, the underlying optimization problem can only be solved approximately. Lavi and Swamy ([15,16]) showed that certain linear programming based approximation algorithms for the social welfare problem can be turned into randomized mechanisms that are truthful-in-expectation, i.e., reporting the truth maximizes the expected utility of an player. The LS-mechanism is powerful (see [4,11,15,16] for applications), but unlikely to be efficient in practice because of its use of the Ellipsoid method. *We show how to use the multiplicative weights update method instead. This results in simpler algorithms at the cost of somewhat weaker approximation and truthfulness guarantees.*

M. Hoefer (Ed.): SAGT 2015, LNCS 9347, pp. 98–109, 2015.
DOI: 10.1007/978-3-662-48433-3_8

We next review the LS-mechanism. It applies to integer linear programming problems of the packing type[1] for which the linear programming relaxation can be solved exactly and for which an α-integrality gap verifier is available (definition below). Let $\mathcal{Q} \subseteq \mathbb{R}^d_{\geq 0}$ be a *packing polytope*, i.e., $\mathcal{Q}$ is the intersection of finitely many halfspaces, and if $y \in \mathcal{Q}$ and $x \leq y$ then $x \in \mathcal{Q}$. We use $\mathcal{Q}_{\mathcal{I}} := \mathcal{Q} \cap \mathbb{Z}^d$ for the set of integral points in $\mathcal{Q}$, x^j for a typical element of $\mathcal{Q}_{\mathcal{I}}$, and $\mathcal{N}$ for the index set of all elements in $\mathcal{Q}_{\mathcal{I}}$. The mechanism consists of three main steps:

1. Let $v_i \in \mathbb{R}^d_{\geq 0}, 1 \leq i \leq n$, be the valuation of the i-th player and let $v = \sum_i v_i$ be the accumulated valuation. Solve the LP-relaxation, i.e., find a maximizer $x^* = \operatorname{argmax}_{x \in Q} v^T x$ for the social welfare of the fractional problem, and determine the VCG prices[2] $p_1, \ldots, p_n$. The allocation x^* and the VCG-prices are a truthful mechanism for the fractional problem.
2. Write $\alpha \cdot x^*$ as a *convex combination of integral solutions* in $\mathcal{Q}$, i.e., $\alpha \cdot x^* = \sum_{j \in \mathcal{N}} \lambda_j x^j$, $\lambda_j \geq 0$, $\sum_{j \in \mathcal{N}} \lambda_j = 1$, and $x^j \in \mathcal{Q}_{\mathcal{I}}$. This step requires an *α-integrality-gap-verifier* for $\mathcal{Q}_{\mathcal{I}}$ for some $\alpha \in [0, 1]$. On input $\bar{v} \in \mathbb{R}^d_{\geq 0}$ and $x^* \in \mathcal{Q}$, an α-integrality-gap-verifier returns an $x \in \mathcal{Q}_{\mathcal{I}}$ such that
$$\bar{v}^T x \geq \alpha \bar{v}^T x^*.$$
3. Pick the integral solution x^j with probability λ_j, and charge the i-th player the price $p_i \cdot (v_i^T x^j / v_i^T x^*)$ and If $v_i^T x^* = 0$, charge zero.

The LS-mechanism approximates social welfare with factor α and guarantees truthfulness-in-expectation, i.e., it converts a truthful fractional mechanism into an α-approximate truthful-in-expectation integral mechanism. With respect to practical applicability, steps 1 and 2 are the two major bottlenecks. Step 1 requires solving a linear program; an exact solution requires the use of the Ellipsoid method (see e.g. [10]), if the dimension is exponential. Furthermore, up to recently, the only method known to perform the decomposition in Step 2 is through the Ellipsoid method. An alternative method avoiding the use of the Ellipsoid method was recently given by Kraft, Fadaei, and Bichler [14]. We comment on their result in the next section.

1.1 Our Results

Our result concerns the design and analysis of a practical algorithm for the LS-scheme. We first consider the case where the LP-relaxation of SWM (social

[1] An example is the combinatorial auction problem. There is a set of m items to be sold to a set of n players. The (reported) value of a set S of items to the i-th player is $v_i(S)$ with $v_i(\emptyset) = 0$ and $v_i(S) \leq v_i(T)$ whenever $S \subseteq T$. Let $x_{i,S}$ be a 0–1 variable indicating that set S is given to player i. Then $\sum_S x_{i,S} \leq 1$ for every player i as at most one set can be given to i, and $\sum_i \sum_{S; j \in S} x_{i,S} \leq 1$ for every item j as any item can be given away only once. The social welfare is $\sum_{i,S} v_i(S) x_{i,S}$. The polytope $\mathcal{Q}$ is obtained by replacing the integrality constraints for $x_{i,s}$ by $0 \leq x_{i,S} \leq 1$. Note that the number d of variables is $n2^m$.

[2] $p_i = \sum_{j \neq i} v_j^T (\hat{x} - x^*)$, where $\hat{x} = \operatorname{argmax}_{x \in Q} \sum_{j \neq i} v_j^T x$.

welfare maximization) in Step 1 of the LS-scheme can be solved exactly and efficiently and then our problem reduces to the design of practical algorithm for the Step 2. In what follows we present an algorithm that is fast and practical for the convex decomposition (i.e., Step 2). Afterwards, we consider a more general problem where the LP-relaxation in Step 1 of the LS-scheme cannot be solved exactly and we only have an approximate solution that maximizes the LP-relaxation within a factor of $1 - \varepsilon$.

Convex Decomposition. Over the past 15 years, simple and fast methods [2,8,9,12,13,18,19] have been developed for solving packing and covering linear programs within an arbitrarily small error guarantee ε. These methods are based on the multiplicative weights update (MWU) method [1], in which a very simple update rule is repeatedly performed until a near-optimal solution is obtained. We show how to replace the use of the Ellipsoid method in Step 2 by an approximation algorithm for covering linear programs. This result is the topic of Sect. 2.

Theorem 1. *Let $\varepsilon > 0$ be arbitrary. Given a fractional point $x^* \in \mathcal{Q}$, and an α-integrality-gap verifier for $\mathcal{Q}_{\mathcal{I}}$, we can find a convex decomposition*

$$\frac{\alpha}{1+4\varepsilon} \cdot x^* = \sum_{j \in \mathcal{N}} \lambda_j x^j.$$

The convex decomposition has size (= number of nonzero λ_j) at most $s(1 + \lceil \varepsilon^{-2} \ln s \rceil)$, where s is the size of the support of x^* (= number of nonzero components). The algorithm makes at most $s\lceil \varepsilon^{-2} \ln s \rceil$ calls to the integrality-gap-verifier.

Kraft, Fadaei, and Bichler [14] obtained a related result independently. However, their construction is less efficient in two aspects. First, it requires $O(s^2\varepsilon^{-2})$ calls of the oracle. Second, the size of their convex decomposition might be as large as $O(s^3\varepsilon^{-2})$. In the combinatorial auction problem, $s = n + m$. Theorem 1 together with Steps 1 and 3 of the LS scheme implies a mechanism that is truthful-in-expectation and has $(\alpha/(1 + 4\varepsilon))$-social efficiency. A mechanism has γ-*social efficiency*, where $\gamma \in [0, 1]$, if the expected social welfare of the allocation returned by the mechanism is at least γ times the maximum possible social value.

Approximately Truthful-in-Expectation Mechanism. In contrast to Lavi-Swamy mechanism, let us assume that we do not want to solve the LP-relaxation exactly but instead, we want to use an ε-approximation algorithm $\mathcal{A}$ for it. Garg and Könemann [8] showed that there is an FPTAS for the packing problem and hence $\mathcal{A}$ exists, for every $\varepsilon > 0$. Using this, we show how to construct a fractional randomized mechanism for given $\varepsilon_0 \in (0, 1/2]$ and $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$ that satisfies:

1. *No positive transfer* (i.e., prices are non-negative).
2. *Individually rational with probability* $1-\varepsilon_0$ (i.e., the utility of any truth-telling player is non-negative with probability at least $1 - \varepsilon_0$).

3. $(1-\varepsilon_0)$-*truthful-in-expectation* (i.e., reporting the truth maximizes the expected utility of an player up to a factor $1-\varepsilon_0$)
4. $(1-\varepsilon)(1-\varepsilon_0)$-social efficiency.

Now, let us assume that x is a fractional allocation obtained from the above mechanism. We apply our convex decomposition technique and Step 3 of the Lavi-Swamy mechanism to obtain an integral randomized mechanism that satisfies the aforementioned conditions 1 to 3 and has $\alpha(1-\varepsilon)(1-\varepsilon_0)/(1+4\varepsilon)$-social efficiency. We show this result in Sect. 4.

Note that our fractional mechanism refines the one given in [5], where the dependency of ε on n and ε_0 is as $\varepsilon = \Theta(\varepsilon_0/n^9)$. A recent experimental study of our mechanism on Display Ad Auctions [6] shows the applicability of these methods in practice.

2 A Fast Algorithm for Convex Decompositions

Let $x^* \in \mathcal{Q}$ be arbitrary. Carr and Vempala [3] showed how to construct a convex combination of points in $\mathcal{Q}_\mathcal{I}$ dominating αx^* using a polynomial number of calls to an α-integrality-gap-verifier for $\mathcal{Q}_\mathcal{I}$. Lavi and Swamy [16] modified the construction to get an exact convex decomposition $\alpha x^* = \sum_{i\in\mathcal{N}} \lambda_i x^i$ for the case of packing linear programs. The construction uses the Ellipsoid method. We show an approximate version that replaces the use of the Ellipsoid method by the multiplicative weights update (MWU) method. For any $\varepsilon > 0$, we show how to obtain a convex decomposition of $\alpha x^*/(1+\varepsilon)$. Let s be the number of non-zero components of x^*. The size of the decomposition and the number of calls to the α-integrality gap verifier are $O(s\varepsilon^{-2}\ln s)$.

This section is structured as follows. We first review Kkandekar's FPTAS for covering linear programs (Subsect. 2.1). We then use it and the α-integrality gap verifier to construct a dominating convex combination for $\alpha x^*/(1+4\varepsilon)$, where $x^* \in \mathcal{Q}$ is arbitrary (Subsect. 2.2). In Subsect. 2.3, we show how to convert a dominating convex combination into an exact convex decomposition. Finally, in Subsect. 2.4, we put the pieces together.

2.1 An FPTAS for Covering Linear Programs

Consider a covering linear program:

$$\min c^T x \quad s.t. \quad \{Ax \geq b, \quad x \geq 0\} \tag{1}$$

where $A \in \mathbb{R}_{\geq 0}^{m\times n}$ is an $m \times d$ matrix with non-negative entries and $c \in \mathbb{R}_{\geq 0}^n$ and $b \in \mathbb{R}_{\geq 0}^m$ are non-negative vectors. We assume the availability of a κ-*approximation oracle* for some $\kappa \in (0,1]$.

$\mathcal{O}_\kappa(z)$: Given $z \in \mathbb{R}_{\geq 0}^m$, the oracle finds a column j of A that maximizes $\frac{1}{c_j}\sum_{i=1}^m \frac{z_i a_{ij}}{b_i}$ within a factor of κ:

$$\frac{1}{c_j}\sum_{i=1}^m \frac{z_i a_{ij}}{b_i} \geq \kappa \cdot \max_{j'\in[n]} \frac{1}{c_{j'}}\sum_{i=1}^m \frac{z_i a_{ij'}}{b_i} \tag{2}$$

Algorithm 1. Covering($\mathcal{O}_\kappa$)

Require: a covering system (A, b, c) given by a $\kappa-$approximation oracle $\mathcal{O}_\kappa$, where $A \in \mathbb{R}^{m\times n}_{\geq 0}$, $b \in \mathbb{R}^m_{>0}$, $c \in \mathbb{R}^n_{>0}$, and an accuracy parameter $\varepsilon \in (0, 1/2]$

Ensure: A feasible solution $\hat{x} \in \mathbb{R}^n_{\geq 0}$ to (1) s.t. $c^T\hat{x} \leq \frac{(1+4\varepsilon)}{\kappa} z^*$

1: $x(0) := 0$; $t := 0$; and $T := \frac{\ln m}{\varepsilon^2}$
2: **while** $M(t) < T$ **do**
3: $\quad t := t + 1$
4: $\quad$ Let $j(t) := \mathcal{O}_\kappa(p(t)/\|p(t)\|_1)$
5: $\quad x_{j(t)}(t) := x_{j(t)}(t-1) + \delta(t)$ and $x_j(t) = x_j(t-1)$ for $j \neq j(t)$
6: **end while**
7: **return** $\hat{x} = \frac{x(t)}{M(t)}$

For an exact oracle $\kappa = 1$, Khandekar [12] gave an algorithm which computes a feasible solution $\hat{x}$ to covering LP (1) such that $c^T\hat{x} \leq (1 + 4\varepsilon)z^*$ where z^* is the value of an optimal solution. The algorithm makes $O(m\varepsilon^{-2} \log m)$ calls to the oracle, where m is the number of rows in A. If the exact oracle in Khandekar's algorithm is replaced by a κ-approximation algorithm, it computes a feasible solution $\hat{x} \in \mathbb{R}^n_{\geq 0}$ to (1) such that $c^T\hat{x} \leq (1 + 4\varepsilon)z^*/\kappa$. The algorithm is given as Algorithm 1 and can be thought of as the algorithmic dual of the FPTAS for multicommodity flows given in [8]. We use A_i to denote the i-th row of A. The algorithm constructs vectors $x(t) \in \mathbb{R}^n_{\geq 0}$, for $t = 0, 1, \ldots,$ until $M(t) := \min_{i\in[m]} A_i x(t)/b_i$ becomes at least $T := \frac{\ln m}{\varepsilon^2}$. Define the *active list* at time t by $L(t) := \{i \in [m] : A_i x(t-1)/b_i < T\}$. For $i \in L(t)$, define

$$p_i(t) := (1 - \varepsilon)^{A_i x(t-1)/b_i}, \tag{3}$$

and set $p_i(t) = 0$ for $i \notin L(t)$. At each time t, the algorithm calls the oracle with the vector $z_t = p(t)/\|p(t)\|_1$, and increases the variable $x_{j(t)}$ by

$$\delta(t) := \min_{i\in L(t) \text{ and } a_{i,j(t)}\neq 0} \frac{b_i}{a_{i,j(t)}}, \tag{4}$$

where $j(t)$ is the index returned by the oracle. Due to lack of space, the proof of following theorem and corollary are presented in the full paper [7].

Theorem 2. *Let $\varepsilon \in (0, \frac{1}{2}]$ and let z^* be the value of an optimum solution to (1). Procedure Covering($\mathcal{O}_\kappa$) (see Algorithm 1) terminates in at most $m\lceil\varepsilon^{-2} \ln m\rceil$ iterations with a feasible solution $\hat{x}$ of (1) of at most $m\lceil\varepsilon^{-2} \ln m\rceil$ positive components. At termination, it holds that*

$$c^T\hat{x} \leq \frac{(1 + 4\varepsilon)}{\kappa} z^*. \tag{5}$$

We observe that the proof of Theorem 2 can be modified to give:

Corollary 1. *Suppose $b = \mathbf{1}$, $c = \mathbf{1}$, and we use the following oracle $\mathcal{O}'$ instead of $\mathcal{O}$ in Algorithm 1:*

$\mathcal{O}'(A, z)$: *Given* $z \in \mathbb{R}^m_{\geq 0}$, *such that* $\mathbf{1}^T z = 1$, *the oracle finds a column* j *of* A *such that* $z^T A \mathbf{1}_j \geq 1$.

Then the algorithm terminates in at most $m\lceil \varepsilon^{-2} \ln m \rceil$ *iterations with a feasible solution* $\hat{x}$ *having at most* $m\lceil \varepsilon^{-2} \ln m \rceil$ *positive entries, such that* $\mathbf{1}^T \hat{x} \leq 1 + 4\varepsilon$.

2.2 Finding a Dominating Convex Combination

Recall that we use $\mathcal{N}$ to index the elements in $\mathcal{Q}_{\mathcal{I}}$. We assume the availability of an α-integrality-gap-verifier $\mathcal{F}$ for $\mathcal{Q}_{\mathcal{I}}$. We will use the results of the preceding section and show how to obtain for any $x^* \in \mathcal{Q}$ and any positive ε a convex composition of points in $\mathcal{Q}_{\mathcal{I}}$ that covers $\alpha x^*/(1 + 4\varepsilon)$. Our algorithm requires $O(s\varepsilon^{-2} \ln s)$ calls to the oracle, where s is the support of x^*.

Theorem 3. *Let* $\varepsilon > 0$ *be arbitrary. Given a fractional point* $x^* \in \mathcal{Q}$ *and an* α*-integrality-gap verifier* $\mathcal{F}$ *for* $\mathcal{Q}_{\mathcal{I}}$, *we can find a convex combination* $\bar{x}$ *of integral points in* $\mathcal{Q}_{\mathcal{I}}$ *such that*

$$\frac{\alpha}{1 + 4\varepsilon} \cdot x^* \leq \bar{x} = \sum_{i \in \mathcal{N}} \lambda_i x^i.$$

The convex decomposition has size at most $s\lceil \varepsilon^{-2} \ln s \rceil$, *where* s *is the number of positive entries of* x^*. *The algorithm makes at most* $s\lceil \varepsilon^{-2} \ln s \rceil$ *calls to the integrality-gap verifier.*

Proof. The task of finding the multipliers λ_i is naturally formulated as a covering LP, namely,

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{N}} \lambda_i \\ s.t. \quad & \sum_{i \in \mathcal{N}} \lambda_i x^i_j \geq \alpha \cdot x^*_j \quad \text{for all } j, \\ & \sum_{i \in \mathcal{N}} \lambda_i \geq 1, \quad \lambda_i \geq 0. \\ & \lambda_i \geq 0. \end{aligned} \tag{6}$$

Clearly, we can restrict our attention to the $j \in S^+ := \{j : x^*_j > 0\}$ and rewrite the constraint for $j \in S^+$ as $\sum_{i \in \mathcal{N}} \lambda_i x^i_j / (\alpha \cdot x^*_j) \geq 1$. For simplicity of notation, we assume $S^+ = [1..s]$. In the language of the preceding section, we have $m = s + 1$, $n = |\mathcal{N}|$, $c = \mathbf{1}$, $b = \mathbf{1}$ and the variable $x = \lambda$. The matrix $A = (a_{j,i})$ is as follows (note that we use j for the row index and i for the column index):

$$a_{j,i} := \begin{cases} x^i_j/(\alpha x^*_j) & 1 \leq j \leq s, i \in \mathcal{N} \\ 1 & j = s + 1, i \in \mathcal{N} \end{cases}$$

Thus we can apply Corollary 1 of Sect. 2.1, provided we can efficiently implement the required oracle $\mathcal{O}'$. We do so using $\mathcal{F}$.

Oracle $\mathcal{O}'$ has arguments $(A, \tilde{z})$ such that $1^T\tilde{z} = 1$. Let us conveniently write $\tilde{z} = (w, z)$, where $w \in \mathbb{R}^s_{\geq 0}$, $z \in \mathbb{R}_{\geq 0}$, and $\sum_{j=s}^{j=1} w_j + z = 1$. Oracle $\mathcal{O}'$ needs to find a column i such that $\tilde{z}^T A\mathbf{1}_i \geq 1$. In our case $\tilde{z}^T A\mathbf{1}_i = \sum_{j=1}^{s} w_j x^i_j/\alpha x^*_j + z$, and we need to find a column i for which this expression is at least one. Since z does not depend on i, we concentrate on the first term. Define

$$V_j := \begin{cases} \frac{w_j}{\alpha x^*_j} & \text{for } j \in S^+ \\ 0 & \text{otherwise.} \end{cases}$$

Call algorithm $\mathcal{F}$ with $x^* \in \mathcal{Q}$ and $V := (V_1, \ldots, V_d)$. $\mathcal{F}$ returns an integer solution $x^i \in \mathcal{Q}_{\mathcal{I}}$ such that

$$V^T x^i = \sum_{j \in S^+} \frac{w_j}{\alpha x^*_j} x^i_j \geq \alpha \cdot V^T x^* = \sum_{j \in S^+} w_j,$$

and hence,

$$\sum_{j \in S^+} \frac{w_j}{\alpha x^*_j} x^i_j + z \geq \sum_{j \in S^+} w_j + z = 1.$$

Thus i is the desired column of A.

It follows by Corollary 1 that Algorithm 1 finds a feasible solution $\lambda' \in \mathbb{R}^{|\mathcal{N}|}_{\geq 0}$ to the covering LP (6), and a set $\mathcal{Q}'_{\mathcal{I}} \subseteq \mathcal{Q}_{\mathcal{I}}$ of vectors (returned by $\mathcal{F}$), such that $\lambda'_i > 0$ only for $i \in \mathcal{N}'$, where $\mathcal{N}'$ is the index set returned by oracle $\mathcal{O}'$ and $|\mathcal{N}'| \leq s\lceil \varepsilon^{-2} \ln s \rceil$ also $\Lambda := \sum_{i \in \mathcal{N}'} \lambda'_i \leq (1 + 4\varepsilon)$. Scaling λ'_i by Λ, we obtain a set of multipliers $\{\lambda_i = \lambda'_i/\Lambda : \ i \in \mathcal{N}'\}$, such that $\sum_{i \in \mathcal{N}'} \lambda_i = 1$ and

$$\sum_{i \in \mathcal{N}'} \lambda_i x^i \geq \frac{\alpha}{1 + 4\varepsilon} x^*. \tag{7}$$

We may assume $x^i_j = 0$ for all $j \notin S^+$ whenever $\lambda_i > 0$; otherwise simply replace x^i by a vector in which all components not in S^+ are set to zero, by using packing property this is possible. □

2.3 From Dominating Convex Combination to Exact Convex Decomposition

We will show how to turn a dominating convex combination into an exact decomposition. The construction is general and uses only the packing property. Such a construction seems to have been observed in [15], but was not made explicit. Kraft, Fadaei, and Bichler [14] describe an alternative construction. Their construction may increase the size of the convex decomposition (= number of non-zero λ_i) by a multiplicative factor s and an additive factor s^2. In contrast, our construction increases the size only by an additive factor s.

Theorem 4. *Let $x^* \in \mathcal{Q}$ be dominated by a convex combination $\sum_{i \in \mathcal{N}} \lambda_i x^i$ of integral points in $\mathcal{Q}_{\mathcal{I}}$, i.e.,*

$$\sum_{i \in \mathcal{N}} \lambda_i x^i \geq x^*. \tag{8}$$

Algorithm 2. Changing a dominating convex decomposition into an exact decomposition

Require: A packing convex set $\mathcal{Q}$ and point $x^* \in \mathcal{Q}$ and a convex combination $\sum_{i\in\mathcal{N}} \lambda_i x^i$ of integral points in $\mathcal{Q}_\mathcal{I}$ dominating x^*.
Ensure: A convex decomposition $x^* = \sum_{i\in\mathcal{N}'} \lambda_i x^i$ with $x^i \in \mathcal{Q}_\mathcal{I}$.
1: **while** there is an $i \in \mathcal{N}$ and a j such that $\lambda_i x^i_j > 0$ and $\sum_{h\in\mathcal{N}} \lambda_h x^h - \lambda_i \mathbf{1}_j \geq x^*$ **do**
2: replace x^i by $x^i - \mathbf{1}_j$.
3: **end while**
4: **while** $\Delta_j := \sum_{i\in\mathcal{N}} \lambda_i x^i - x^*_j > 0$ for some j **do**
5: {for all $i \in \mathcal{N}$ and all j: if $\lambda_i x^i_j > 0$ then $\sum_{h\in\mathcal{N}} \lambda_h x^h - \lambda_i \mathbf{1}_j < x^*$}
6: Let j be such that $\Delta_j > 0$ and let i be such that $\lambda_i x^i_j > 0$. Let $B = \{j \in S^+ : x^i_j \neq 0 \text{and} \Delta_j > 0\}$ and let $b = |B|$. Renumber the coordinates such that $B = \{1, \ldots, b\}$ and $\Delta_1/x^i_1 \leq \ldots \leq \Delta_b/x^i_b$.
7: For $\ell \in \{0, \ldots, b\}$ define a vector y^ℓ by $y^\ell_j = x^i_j$ for $j \leq \ell$ and $y^\ell_j = 0$ for $j > \ell$.
8: Change the left-hand side of (8) as follows: replace λ_i by $\lambda_i - \Delta_b/x^i_b$; for $1 \leq \ell < b$, increase the coefficient of y^ℓ by $\Delta_{\ell+1}/x^i_{\ell+1} - \Delta_\ell/x^i_\ell$; and increase the coefficient of y^0 by Δ_1/x^i_1.
9: **end while**

Then Algorithm 2 achieves equality in (8). It increases the size of the convex combination by at most s, where s is the number of positive components of x^*.

Proof. Let $\mathbf{1}_j$ be the j-th unit vector. As long as there is an $i \in \mathcal{N}$ and a j such that $\lambda_i x^i_j > 0$ and replacing x^i by $x^i - \mathbf{1}_j$ maintains feasibility, i.e., satisfies constraint (8), we perform this replacement. Note that x^i is an integer vector in $\mathcal{Q}_\mathcal{I}$, therefore $x^i - \mathbf{1}_j$ remains positive vector and with using packing property, it is also in $\mathcal{Q}_\mathcal{I}$. We may therefore assume that the set of vectors indexed by $\mathcal{N}$ satisfy a minimality condition which is for all $i \in \mathcal{N}$ and $j \in S^+$ with $\lambda_i x^i_j > 0$

$$\sum_{h\in\mathcal{N}} \lambda_h x^h_j - \lambda_i \mathbf{1}_j < x^*_j \tag{9}$$

We will establish (9) as an invariant of the second while-loop.

For $j \in S^+$, let $\Delta_j = \sum_{i\in\mathcal{N}} \lambda_i x^i_j - x^*_j$. Then $\Delta_j \geq 0$ and, by (9), for every $j \in S^+$ and $i \in \mathcal{N}$, with $\lambda_i \neq 0$ either $x^i_j = 0$ or $\Delta_j < \lambda_i \leq \lambda_i x^i_j$. If $\Delta_j = 0$ for all $j \in S^+$, we are done. Otherwise, choose j and $i \in \mathcal{N}$ such that $\Delta_j > 0$ and $x^i_j > 0$. Let $B = \{j \in S^+ : x^i_j \neq 0 \text{ and } \Delta_j > 0\}$ be the indices in the support of x^i for which Δ_j is non-zero. We will change the left-hand side of (8) such that equality holds for all indices in B. The change will not destroy an already existing equality for an index outside B and hence the number of indices for which equality holds increases by $|B|$.

Let $b = |B|$. By renumbering the coordinates, we may assume $B = \{1, \ldots, b\}$ and $\Delta_1/x^i_1 \leq \ldots \leq \Delta_b/x^i_b$. For $j \in [b]$, we clearly have

$$\lambda_i - \frac{\Delta_j}{x^i_j} = \lambda_i - \frac{\Delta_b}{x^i_b} + \frac{\Delta_b}{x^i_b} - \frac{\Delta_{b-1}}{x^i_{b-1}} + \cdots + \frac{\Delta_{j+1}}{x^i_{j+1}} - \frac{\Delta_j}{x^i_j}.$$

Multiplying by x^i_j and adding zero a few times, we obtain

$$\lambda_i x^i_j - \Delta_j = \left(\lambda_i - \frac{\Delta_b}{x^i_b}\right) x^i_j + \sum_{\ell=j}^{b-1} \left(\frac{\Delta_{\ell+1}}{x^i_{\ell+1}} - \frac{\Delta_\ell}{x^i_\ell}\right) x^i_j + \sum_{\ell=1}^{j-1} \left(\frac{\Delta_{\ell+1}}{x^i_{\ell+1}} - \frac{\Delta_\ell}{x^i_\ell}\right) 0 + \frac{\Delta_1}{x^i_1} 0.$$

For $\ell \in \{0, \ldots, b-1\}$ define a vector y^ℓ by $y^\ell_j = x^i_j$ for $j \le \ell$ and $y^\ell_j = 0$ for $j > \ell$. Then $x^i_j = y^\ell_j$ for $\ell \ge j$ and $0 = y^\ell_j$ for $\ell < j$. Hence for all $j \le b$

$$\lambda_i x^i_j - \Delta_j = \left(\lambda_i - \frac{\Delta_b}{x^i_b}\right) x^i_j + \sum_{\ell=1}^{b-1} \left(\frac{\Delta_{\ell+1}}{x^i_{\ell+1}} - \frac{\Delta_\ell}{x^i_\ell}\right) y^\ell_j + \frac{\Delta_1}{x^i_1} y^0_j. \tag{10}$$

Note that the coefficients on the right-hand side of (10) are non-negative and sum up to λ_i. Also, by the packing property of $\mathcal{Q}$, $y^\ell \in \mathcal{Q}_\mathcal{I}$ for $0 \le \ell < b$. We now change the left-hand side of (8) as follows: we replace λ_i by $\lambda_i - \Delta_b/x^i_b$; for $1 \le \ell < b$, we increase the coefficient of y^ℓ by $\Delta_{\ell+1}/x^i_{\ell+1} - \Delta_\ell/x^i_\ell$; and we increase the coefficient of y^0 by Δ_1/x^i_1. As a result, we now have equality for all indices in B. The Δ_j for $j \notin B$ are not affected by this change.

We still need to establish that (9) holds for the vectors y^ℓ, $0 \le \ell < b$, that have a non-zero coefficient in the convex combination. Note first that $y^\ell_j > 0$ implies $j \in B$. Also (8) holds with equality for all $j \in B$. Thus (9) holds.

Consider any iteration of the second while-loop. It adds up to b vectors to the convex decomposition and decreases the number of nonzero Δ's by b. Thus the total number of vectors added to the convex decomposition is at most s. □

2.4 Fast Convex Decomposition

Proof (of Theorem 1). Theorem 3 yields a convex combination of integer points of $\mathcal{Q}_I$ dominating $\alpha x^*/1 + 4\varepsilon$. Theorem 4 turns this dominating convex combination into an exact combination. It adds up to $|S|$ additional vectors to the convex combination. The complexity bounds follow directly from the referenced theorems. □

3 Approximately Truthful-in-Expectation Fractional Mechanisms

In this section we assume that we do not want to solve the LP-relaxation of SWM exactly and there is an FPTAS for it. Then by using the FPTAS, we construct a randomized fractional mechanism, Algorithm 3, and state the following theorem. For the proof of the theorem and FPTAS algorithm see the full paper [7].

Theorem 5. *Let $\varepsilon_0 \in (0, 1/2]$, $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$ and $\gamma = (1-\varepsilon)(1-\varepsilon_0)$. Given an ε-approximation algorithm for $\mathcal{Q}$, Algorithm 3 defines a fractional randomized mechanism with the following conditions:*

No positive transfer. (11)

Individually rational with probability $1 - \varepsilon_0$. (12)

$(1-\varepsilon_0)$*-truthful-in-expectation.* (13)

γ*-social efficiency, where* γ *depends on* ε_0, *and* ε. (14)

In order to present Algorithm 3, we make some assumption and define some notation. Let us assume that the problem is *separable* that means the variables can be partitioned into disjoint groups, one for each player, such that the value of an allocation for a player depends only on the variables in his group, i.e.,

$$v_i(x) = v_i(x_i),$$

where x_i is the set of variables associated with player i. Formally, any outcome $x \in \mathcal{Q} \subseteq \mathbb{R}^d$ can be written as $x = (x_1, \ldots, x_n)$ where $x_i \in \mathbb{R}^{d_i}$ and $d = d_1 + \ldots + d_n$.[3] We further assume that for each player $i \in [n]$, there is an optimal allocation $u^i \in \mathcal{Q}$ that maximizes his value for every valuation v_i, i.e.,

$$v_i(u^i) = \max_{z \in \mathcal{Q}} v_i(z), \tag{15}$$

for every $v_i \in \mathcal{V}_i$, where $\mathcal{V}_i$ denote the all possible valuations of player i. In combinatorial auction, the allocation u^i allocates all the items to player i. Let

$$L_i := \sum_{j \neq i} v_j(u^j) \quad \text{and} \quad \beta_i := \varepsilon L_i. \tag{16}$$

Note that L_i does not depend on the valuation of player i. Let $\mathcal{A}$ be an FPTAS for LP relaxation of SWM. We use $\mathcal{A}(v, \varepsilon)$ to denote the outcome of $\mathcal{A}$ on input v and ε. If ε is understood, we simply write $\mathcal{A}(v)$; $\mathcal{A}(v)$ is a fractional allocation in Q. In the following, we will apply $\mathcal{A}$ to different valuations which we denote by $v = (v_i, v_{-i})$, $\bar{v} = (\bar{v}_i, v_{-i})$, and $v' = (\mathbf{0}, v_{-i})$. Here v_i is the reported valuation of player i, $\bar{v}_i$ is his true valuation and $v'_i = \mathbf{0}$. We denote the allocation returned by $\mathcal{A}$ on input v (resp., $\bar{v}$, v') by x (resp., $\bar{x}$, x') and use the payment rule:

$$p_i(v) := \max\{p_i^{VCG}(v) - \beta_i, 0\} \tag{17}$$

where

$$p_i^{VCG}(v) := v_{-i}(x') - v_{-i}(x).$$

$v_{-i}(x) = \sum_{j \neq i} v_j(x)$, $x = \mathcal{A}(v)$ and $x' = \mathcal{A}(0, v_{-i})$. Observe the similarity in the definition of $p_i^{VCG}(v)$ to the VCG payment rule. In both cases, the payment is defined as the difference of the total value of two allocations to the players different from i. The first allocation ignores the influence of player i ($x' = \mathcal{A}(0, v_{-i})$) and the second allocation takes it into account ($x = \mathcal{A}(v)$). Define $q_0 = (1 - \frac{\varepsilon_0}{n})^n$, $\bar{\varepsilon} = \varepsilon_0/2$, and $q_j = (1 - q_0)/n$ for $1 \leq j \leq n$. Let $\eta = \bar{\varepsilon}(1 - q_0)^2/n^3$, $\eta' = \eta/q_j$, and $\varepsilon = \eta\bar{\varepsilon}(1 - q_0)/(8n)$. Let $U_i(v)$ be the utility of player i obtained by the mechanism which has an allocation function $\mathcal{A}$ and payment rule (17). Following [5], we call player i *active* if the following two conditions hold:

$$U_i(v) + \frac{\bar{\varepsilon} q_i}{q_0} v_i(u^i) \geq \frac{q_i}{q_0} \eta' L_i, \tag{18}$$

$$v_i(u^i) \geq \eta L_i. \tag{19}$$

[3] In the combinatorial auction problem, variable x_i comprises all variables $x_{i,S}$ and the value of an allocation for player i depends only on the variables $x_{i,S}$.

Algorithm 3. The mechanism M of Theorem 5. The vectors u^i are defined as in (15) and the quantities L_i are defined in (16). The definitions of q_0, q_j, active and inactive player are given in the proof of Theorem 5.

Require: A valuation vector v, a packing convex set $\mathcal{Q}$ and an approximation scheme $\mathcal{A}$.

Ensure: An allocation $x \in \mathcal{Q}$ and a payment $p \in \mathbb{R}^n$

1: Let ε be defined as in the below.
2: Choose an index $j \in \{0, 1, \ldots, n\}$, where 0 is chosen with probability q_0 and $j \in \{1, \ldots, n\}$ is chosen with probability $q_j = (1 - q_0)/n$.
3: **if** $j = 0$ **then**
4: Use ε-approximation algorithm $\mathcal{A}$ to compute an allocation $x = (x_1, \ldots, x_n) \in \mathcal{Q}$ and compute payments with payment rule (17). For all inactive i, change x_i and p_i to zero.
5: **else**
6: For every $1 \leq i \leq n$, set

$$\begin{cases} x_i = u^i, p_i = \eta' L_i & \text{if } i = j \text{ and } i \text{ is active,} \\ x_i = u^i, p_i = 0 & \text{if } i = j \text{ and } i \text{ is inactive,} \\ x_i = 0, p_i = 0 & \text{if } i \neq j. \end{cases}$$

7: **end if**
8: **return** (x, p)

Now, we briefly explain Algorithm 3. Let us choose a random number $j \in \{0, 1, \ldots, n\}$ with probability q_j. If $j = 0$, we run ε-approximation algorithm $\mathcal{A}$ on v to compute allocation $x = (x_1, \ldots, x_n)$. Then we change x_i and p_i to zero for all inactive i. And if $j \neq 0$, we give optimal set u^j to j-th player and charged him with a price $\eta' L_j$ if he is active and zero otherwise. For all other players, we do not assign any item to them and do not charge them any price.

4 Approximately Truthful-in-Expectation Integral Mechanisms

In this subsection we obtain a randomized mechanism M' which returns an integral allocation. Let $\varepsilon > 0$ be arbitrary. First run Algorithm 3 to obtain x and $p(v)$. Then compute a convex decomposition of $\frac{\alpha}{1+4\varepsilon}x$, which is $\frac{\alpha}{1+4\varepsilon}x = \sum_{j \in \mathcal{N}} \lambda_j^x x^j$. Finally with probability λ_j^x (we used superscript to distinguish the convex decompositions of x) return the allocation x^j and charge i-th player, the price $p_i(v)\frac{v_i(x^j)}{v_i(x)}$, if $v_i(x) > 0$, and zero otherwise. In the following theorem we show mechanism M' is indeed an approximately truthful-in-expectation integral mechanism whose proof is appeared in the full paper [7].

Theorem 6. *Suppose that $\varepsilon_0 \in (0, 1/2]$ be any constant, $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$ and $\gamma = \alpha(1 - \varepsilon)(1 - \varepsilon_0)/(1 + 4\varepsilon)$. Then we obtain a randomized integral mechanism satisfying Conditions (11) to (14).*

References

1. Arora, S., Hazan, E., Kale, S.: Multiplicative weights method: a meta-algorithm and its applications. Technical report, Princeton University, USA (2006)
2. Bienstock, D., Iyengar, G.: Approximating fractional packings and coverings in O(1/epsilon) iterations. SIAM J. Comput. **35**(4), 825–854 (2006)
3. Carr, R.D., Vempala, S.: Randomized metarounding. Random Struct. Algorithms **20**(3), 343–352 (2002)
4. Christodoulou, G., Elbassioni, K., Fouz, M.: Truthful mechanisms for exhibitions. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 170–181. Springer, Heidelberg (2010)
5. Dughmi, S., Roughgarden, T., Vondrák, J., Yan, Q.: An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries. CoRR abs/1109.1053 (2011)
6. Elbassioni, K., Jha, M.: On the power of combinatorial bidding in web display ads. In: ICIW (2015, to appear)
7. Elbassioni, K., Mehlhorn, K., Ramezani, F.: Towards more practical linear programming-based techniques for algorithmic mechanism design. In: Arxive (2015)
8. Garg, N., Könemann, J.: Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In: FOCS, pp. 300–309 (1998)
9. Grigoriadis, M.D., Khachiyan, L.G.: A sublinear-time randomized approximation algorithm for matrix games. Oper. Res. Lett. **18**(2), 53–58 (1995)
10. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, New York (1988)
11. Hoefer, M., Kesselheim, T., Vöcking, B.: Approximation algorithms for secondary spectrum auctions. In: SPAA, pp. 177–186 (2011)
12. Khandekar, R.: Lagrangian Relaxation Based Algorithms for convex Programming Problems. PhD thesis, Indian Institute of Technology, Delhi, India (2004)
13. Koufogiannakis, C., Young, N.E.: Beating simplex for fractional packing and covering linear programs. In: FOCS, pp. 494–504 (2007)
14. Kraft, D., Fadaei, S., Bichler, M.: Fast convex decomposition for truthful social welfare approximation. In: Liu, T.-Y., Qi, Q., Ye, Y. (eds.) WINE 2014. LNCS, vol. 8877, pp. 120–132. Springer, Heidelberg (2014)
15. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. In: FOCS, pp. 595–604 (2005)
16. Lavi, R., Swamy, C.: Truthful and near-optimal mechanism design via linear programming. J. ACM **58**(6), 25 (2011)
17. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
18. Plotkin, S.A., Shmoys, D.B., Tardos, É.: Fast approximation algorithms for fractional packing and covering problems. In: FOCS, pp. 495–504 (1991)
19. Young, N.E.: Sequential and parallel algorithms for mixed packing and covering. In: FOCS, pp. 538–546 (2001)