Application of Multiplicative Weights Update Method in Algorithmic Game Theory

Dissertation

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von

Fahimeh Ramezani

Saarbrücken
2015

Tag des Kolloquiums:
17. August. 2015

Dekan der Naturwissenschaftlich-Technischen Fakultät I:
Prof. Dr. Markus Bläser

Prüfungsausschuss:

Prof. Dr. Holger Hermanns (Vorsitzender des Prüfungsausschusses)

Prof. Dr. Dr. h.c. mult. Kurt Mehlhorn (Gutachter)

Dr. Khaled Elbassioni (Gutachter)

Dr. Parinya Chalermsook (Akademischer Beisitzer)

Abstract

In this thesis, we apply the Multiplicative Weights Update Method (MWUM) to the design of approximation algorithms for some optimization problems in game-theoretic settings.

Lavi and Swamy [LS05, LS11] introduced a randomized mechanism for combinatorial auctions that uses an approximation algorithm for the underlying optimization problem, so-called social welfare maximization and converts the approximation algorithm to a randomized mechanism that is truthful-in-expectation, which means each player maximizes its expected utility by telling the truth. The mechanism is powerful (e.g., see [LS05, LS11, CEF10, HKV11] for applications), but unlikely to be efficient in practice, because it uses the Ellipsoid method. In Chapter 2, we follow the general scheme suggested by Lavi and Swamy and replace the Ellipsoid method with MWUM. This results in a faster and simpler approximately truthful-in-expectation mechanism. We also extend their assumption regarding the existence of an exact solution for the LP-relaxation of social welfare maximization. We assume that there exists an approximation algorithm for the LP and establish a new randomized approximation mechanism.

In Chapter 3, we consider the problem of computing an approximate saddle point, or equivalently equilibrium, for a convex-concave functions $F : X \times Y \to \mathbb{R}$, where X and Y are convex sets of arbitrary dimensions. Our main contribution is the design of a randomized algorithm for computing an ε-approximation saddle point for F. Our algorithm is based on combining a technique developed by Grigoriadis and Khachiyan [GK95], which is a randomized variant of Brown's fictitious play [Bro51], with the recent results on random sampling from convex sets (see, e.g., [LV06, Vem05]). The algorithm finds an ε-approximation saddle point in an expected number of $O\left(\frac{\rho^2(n+m)}{\varepsilon^2}\log\frac{R}{\varepsilon}\right)$ iterations, where in each iteration two points are sampled from log-concave distributions over strategy sets. It is assumed that X and Y have inscribed balls of radius $1/R$ and circumscribing balls of radius R and $\rho = \max_{x\in X, y\in Y} |F(x,y)|$. In particular, the algorithm requires $O^*\left(\frac{\rho^2(n+m)^6}{\varepsilon^2}\log R\right)$ calls to a membership oracle, where $O^*(\cdot)$ suppresses polylogarithmic factors that depend on n, m, and ε.

Zusammenfassung

In dieser Doktorarbeit verwenden wir die Multiplicative Weights Update Method (MWUM) für den Entwurf von Approximationsalgorithmen für bestimmte Optimierungsprobleme im spieltheoretischen Umfeld.

Lavi und Swamy [LS05, LS11] präsentierten einen randomisierten Mechanismus für kombinatorische Auktionen. Sie verwenden dazu einen Approximationsalgorithmus für die Lösung des zugrundeliegenden Optimierungsproblem, das so genannte Social Welfare Maximization Problem, und wandeln diesen zu einem randomisierten Mechanismus um, der im Erwartungsfall anreizkompatibel ist. Dies bedeutet jeder Spieler erreicht den maximalen Gewinn, wenn er sich ehrlich verhält. Der Mechanismus ist sehr mächtig (siehe [LS05, LS11, CEF10, HKV11] für Anwendungen); trotzdem ist es unwahrscheinlich, dass er in der Praxis effizient ist, da hier die Ellipsoidmethode verwendet wird. In Kapitel 2 folgen wir dem von Lavi und Swamy vorgeschlagenem Schema und ersetzen die Ellipsoidmethode durch MWUM. Das Ergebnis ist ein schnellerer, einfacherer und im Erwartungsfall anreizkompatibler Approximationsmechanismus. Wir erweitern ihre Annahme zur Existenz einer exakten Lösung der LP-Relaxierung für das Social Welfare Maximization Problem. Wir nehmen an, dass ein Approximationsalgorithmus für das LP existiert und beschreiben darauf basierend einen neuen randomisierten Approximationsmechanismus.

In Kapitel 3 betrachten wir das Problem für konvexe und konkave Funktionen $F : X \times Y \to \mathbb{R}$, wobei X und Y konvexe Mengen von beliebiger Dimension sind, einen Sattelpunkt zu approximieren (oder gleichbedeutend ein Equilibrium). Unser Hauptbeitrag ist der Entwurf eines randomisierten Algorithmus zur Berechnung einer ϵ-Näherung eines Sattelpunktes von F. Unser Algorithmus beruht auf der Kombination einer Technik entwickelt durch Grigoriadis und Khachiyan [GK95], welche eine zufallsbasierte Variation von Browns Fictitious Play [Bro51] ist, mit kürzlich erschienenen Resultaten im Bereich der zufälligen Stichprobennahme aus konvexen Mengen (siehe [LV06, Vem05]). Der Algorithmus findet eine ϵ-Näherung eines Sattelpunktes im Erwartungsfall in $O(\frac{\rho^2(n+m)^6}{\epsilon^2} \log \frac{R}{\epsilon})$ Rechenschritten, wobei in jedem Rechenschritt zwei Punkte zufällig gemäß einer log-konkaven Verteilungen über Strategiemengen gezogen werden. Hier nehmen wir an, dass X und Y einbeschriebene Kugeln mit Radius $1/R$ und umschreibende Kugeln von Radius R besitzen und $\rho = \max_{x \in X, y \in Y} |F(x, y)|$. Der Algorithmus benötigt dabei $O^*(\frac{\rho^2(n+m)^6}{\epsilon^2} \log R)$ Aufrufe eines Zugehörigkeitsorakels, hier versteckt $O^*(\cdot)$ polylogarithmische Faktoren, die von n, m und ϵ abhängen.

Acknowledgments

I would like to thank my supervisors Kurt Mehlhorn and Khaled Elbassioni for their continuous support. Special thanks to Kurt for his guidance and providing me an opportunity to work in his group. I would like to express my appreciation to Khaled who introduced me to the field of Optimization and for helpful discussions during my PhD.

I am sincerely grateful to my family for their great support and encouragement. Thanks, Ali, for your help, care and scientific discussion during my PhD and grateful to my little son, Mohammad Mahdi for his love.

Contents

1. Introduction **9**
1.1. Algorithmic Mechanism Design 10
1.2. Approximating Saddle Points 12

2. Towards More Practical Algorithmic Mechanism Design **15**
2.1. Notations, Definitions, and Preliminaries 18
2.2. The Lavi-Swamy Mechanism 21
2.3. The Fast Algorithm for Convex Decompositions 25
2.4. Approximately Truthful-in-Expectation Mechanisms 34

3. Randomized Fictitious Play for Approximating Saddle Points **45**
3.1. Definitions, Notations, and Preliminaries 47
3.2. Relation to Previous Works 50
3.3. Main Results 53
3.4. The Analysis of Algorithm 5 55
3.5. Applications in Combinatorial Optimization 63

Bibliography **67**

A. Multiplicative Weights Update Algorithms **71**
A.1. The Weighted Majority Algorithm 72
A.2. The Multiplicative Weights Update Algorithm 73
A.3. The Approximation Algorithm for Feasibility Check 74
A.4. The Width-Independent Algorithm for Optimization Problems 77

1 Introduction

The design and analysis of efficient algorithms for optimization problems is a fundamental task in computer science. Due to the wide range of applications from economics and game theory to extremal combinatorics and machine learning, several techniques such as the simplex and interior point methods have been developed. Among the existing methods, the *Ellipsoid* is a powerful one developed by Shor, Nemirovsky and Yudin and successfully applied for convex optimization problems (e.g., see [NN94]). Later on, Khachiyan [Kha79] modified the Ellipsoid method to obtain the first polynomial algorithm for linear programming. This caused great excitement in the community and led to a large number of research papers in this area (see [BGT81] for a survey about the Ellipsoid method and the first follow-up results). Although, the method provides a rich framework for theoretical studies, it converges slowly to an optimal solution in practice as its running time depends highly on the bit description of a problem. Therefore, the design of faster algorithms for linear programming is a major problem in computer science. *Multiplicative Weights Update Method (MWUM)* is a fast and simple approximation scheme that has been applied in many fundamental constrained problems such as convex optimization, packing, and covering linear programming (e.g., see [PST91, GK98, GK04, KY07]). The method is quite general, and it might be considered a potential candidate to replace the Ellipsoid method, where an approximate solution suffices. It is usually explained as a learning process for solving a *stock market prediction* problem over time, which is defined as follows: There is a decision maker who wants to predict a stock market at the beginning of each day, and he has a set of n experts telling their predictions at the beginning of day, and the value of each prediction is made known at the end of the day. The decision maker's goal is to design an algorithm to minimize his cost in a long run, which is the consequence of his predictions. One of the first results for solving this problem is from Freund and Schapire [FS99], who presented a *weighted majority algorithm* (see Section A.1). They showed that the cost incurred by this algorithm is bounded from above by the the cost of an algorithm that follows the best expert, who is unknown

to the decision maker, with an additive error $O(\log n)$. When the method is applied for linear programming, it iteratively combines the constraints into a single weighted constraint, and then solves just one constraint. The weights get updated based on the solution obtained in previous round, and increased when the solution satisfies the constraint (see Section A.3).

One key idea that lies at the core of these algorithms is that the decision maker maintains a list of weights for the experts. Then, according to the value of each expert's prediction, he updates the experts' weights with a multiplicative factor, say $1 + \varepsilon$ if an expert's prediction is correct, or a factor $1 - \varepsilon$, if it is wrong, where $\varepsilon > 0$ is a given parameter to the algorithm. The decision maker then decides which expert to follow based on new weights. Intuitively, the idea works because it adds higher weights for experts with fewer mistakes in a long run. This simple idea is powerful enough to give efficient algorithms for various kinds of problems. For instance, Garg and Könemman [GK98] applied this framework for the multicommodity flows problems and designed a fast algorithm. More recently, Arora, Hazan, and Kale [AHK06] unified these algorithms and presented them as a meta algorithm. Grigoriadis and Khachiyan [GK95] introduced *randomized fictitious play* method for computing the mixed strategies of a class of zero-sum games. The method can also be considered as MWUM since in each round the probability distribution on the set of strategies gets updated such that the better strategies are more likely to be picked.

In this thesis, we adopt the Multiplicative Weights Update Method for solving some optimization problems arising in game theory and show the efficiency and improvements over previous works that applied the Ellipsoid method. We basically consider two fundamental optimization problems that appear in algorithmic game theory, namely, the social welfare maximization problem and computing an equilibrium for a class of zero-sum games. We briefly discuss our results in the following sections.

1.1. Algorithmic Mechanism Design

Algorithmic mechanism design studies optimization problems in which part of the input is collected from self-interested players who can manipulate the algorithm by mis-reporting their parts of the input if that improves their own objective functions. Combinatorial auctions are an important class of such problems in which the auctioneer tries to design a mechanism for selling items to players such that (1) he motivates players to reveal their true valuations, so-called truthful-mechanism, and (2) he maximizes the total valuations of players, called the social welfare maximization (SWM). In this setting, a considerable volume of data is reported by the buyers, which are valuations for every subset of items. The celebrated VCG mechanism is the only ever known mechanism that is truthful and also maximizes the social welfare. Applying VCG requires solving SWM exactly and then, according to the optimal solution, the mechanism assigns to each player a subset of items. The mechanism also charges each player according to a specific payment rule called VCG payment, which we will explain in Chapter 2. Unfortunately, SWM is known to be NP-hard, and hence, the VCG mechanism is intractable.

Formally speaking, a combinatorial auction consists of a set of m items, say $[m]$, n

players, and every player $i \in [n]$ has a valuation function v_i over the set of all subsets of items. SWM is formulated as follows:

$$\begin{aligned}
&\max \sum_{i,S} v_i(S) x_{i,S} \\
&\sum_{S} x_{i,S} \leqslant 1 \qquad \text{for all } i \in [n] \\
&\sum_{i,S;j\in S} x_{i,S} \leqslant 1 \qquad \text{for all items } j \in [m] \\
&x_{i,S} \in \{0,1\}
\end{aligned}$$

where $x_{i,S} = 1$ indicates the assignment of S to the i-th player. Note that the first set of constraints means that each player gets at most one subset of items, and the second one is concerned with the disjointness of the allocated subsets to players.

Based on the VCG mechanism, Lavi and Swamy [LS05, LS11] introduced a randomized mechanism for solving this problem. The mechanism approximates social welfare within a factor $\alpha \in [0,1]$ and guarantees truthfulness-in-expectation, which means each player maximizes its expected utility by telling the truth.

Basically, their technique is based on two assumptions: the LP-relaxation of SWM can be solved exactly, and an *α-integrality gap verifier* is available. The latter implies that if one scales down the polytope of the relaxed version of SWM, say Q, by a factor of α, then points in the new polytope are contained in the convex hull of the integral points in the original polytope. Therefore, any fractional point that maximizes the social welfare within factor α can be written as a convex combination of integral points, called the *convex decomposition*, which provides a distribution over a set of integral points. So, the randomized mechanism picks an integral solution according to the probability distribution and the rest of the mechanism almost follows the VCG mechanism. The Lavi-Swamy mechanism uses the Ellipsoid method both for solving the LP and for finding the convex decomposition, which are not efficient in practice. Our results in Chapter 2 is concerned with the design and analysis of a combinatorial algorithm for the Lavi-Swamy mechanism. Our contribution includes the design of a faster algorithm for convex decomposition, and then we consider a more general problem in which we do not solve the LP-relaxation of SWM exactly but instead apply the Multiplicative Weights Update Method for solving it and show that our mechanism is approximately truthful-in-expectation. In the following, we briefly explain our results.

Convex Decomposition. Similar to the Lavi-Swamy's framework, we first assume that the LP-relaxation of social welfare maximization can be solved exactly and efficiently and let x^* denote the optimal solution of the LP-relaxation. Then, our problem reduces to the design of a practical algorithm for the convex decomposition. Carr and Vempala [CV02] showed how to construct a convex combination of integral points in the polytope Q dominating x^* using a polynomial number of calls to an α-integrality-gap-verifier. The Lavi-Swamy mechanism [LS11] modified the construction to get an exact convex decomposition. In fact, they use the Ellipsoid method to get such a convex decomposition. We show how to replace the use of the Ellipsoid method with the Multiplicative Weights Update Method and obtain a convex decomposition

of $\alpha x^*/(1+4\varepsilon)$ for every $\varepsilon > 0$. We also derive an upper bound of $O(s\varepsilon^{-2}\log s)$ for the number of calls to the α-integrality-gap-verifier where s is the number of non zero entries in x^*. Recently, Kraft, Fadaei, and Bichler [KFB14] considered the same problem and described an alternative method for finding a convex combination. However, their construction is less efficient in two aspects. First, it requires $O(s^2\varepsilon^{-2})$ calls of the α-integrality-gap-verifier. Second, the number of non-zero coefficients in the convex decomposition might be as large as $O(s^3\varepsilon^{-2})$, while in our case it is $O(s\varepsilon^{-2}\log s)$.

Faster Mechanism Design. We consider a more general problem, that is, we do not want to solve LP-relaxation of SWM exactly and assume that there is an ε-approximation algorithm for the LP-relaxation of SWM for every $\varepsilon > 0$, denoted by A. Garg and Könemann [GK98] show how to modify the Multiplicative Weights Update Method for packing linear programming and obtain an FPTAS for it; hence, our assumption regarding the existence of A is valid. Using this, we show how to construct a randomized mechanism and prove that for a given ε_0, our mechanism is $(1-\varepsilon_0)$-*truthful-in-expectation*, which means every player maximizes his expected utility up to a factor of $1-\varepsilon_0$ by telling the truth. Note that this step is useful in particular, when the dimension is exponential as is the case in combinatorial auction's.

The results of Chapter 2 have been published in proceedings of the 8th International Symposium on Algorithmic Game Theory, SAGT'15, [EMR15].

1.2. Approximating Saddle Points

Approximating a saddle point of a given function is an overarching and fundamental problem that has various kinds of applications from economics and game theory to dynamical systems and optimization, for example. Suppose that $F : X \times Y \to \mathbb{R}$, $X \subseteq \mathbb{R}^m$ and $Y \subseteq \mathbb{R}^n$, is a function that is convex in X and concave in Y. Then the *saddle point* $(x^*, y^*) \in X \times Y$ of F is defined as

$$F(x^*, y^*) = \inf_{x \in X} F(x, y^*) = \sup_{y \in Y} F(x^*, y).$$

Let us consider an important class of games, known as a 2-player zero-sum game, with a payoff function $F : X \times Y \to \mathbb{R}$ in which a saddle point of F (or equivalently, an equilibrium point) plays an important role. In this game, there are two players: one is the minimizer who selects point $x \in X$, so-called strategy x, and the other selects strategy $y \in Y$, and each is unaware of the choice of the other. Then, their choices are made known and the minimizer pays the amount of $F(x, y)$ to the maximizer. It is not hard to check that any saddle point for F corresponds to a Nash equilibrium because no player has an interest to change his current strategy, given that the other player sticks to his current strategy. Hence, saddle points are translated to the optimal strategies in a game-theoretical setting. The famous Saddle-Point Theorem (e.g., see [Roc70]) says that when X and Y are closed convex sets, a saddle point for F exists. However, computing the saddle point is not as clear as the existential result as it depends on F and its domain, which might be quite challenging. In the 50s, Brown [Bro51] introduced the *fictitious play* technique for computing the equilibrium of a

2-player zero-sum game, with a payoff function $F(x, y) = x^T Ay$, where $A \in \mathbb{R}^{m\times n}$, and the strategy sets are probability distributions over rows and columns, known as *mixed strategies*. Here, the minimizer, row player, instead of choosing a point, selects a probability distribution over rows; and the maximizer, column player, chooses a distribution over columns, and $x^T Ay$ is the expected payoff which the row player should pay to the column player. The fictitious play method is a learning rule that proceeds in rounds. In each round, each player updates his strategy by applying the best response strategy to the current opponent's strategy. Robinson [Rob51] showed that if the number of rounds goes to infinity, then the fictitious play converges to the optimal strategies for players. Due to the computation overhead and technical reasons, the notion of an ε-approximation saddle point emerged: for every $\varepsilon \in (0, 1)$, point $(x^*, y^*) \in X \times Y$ is an *ε-approximation saddle point* of $F : X \times Y \to \mathbb{R}$ if we have

$$\sup_{y\in Y} F(x^*, y) \leqslant \inf_{x\in X} F(x, y^*) + \varepsilon.$$

Based on fictitious play, Grigoriadis and Khachiyan [GK95] presented a randomized algorithm for computing *ε-optimal strategies* (or equivalently, the ε-approximation saddle point) for a 2-player zero-sum game with a payoff function $x^T Ay$, where $A \in [-1, 1]^{m\times n}$. Generally speaking, in randomized fictitious play, each player takes a random direction to update his strategy according to a given distribution which gives large weight to good responses, instead of applying the best response. They also proved an expected upper bound $O((m + n) \log(n + m)/\varepsilon^2)$ for the running time of the algorithm. When the strategy sets are convex sets, one can easily see that computing the ε-approximation saddle point can be reformulated as a convex minimization problem over a convex set, and hence, any algorithm for solving this class of problems, e.g., the Ellipsoid method, can be used to compute them in time polynomial in the input size and polylog$(\frac{1}{\varepsilon})$ (see, e.g., [GLS93]). However, there has recently been an increasing interest in finding simpler and faster approximation algorithms for this type of problem, sacrificing the dependence on ε from polylog$(\frac{1}{\varepsilon})$ to poly$(\frac{1}{\varepsilon})$, in exchange for efficiency in terms of other input parameters; see e.g. [AHK05, AK07, BBR04, GK92, GK95, GK96, GKPV01, GK98, GK04, Kha04, Kal07, LN93, KY07, You01, DJ07, PST91]. In Chapter 3 we show that it is possible to design a randomized algorithm for computing an ε-approximation saddle point of convex-concave functions over the product of two convex bounded sets. Our algorithm is based on combining a technique developed by Grigoriadis and Khachiyan [GK95], which is a randomized variant of Brown's fictitious play [Bro51], with the recent results on random sampling from convex sets (see, e.g., [LV06, Vem05]). The algorithm finds an ε-approximation saddle point for a convex-concave functions in an expected number of $O\left(\frac{\rho^2(n+m)}{\varepsilon^2} \log \frac{R}{\varepsilon}\right)$ iterations, where in each iteration two points are sampled from log-concave distributions over strategy sets. It is assumed that X and Y have inscribed balls of radius $1/R$ and circumscribing balls of radius R and $\rho = max_{x\in X, y\in Y}|F(x, y)|$. In particular, the algorithm requires $O^*\left(\frac{\rho^2(n+m)^6}{\varepsilon^2} \log R\right)$ calls to a membership oracle, where $O^*(\cdot)$ suppresses polylogarithmic factors that depend on n, m, and ε.

The results of Chapter 3 have been published in the Journal of Algorithmica 2014 [EMMR14].

2

Towards More Practical Algorithmic Mechanism Design

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 who can manipulate the algorithm by mis-reporting their parts of the input, if that improves their own objective functions. Algorithmic mechanism design aims at polynomial-time algorithms that (approximately) optimize a global objective function, subject to the *strategic requirement* that the best strategy for the players is to *truthfully* report their part of the input. Such algorithms are called *truthful mechanisms*.

Combinatorial auctions provide a general framework in game theory that consists of m distinct items to be sold to a set of n players. Each player has a non-negative (reported) *valuation function* over all subsets of items that has a non-decreasing property and assigns zero to the empty set. In this setting, the objective function is called *social welfare* and is defined as the total valuation of players for assigned items. The goal is to design a mechanism that charges each player a price and allocates disjoint subsets of items to the players such that social welfare is maximized and players are motivated to declare their true valuations.

If the underlying optimization problem can be efficiently solved to optimality, the celebrated VCG mechanism (see, e.g., [NRTV07]) achieves truthfulness, maximized social-welfare, and polynomial running time.

In general, the underlying optimization problem is NP-hard and can only be solved approximately. Lavi and Swamy [LS05, LS11] introduced a general technique that uses an approximation algorithm of the underlying problem (social welfare) and converts it to a randomized mechanism that is *truthful-in-expectation*, which means each player maximizes its expected utility by telling the truth. Basically, their technique is based on two assumptions, namely, the LP-relaxation of the underlying optimization problem can be solved exactly and an α-*integrality gap verifier* is available. The latter implies

that after scaling down any feasible fractional point by a factor α, it can be represented as a convex combination of integral points, that is, a distribution over a set of integral points.

It turns out that the Lavi-Swamy mechanism is a powerful method (e.g., see [LS05, LS11, CEF10, HKV11] for applications), but is unlikely to be efficient in practice, because the mechanism uses the Ellipsoid method both for solving the LP and for finding the convex decomposition. In order to have a clearer picture of the mechanism, we summarize the mechanism in three steps (more details in Section 2.2):

1. Let v_i be the linear valuation function of the i-th player and $v = \sum_i v_i$ be the accumulated valuation. Solve the LP-relaxation of social welfare with packing polytope $\mathcal{Q}$, i.e., find a maximizer $x^* = \text{argmax}_{x \in \mathcal{Q}} v(x)$, and determine the VCG price p_i, for every $i \in [n]$. 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*, i.e., $\alpha \cdot x^* = \sum_{l \in \mathcal{N}} \lambda_l x^l$, where $\{x^l : l \in \mathcal{N}\}$ is the set of integral points contained in $\mathcal{Q}$, namely $\mathcal{Q}_\mathcal{I}$, $\lambda_l \geqslant 0$, and $\sum_{l \in \mathcal{N}} \lambda_l = 1$.

3. Pick the integral solution x^l with probability λ_l, and charge the i-th player the price $p_i v_i(x^l)/v_i(x^*)$ if $v_i(x^*) > 0$ and zero, otherwise.

The Lavi-Swamy mechanism approximates social welfare within a factor of α, called *α-social efficiency*, and guarantees truthfulness-in-expectation, i.e., it converts a truthful fractional mechanism into an 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 programming problem; an exact solution requires the use of the Ellipsoid method (see e.g. [GLS88]), if the dimension is exponential. Furthermore, until recently, the only method known to perform the decomposition in Step 2 is through the Ellipsoid method. An alternative method that avoids the use of the Ellipsoid method was recently given by Kraft, Fadaei, and Bichler [KFB14]. We comment on their result in the next section.

Our Results

Our result concerns the design and analysis of a combinatorial algorithm for the Lavi-Swamy mechanism. We first consider the case where the LP-relaxation of social welfare maximization, denoted by SWM, in Step 1 of the Lavi-Swamy mechanism can be solved exactly and efficiently. Then, our problem reduces to the design of a practical and combinatorial algorithm for the convex decomposition (i.e., Step 2).

Subsequently, we consider a more general problem, namely, the LP-relaxation in Step 1 of the Lavi-Swamy mechanism cannot be solved exactly, and we only have an approximate solution that maximizes the LP-relaxation within a factor of $1 - \varepsilon$. We now briefly explain our contribution.

Convex Decomposition. Suppose x^* is an arbitrary point of polytope $\mathcal{Q}$ and let $\mathcal{Q}_\mathcal{I}$ be the set of integral points contained in $\mathcal{Q}$. It is not hard to see that the convex decomposition of αx^* into points of $\mathcal{Q}_\mathcal{I}$ can be formulated as an LP (see (2.2) in

Section 2.2). Carr and Vempala [CV02] 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 [LS11] modified the construction to get an exact convex decomposition:

$$\alpha x^* = \sum_{i \in \mathcal{N}} \lambda_l x^l.$$

Their construction uses the Ellipsoid method.

Over the past 15 years, simple and fast methods [BI06, GK95, GK98, Kha04, KY07, PST91, You01] have been developed for solving packing and covering linear programming problem within an arbitrarily small error guarantee ε. These methods are based on the multiplicative weights update method (MWUM) [AHK06], 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 by the multiplicative weights update method and obtain a convex decomposition of $\alpha x^*/(1+4\varepsilon)$ for every $\varepsilon > 0$. Let s be the number of non-zero components of x^*. We also derive the upper bound $O(s\varepsilon^{-2} \log s)$ for both the number of calls to the α-integrality gap verifier and the size of the decomposition which is the number of nonzero coefficients.

Kraft, Fadaei, and Bichler [KFB14] considered the same problem and describe an alternative method for finding a dominating convex combination. However, their construction is less efficient in two aspects. First, it requires $O(s^2\varepsilon^{-2})$ calls to the α-integrality gap verifier. Second, the size of their convex decomposition might be as large as $O(s^3\varepsilon^{-2})$. In the combinatorial auction problem, $s = n + m$.

Our fast convex decomposition algorithm, together with Steps 1 and 3 of the Lavi-Swamy mechanism, implies a mechanism that is truthful-in-expectation and has $(\alpha/(1+4\varepsilon))$-social efficiency, that is, the expected social welfare is at least $\alpha/(1+4\varepsilon)$ times the maximum possible social welfare value.

Approximately Truthful-in-Expectation Mechanisms. In contrast to the Lavi-Swamy mechanism, let us assume that we do not want to solve the LP-relaxation of SWM exactly, but instead we want to use an ε-approximation algorithm $\mathcal{A}$ for it. Garg and Könemann [GK98] showed that there is an FPTAS for the packing problem, and hence, such $\mathcal{A}$ exists for every $\varepsilon > 0$. Using this, we first 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 a 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 $1-3$ and has $\alpha(1-\varepsilon)(1-\varepsilon_0)/(1+4\varepsilon)$-social efficiency.

Note that our fractional mechanism refines the one given in [DRVY11], 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 [EJ15] shows the applicability of these methods in practice.

Outline. In Section 2.1, we formally define notations and state preliminary results that are required in this chapter. In Section 2.2, we give an overview of the Lavi-Swamy mechanism. In Sections 2.3 and 2.4, we present our main results concerning a fast convex decomposition algorithm and the construction of an approximately truthful-in-expectation integral mechanism, respectively.

2.1. Notations, Definitions, and Preliminaries

In this section, we provide notations, definitions and some results that are required. We follow the standard notations in [NRTV07, Chapter 9].

Assume that we have a set of players $[n] = \{1, 2, \ldots, n\}$ and a set of all possible outcomes, denoted by Ω. Every player has a preference list over the outcomes, which is modeled by a *valuation function* $v_i : \Omega \to \mathbb{R}_+$, where $v_i \in \mathcal{V}_i$ and $\mathcal{V}_i$ is a set of all possible valuation functions for player i. Note that for each player $i \in [n]$, the *reported valuation function* $v_i \in \mathcal{V}_i$, in general, may differ from his *true valuation function* $\bar{v}_i \in \mathcal{V}_i$. For example, in *combinatorial auction*, we have m items to be sold to a set of n players. We consider the case when we have one copy of each item in our set. Here, the set of outcomes Ω is the set of all possible allocations of items to the players such that no two players share an item. In this setting, every player $i \in [n]$ gives the same value to any two outcomes that allocate the same subset to him. Thus, one can see v_i as specifying a value for each set S of items. A valuation function must be monotone, i.e., for $S \subseteq T$ we have $v_i(S) \leqslant v_i(T)$, and it should be normalized, i.e., $v_i(\emptyset) = 0$. Let $v = (v_1, \ldots, v_n)$ be an n-dimensional vector; $v_{-i} = (v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n)$ denotes an $(n-1)$-dimensional vector in which the i-th coordinate is removed. Thus, we have three different notations: $v = (v_1, \ldots, v_n) = (v_i, v_{-i})$.

A *mechanism* (A, p) consists of an allocation function $A : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \Omega$ and a vector of payment functions $p = (p_1, \ldots, p_n)$, where $p_i : \mathcal{V}_1 \times \cdots \times \mathcal{V}_n \to \mathbb{R}$ is the amount that player $i \in [n]$ should pay. The mechanism first gets valuation $v = (v_1, \ldots, v_n)$ from the players, then computes allocation $A(v)$ and charges player i the payment $p_i(v)$. The payments should be carefully chosen as to motivate all players to report their valuation truthfully. However, each player i is selfish and wants to maximize his own utility, which is

$$U_i(v) = \bar{v}_i(A(v)) - p_i(v),$$

where $\bar{v}_i$ and v_i denote the true and reported valuation of player i, respectively. A mechanism (A, p) is called *truthful* if, for every player i and every $v = (v_1, \ldots, v_n)$, we have

$$\bar{v}_i(A(\bar{v}_i, v_{-i})) - p_i(\bar{v}_i, v_{-i}) \geqslant \bar{v}_i(A(v_i, v_{-i})) - p_i(v_i, v_{-i}).$$

Intuitively, this means that every player would have no interest in misreporting his true valuation to the mechanism, since misreporting would not give him higher utility.

For a given allocation $\omega \in \Omega$, $\sum_{i\in[n]} v_i(\omega)$ is called the *social welfare*, which is an important quantity to measure the quality of the allocation ω.

Definition 2.1.1 (FPTAS)**.** *Suppose that $\mathcal{A}$ is an algorithm that solves an optimization instance within $(1 \pm \epsilon)$ factor of the optimum solution and its running time is a polynomial in the size of the instance and $1/\epsilon$. Then $\mathcal{A}$ is called a fully polynomial time approximation scheme, abbreviated as FPTAS.*

2.1.1. VCG Mechanism

In this subsection we explain the celebrated *VCG* mechanism which is the only known truthful mechanism for arbitrary valuation functions that also maximizes social welfare.

Definition 2.1.2 (VCG Mechanism)**.** *Mechanism (A, p) is called a VCG mechanism, if we have,*

I. $A(v) \in \operatorname{argmax}_{\omega\in\Omega} \sum_i v_i(\omega)$*; that is, A maximizes social welfare.*

II. For each $i \in [n]$,

$$p_i(v) = h_i(v_{-i}) - \sum_{j\neq i} v_j(A(v)),$$

where $h_i : \mathcal{V}_{-i} \to \mathbb{R}$ is a function which does not depend on v_i.

Now, we state the famous Vickrey-Clarke-Groves theorem and its proof.

Theorem 2.1.3 (Vickrey-Clarke-Groves [NRTV07])**.** *Every VCG mechanism is truthful.*

Proof. Fix i, v_{-i}, v_i and $\bar{v}_i$. We need to show that for player i with true valuation $\bar{v}_i$, the utility when declaring $\bar{v}_i$ is not less than the utility when declaring v_i. Denote $\bar{\omega} = A(\bar{v}_i, v_{-i})$ and $\omega = A(v_i, v_{-i})$. The utility of player i, when declaring $\bar{v}_i$ is

$$\bar{v}_i(\bar{\omega}) + \sum_{j\neq i} v_j(\bar{\omega}) - h_i(v_{-i}).$$

But when declaring v_i, the utility will be

$$\bar{v}_i(\omega) + \sum_{j\neq i} v_j(\omega) - h_i(v_{-i}).$$

Since $\bar{\omega}$ maximizes social welfare with valuation function $(\bar{v}_i, v_{-i})$ over all the allocations, we have,

$$\bar{v}_i(\bar{\omega}) + \sum_{j\neq i} v_j(\bar{\omega}) \geqslant \bar{v}_i(\omega) + \sum_{j\neq i} v_j(\omega).$$

Thus the same inequality holds when subtracting the same term $h_i(v_{-i})$ from both sides. □

2.1.2. Clarke Pivot Rule

In this subsection, we define Clarke pivot rule which provides a method to find a right function h_i in VCG mechanism, for each $i \in [n]$, such that the mechanism makes more sense. Therefore, in order to have a realistic mechanism we assume that the mechanism should satisfy following criteria.

No positive transfer. This means no player is ever paid money. Formally, for every $i \in [n]$ and $v = (v_1, \ldots, v_n)$ we have, $p_i(v) \geqslant 0$.

Individual rationality. This means players do not lose anything by participating truthfully in the mechanism, i.e., when every player i reports truth valuation he does not receive negative utility. Formally, for every $v = (v_1, \ldots, v_n)$ we have

$$\bar{v}_i(A(\bar{v}_i, v_{-i})) - p_i(\bar{v}_i, v_{-i}) \geqslant 0.$$

Now, we state the Clarke pivot rule and show the rule satisfies the above criteria.

Definition 2.1.4 (Clarke pivot rule)**.** *Set*

$$h_i(v_{-i}) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega).$$

Under this rule, we will have,

$$p_i(v) = \max_{\omega \in \Omega} \sum_{j \neq i} v_j(\omega) - \sum_{j \neq i} v_j(\omega^*),$$

where $\omega^* = A(v_1, \ldots, v_n)$, is the allocation given by the VCG mechanism (A, p).

intuitively, the rule means the player i pays an amount equal to the total damage that he causes the other players, the difference between the social welfare of the others with and without i's participation.

Lemma 2.1.5. *[NRTV07, Lemma 9.20] A VCG mechanism with Clarke pivot payment rule makes no positive transfers. If v_i for every i is a positive function then it is also individually rational.*

Proof. Assume that (A, p) be a VCG mechanism with the Clarke pivot rule. Define ω' to be the allocation maximizing $\sum_{j \neq i} v_j(\omega)$. To show the property of having no positive transfers, by definition we have

$$p_i(v) = \underbrace{\sum_{j \neq i} v_j(\omega')}_{= h_i(v_{-i})} - \sum_{j \neq i} v_j(\omega^*) \geqslant 0,$$

where the inequality follows from the fact that $\sum_{j \neq i} v_j(\omega') \geqslant \sum_{j \neq i} v_j(\omega)$, for every $\omega \in \Omega$ including ω^*. Let $\bar{\omega} = A(\bar{v}_i, v_{-i})$ be the VCG allocation, then the utility of

telling the truth for each player i, $i \in [n]$, is

$$\begin{aligned}\bar{v}_i(\bar{\omega}) - p_i(\bar{v}_i, v_{-i}) &= \bar{v}_i(\bar{\omega}) - \Big\{ \underbrace{\sum_{j \neq i} v_j(\omega')}_{h_i(v_{-i})} - \sum_{j \neq i} v_j(\bar{\omega}) \Big\} \\ &\geqslant \bar{v}_i(\bar{\omega}) + \sum_{j \neq i} v_j(\bar{\omega}) - \sum_{j \neq i} v_j(\omega') - \bar{v}_i(\omega') \geqslant 0,\end{aligned}$$

where the first inequality follows from the assumption, which is $\bar{v}_i(\omega') \geqslant 0$. And the second one follows from the fact that $\bar{\omega}$ maximizes the social welfare with valuation $(\bar{v}_i, v_{-i})$. Therefore, the mechanism is individually rational. □

In this chapter we simply call the VCG mechanism with Clarke pivot payment rule the VCG mechanism.

2.1.3. Randomized Mechanism Design

In this subsection we provide some definitions relating to *randomized mechanisms*. A *randomized mechanism* flips coins to determine $A(v)$ and $p_i(v)$, in which case $A(v)$ and $p_i(v)$ are random variables and a player's utility is also a random variable. We say a randomized mechanism is individually rational with probability $1 - \varepsilon$, for some $\varepsilon \in (0, 1)$, if the utility of any truth-telling player is non-negative with probability at least $1 - \varepsilon$.

Definition 2.1.6 (Υ-social efficiency)**.** *A randomized mechanism is called Υ-social efficiency if the expected social welfare is at least Υ times the maximum possible social welfare.*

Definition 2.1.7 (Truthful in Expectation)**.** *A randomized mechanism is truthful-in-expectation if the expected utility of every player i, $i \in [n]$, is maximized when he declares his true valuation $\bar{v}_i$ regardless of what others declare. That is, for any player i, any valuation $v_{-i} \in \mathcal{V}_{-i}$ and any true valuation $\bar{v}_i \in \mathcal{V}_i$*

$$\mathbb{E}[\bar{v}_i(A(\bar{v}_i, v_{-i})) - p_i(\bar{v}_i, v_{-i})] \geqslant \mathbb{E}[\bar{v}_i(A(v_i, v_{-i})) - p_i(v_i, v_{-i})].$$

Also we define the notion of $(1 - \delta)$-truthful-in-expectation, for every $\delta \in [0, 1)$, that is,

$$\mathbb{E}[\bar{v}_i(A(\bar{v}_i, v_{-i})) - p_i(\bar{v}_i, v_{-i})] \geqslant (1 - \delta)\mathbb{E}[\bar{v}_i(A(v_i, v_{-i})) - p_i(v_i, v_{-i})].$$

2.2. The Lavi-Swamy Mechanism

In this section we present an overview of the Lavi-Swamy mechanism. The VCG mechanism is the only known truthful mechanism for general valuation functions that also maximizes social welfare. However, this mechanism requires solving an optimization problem (i.e., $\max_{\omega \in \Omega} \sum_i v_i(\omega)$, where ω is a feasible allocation) that is NP-hard and hence computationally intractable for general valuation functions. In general, this underlying optimization problem (social welfare) can only be solved approximately.

Lavi and Swamy ([LS05, LS11]) showed that how an approximation algorithm can be turned into an approximation mechanism which is truthful-in-expectation. Their method applies to integer programming problems of the packing type for which the LP-relaxation can be solved exactly and an α-integrality gap verifier is available (for a formal definition see below). Let $\mathcal{Q} \subseteq \mathbb{R}^d_{\geqslant 0}$, be a packing polytope, i.e., $\mathcal{Q}$ is the intersection of finitely many half-spaces, and if $y \in \mathcal{Q}$ and $x \leqslant y$ then $x \in \mathcal{Q}$. Let us formulate social welfare maximization (SWM) in a combinatorial auction as a packing integer program. Suppose that $x_{i,S} \in \{0,1\}$ is a variable taking one if subset S is given to player i and zero otherwise. Then finding $\omega^* \in \operatorname{argmax}_{\omega\in\Omega} \sum_i v_i(\omega)$ reduces to solving the following integer programming:

$$\begin{aligned} &\max \sum_{i,S} v_i(S) x_{i,S} && (2.1)\\ &\sum_{S} x_{i,S} \leqslant 1 \qquad \text{for all } i \in [n] \\ &\sum_{i,S;j\in S} x_{i,S} \leqslant 1 \qquad \text{for all item } j \in [m] \\ &x_{i,S} \in \{0,1\} \end{aligned}$$

Note that the first n constraints guarantee that each player receives at most one subset of items and the next m constraints say that no two players share an item. The polytope $\mathcal{Q}$ is obtained by replacing the integrality constraints for $x_{i,S}$ by $0 \leqslant x_{i,S} \leqslant 1$. Note that the number of variables is $d = n2^m$. We use $\mathcal{Q}_{\mathcal{I}} := \mathcal{Q} \cap \mathbb{Z}^d$ for the set of integral points in $\mathcal{Q}$. In this example, Ω, the set of all possible outcomes of the mechanism, is $\mathcal{Q}_{\mathcal{I}}$. For simplicity, we denote the relaxed version of (2.1) as follows:

$$\max_{x\in\mathcal{Q}} V^T \cdot x,$$

where $V \in \mathbb{R}^d$ and $V^T \cdot x = \sum_{i,S} v_i(S) x_{i,S}$. We extend the definition of the valuation function to fractional allocations and define $v_i(x) = \sum_S v_i(S) \cdot x_{i,S}$, for every $x \in \mathbb{R}^d$. Let us fix an arbitrary $i \in [n]$ and set $x_{i,S} = 0$ for all set S in the above LP. Then we will get another packing polytope with a different objective function, which is denoted by

$$\max_{x\in\mathcal{Q}_{-i}} V_{-i}^T \cdot x.$$

Definition 2.2.1 (α-integrality gap verifier)**.** *A polynomial time algorithm $\mathcal{F}$ is an α-integrality gap verifier if for input $V \in \mathbb{R}^d$ and $x^* \in \mathcal{Q}$, $\mathcal{F}$ finds an integer solution $x \in \mathcal{Q}_{\mathcal{I}}$ such that*

$$V^T \cdot x \geqslant \alpha V^T \cdot x^*.$$

Throughout this section we assume that an α-integrality gap verifier for integer program (2.1), exists. Lavy and Swamy ([LS05, LS11]) used the α-integrality gap verifier to present an α-approximation randomized mechanism, their method is based on the VCG mechanism and has three main steps:

Step 1: Solving Packing LPs. In this step, the mechanism solves $n+1$ LPs; first it solves the LP-relaxation of (2.1), i.e., finds a maximizer $x^* = \operatorname{argmax}_{x \in Q} V^T \cdot x$, which gives a fractional VCG allocation. Then for determining VCG prices, it computes $\hat{x}_i \in \operatorname{argmax}_{\mathcal{Q}_{-i}} V_{-i}^T \cdot x$, for every $i \in [n]$. Finally using the Clarke pivot rule, for each player $i \in [n]$ the VCG price p_i is defined as follows:

$$p_i = \sum_{j \neq i} v_j(\hat{x}_i) - \sum_{j \neq i} v_j(x^*).$$

The allocation x^* and the VCG prices are a truthful mechanism for the fractional problem.

Step 2: Convex Decomposition. In this step, we explain how a given fractional solution x^* is represented as a convex combination of integral points contained in $\mathcal{Q}_{\mathcal{I}}$. Let $\mathcal{N}$ be an index set of points in $\mathcal{Q}_{\mathcal{I}} = \{x^l : l \in \mathcal{N}\}$. Finding convex combination can be formulated as the following LP:

$$\min \sum_{l \in \mathcal{N}} \lambda_l \tag{2.2}$$

$$\sum_{l \in \mathcal{N}} \lambda_l x_j^l = \alpha x_j^* \qquad \text{for all } j \tag{2.3}$$

$$\sum_{l \in \mathcal{N}} \lambda_l \geqslant 1$$

$$\lambda_l \geqslant 0 \qquad l \in \mathcal{N}$$

Since x^* is an optimal fractional solution of relaxed integer programing (2.1), it is a basic feasible solution and has $n+m$ non-zero entries. Let us define a non-empty set

$$E := \{j : x_j^* > 0\}.$$

Packing property allows us to restrict our attention to the constraints corresponding to the elements of E, i.e., consider equality constraints (2.3) for all $j \in E$. Clearly, the problem does not change, because we may replace any x^l with $x_j^l > 0$ for some $j \notin E$ by an $x^{l'}$ for which this component is zero. LP (2.2) has exponential number of variables, so we may consider its dual that has variables z and w_j for $j \in E$.

$$\max z + \alpha \sum_{j \in E} x_j^* w_j \tag{2.4}$$

$$z + \sum_{j \in E} x_j^l w_j \leqslant 1 \qquad \text{for all } l \in \mathcal{N} \tag{2.5}$$

$$z \geqslant 0$$

We first argue that the optimal value of the dual is one, and hence by strong duality the objective function in primal (2.2) also attains the same value. So an optimal solution to (2.2) yields the convex decomposition coefficients. By setting $z = 1$ and $w_j = 0$ for every $j \in E$, we have a feasible solution for dual program for which the

objective value is one. We claim that any feasible solution (z, w) has value at most one. Assume that there is pair (z, w) for which

$$z + \alpha \cdot \sum_{j \in E} x_j^* w_j > 1.$$

Let us define vector $W \in \mathbb{R}^d$, in which $W_j = w_j$, for $j \in E$ and $W_j = 0$, for $j \notin E$. Run the α-integrality gap verifier with vector W; it produces an integral solution x such that

$$W^T \cdot x = \sum_{j \in E} x_j w_j \geqslant \alpha \cdot W^T \cdot x^* = \alpha \sum_{j \in E} x_j^* w_j > 1 - z.$$

The above inequality shows that pair (z, w) violates the constraint (2.5) which contradicts (z, w) is a feasible solution.

We may therefore add the constraint $z + \alpha \sum_{j \in E} x_j^* w_j \leqslant 1$ to the dual with affecting the optimal solution. Lavi and Swamy [LS05, LS11] applied the Ellipsoid method to solve the dual with above extra constraint. Now, we see how they use the α-integrality gap verifier in the Ellipsoid method and this extra constraint. Let (z, w) be the current center of the Ellipsoid. They consider two cases:

Case 1.

$$z + \alpha \sum_{j \in E} x_j^* w_j > 1.$$

Then, by calling the α-integrality gap verifier, (similar to the proof of the upper bound for the objective function in the dual), we get an integral solution x so that

$$\sum_{j \in E} x_j w_j \geqslant \alpha \sum_{j \in E} x_j^* w_j > 1 - z,$$

and hence $z + \sum_{j \in E} x_j w_j \leqslant 1$ is a violated inequality in the dual which is required for the Ellipsoid method.

Case 2.

$$z + \alpha \sum_{j \in E} x_j^* w_j \leqslant 1$$

Then, they use this constraint to half the current ellipsoid.

The Ellipsoid method makes polynomially many cuts. The cuts appeared in first case provide polynomially many integral pints contained in $\mathcal{Q}_{\mathcal{I}}$. Solving the primal with the variables for these integral points gives the desired convex combination.

Step 3: Mechanism. Using the VCG fractional allocation x^* and prices p_i obtained in step 1 together with all λ_l computed in the previous steps, we state the Lavi-Swamy randomized mechanism as follows:

For every $l \in \mathcal{N}$, with probability λ^l do:

- pick x^l
- charge player i the price $p_i v_i(x^l)/v_i(x^*)$ if $v_i(x^*) > 0$ and zero otherwise.

The Lavi-Swamy mechanism approximates social welfare within factor α and guarantees truthfulness-in-expectation, i.e., it converts a truthful fractional mechanism into an α-approximately truthful-in-expectation integral mechanism. Here, for the sake of completeness we sketch the proof.

Observation 2.2.2. *The Lavi-Swamy Mechanism is truthful-in-expectation.*

Proof. Let us fix some $i \in [n]$. By using the linearity of valuation function $\bar{v}_i(x)$ and the fact that $\alpha x^* = \sum_{l \in \mathcal{N}} \lambda_l x^l$, we have

$$\begin{aligned} E[u_i(\bar{v}_i, v_{-i})] &= \sum_{l \in \mathcal{N}} \lambda_l \left(\bar{v}_i(x^l) - p_i \frac{\bar{v}_i(x^l)}{\bar{v}_i(x^*)} \right) \\ &= \bar{v}_i \left(\sum_{l \in \mathcal{N}} \lambda_l x^l \right) - p_i \bar{v}_i \left(\sum_{l \in \mathcal{N}} \lambda_l x^l \right) / \bar{v}_i(x^*) \\ &= \bar{v}_i(\alpha x^*) - p_i \bar{v}_i(\alpha x^*) / \bar{v}_i(x^*) \\ &= \alpha(\bar{v}_i(x^*) - p_i). \end{aligned}$$

and hence truthfulness-in-expectation follows from the truthfulness of the VCG mechanism for the fractional problem. □

2.3. The Fast Algorithm for Convex Decompositions

In this section we present an algorithm to obtain an exact convex decomposition $\alpha x^* = \sum_{i \in \mathcal{N}} \lambda_l x^l$ for the case of packing linear programmings. The Lavi-Swamy mechanism uses the Ellipsoid method to get these coefficients. We show an approximate version that replaces the use of the Ellipsoid method by the multiplicative weights update method (MWUM). In the following theorem, we state the main result of this section.

Theorem 2.3.1. *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*

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

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

This section is structured as follows. We first review Khandekar's FPTAS for covering linear programming (Subsection 2.3.1). Using the FPTAS and the α-integrality gap verifier we construct a dominating convex combination for $\alpha x^*/(1 + 4\varepsilon)$, where $x^* \in \mathcal{Q}$ is arbitrary (Subsection 2.3.2). In Subsection 2.3.3, we show how to convert a dominating convex combination into an exact convex decomposition. Finally, in Subsection 2.3.4, we put the pieces together and prove the above theorem.

2.3.1. An FPTAS for Covering Linear Programs

In this subsection, we present an FPTAS for covering LP's and then show its correctness and runtime. In general, a covering LP is formulated as follows:

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \geqslant b \\ & x \geqslant 0 \end{aligned} \tag{2.6}$$

where $A \in \mathbb{R}^{m \times n}_{\geqslant 0}$ is an $m \times n$ matrix with non-negative entries, $c \in \mathbb{R}^n_{\geqslant 0}$ and $b \in \mathbb{R}^m_{\geqslant 0}$ are non-negative vectors. Note that a constraint with $b_i = 0$ may be dropped since any solution satisfying the remaining constraints will also satisfy it. Column j with $c_j = 0$, allows one to drop all rows i with $a_{ij} > 0$, because these constraints can be satisfied by making x_j sufficiently large. After deleting these rows, the j-th column contains no non-zero entry and hence it and variable x_j may be deleted. We may therefore assume that $c \in \mathbb{R}^n_{>0}$ and $b \in \mathbb{R}^m_{>0}$ are positive vectors. We may also assume that each column of A contains at least one positive entry as otherwise the corresponding variable can be dropped and that each row of A contains at least one positive entry as otherwise the problem is infeasible. The linear program does not have to be given explicitly. Rather, we assume the availability of a *κ-approximation oracle* for some $\kappa \in (0, 1]$, denoted by $\mathcal{O}_\kappa(\cdot)$ and defined as follows,

$\mathcal{O}_\kappa(z)$**:** Given $z \in \mathbb{R}^m_{\geqslant 0}$, 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:

$$\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i} \geqslant \kappa \cdot \max_{j' \in [n]} \frac{1}{c_{j'}} \sum_{i=1}^m \frac{z_i a_{ij'}}{b_i} \tag{2.7}$$

For a given exact oracle (i.e., $\kappa = 1$), Khandekar [Kha04] gave an algorithm which computes a feasible solution $\hat{x}$ to the covering LP (2.6) such that $c^T \hat{x} \leqslant (1 + 4\varepsilon) z^*$ 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.

We extend Khandekar's algorithm and show that if the exact oracle is replaced by a κ-approximation oracle, it computes a feasible solution $\hat{x} \in \mathbb{R}^n_{\geqslant 0}$ to (2.6) such that $c^T \hat{x} \leqslant (1 + 4\varepsilon) z^* / \kappa$. The algorithm is given as Algorithm 1 and can be thought of as the algorithmic dual of the FPTAS for multicommodity flows given in [GK98].

For completeness, we include a proof of correctness and an upper bound for the number of iterations; our argument is slightly shorter than the Khandekar's proof since Khandekar actually solved a more general problem. In what follows we present the algorithm.

Theorem 2.3.2. *Let $\varepsilon \in (0, \frac{1}{2}]$ and let z^* be the value of an optimum solution to (2.6). Algorithm 1 terminates in at most $m\lceil \varepsilon^{-2} \log m \rceil$ iterations with a feasible solution $\hat{x}$ of (2.6) of at most $m\lceil \varepsilon^{-2} \log m \rceil$ positive components. At termination, it holds that*

$$c^T \hat{x} \leqslant \frac{(1 + 4\varepsilon)}{\kappa} z^*. \tag{2.8}$$

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

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

Ensure: A feasible solution $\hat{x} \in \mathbb{R}^n_{\geqslant 0}$ to (2.6) s.t. $c^T\hat{x} \leqslant \frac{(1+4\varepsilon)}{\kappa} z^*$

1: $x(0) := 0$; $t := 0$; and $T := \frac{\log m}{\varepsilon^2}$
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** $\hat{x} = \frac{x(t)}{M(t)}$

Proof. We use A_i to denote the i-th row of A. The algorithm constructs vectors $x(t) \in \mathbb{R}^n_{\geqslant 0}$, for $t = 0, 1, \ldots,$ until $M(t) := \min_{i\in[m]} A_i x(t)/b_i$ becomes at least $T := \frac{\log m}{\varepsilon^2}$. 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

$$p_i(t) := (1-\varepsilon)^{A_i x(t-1)/b_i}, \tag{2.9}$$

and set $p_i(t) = 0$ for $i \notin L(t), i \in [m]$. At each time t, the algorithm calls the oracle with the vector[1] $z_t = p(t)/\|p(t)\|_1$, and increases the variable $x_{j(t)}$ by

$$\delta(t) := \min_{i\in L(t) \text{ and } a_{i,j(t)}\neq 0} \frac{b_i}{a_{i,j(t)}}, \tag{2.10}$$

where $j(t)$ is the index returned by the oracle. Note that the RHS of (2.7) is positive for our choice of z_t since every row of A contains a non-zero entry and hence $\sum_{i\in L(t)} p_i(t)a_{ij(t)}/(b_i c_{j(t)}) > 0$. This conclude that there exist at least one $i \in L(t)$ which $a_{i,j(t)}$ is non zero and thus $\delta(t) > 0$ always. In each iteration, some entry of x is increased and hence the values $A_i x(t)/b_i$ are non-decreasing. Thus $L(t+1) \subseteq L(t)$ for all t. At the end, we scale $x(t)$ by $M(t)$ to guarantee feasibility.

Let $\mathbf{1}_j$ denote the j-th unit vector of dimension n and $B \in \mathbb{R}^{m\times m}$ be a diagonal matrix with entries $b_{ii} = b_i$. Feasibility is obvious since we scale by $M(t)$. The bound on the number of iterations is also obvious since in each iteration at least one of the $A_i x/b_i$ increases by one and we remove i from the active list once $A_i x/b_i$ reaches T. We conclude that the number of iterations is bounded by $m\lceil T\rceil$. Let t_0 be the number of iterations, i.e., vectors $x(0)$, $x(1)$, $\ldots$, $x(t_0)$ are defined and $M(t_0 - 1) < T \leqslant M(t_0)$. In the t-th iteration exactly one entry of x is increased by $\delta(t)$ and hence $\mathbf{1}^T x(t_0) = \sum_{1\leqslant t\leqslant t_0} \delta(t)$ and $A_i x(t)/b_i \leqslant A_i x(t-1)/b_i + 1$ for $i \in L(t)$.

[1] We could do without the normalization of $p(t)$ by $\|p(t)\|_1$. However, for Corollary 3.4.2 the normalization is convenient.

To show (2.8), we analyze the decrease of $\|p(t)\|_1$. Let $t \leqslant t_0$. Then

$$\begin{aligned}
\sum_{i \in L(t)} (1-\varepsilon)^{A_i x(t)/b_i} &= \sum_{i \in L(t)} (1-\varepsilon)^{A_i x(t-1)/b_i + \delta(t) A_i \mathbf{1}_{j(t)}/b_i} \\
&= \sum_{i \in L(t)} p_i(t)(1-\varepsilon)^{\delta(t) A_i \mathbf{1}_{j(t)}/b_i} \\
&\leqslant \sum_{i \in L(t)} p_i(t)(1 - \varepsilon\delta(t) A_i \mathbf{1}_{j(t)}/b_i) \\
&\quad \text{(using (2.10) conclude that } \delta(t) A_i \mathbf{1}_{j(t)}/b_i \leqslant 1 \text{ and} \\
&\quad (1-\varepsilon)^x \leqslant 1 - \varepsilon x \text{ for all } \varepsilon \in [0,1),\ x \in [0,1]) \\
&= \|p(t)\|_1 \left(1 - \frac{\varepsilon\delta(t) p(t)^T B^{-1} A \mathbf{1}_{j(t)}}{\|p(t)\|_1}\right) \\
&\leqslant \|p(t)\|_1 e^{-\varepsilon\delta(t)\frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}} \quad \text{since } 1 - x \leqslant e^{-x}. \qquad (2.11)
\end{aligned}$$

By using $L(t+1) \subseteq L(t)$, we have

$$\begin{aligned}
\|p(t+1)\|_1 &= \sum_{i \in L_{(t+1)}} (1-\varepsilon)^{A_i x(t)/b_i} \\
&\leqslant \sum_{i \in L_{(t)}} (1-\varepsilon)^{A_i x(t)/b_i} \qquad (2.12)
\end{aligned}$$

and hence, applying inequalities (2.11) and (2.12) we get

$$\|p(t+1)\|_1 \leqslant \sum_{i \in L(t)} (1-\varepsilon)^{A_i x(t)/b_i} \leqslant \|p(t)\|_1 e^{-\varepsilon\delta(t)\frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}}. \qquad (2.13)$$

Let $i_0 \in L(t_0)$ be arbitrary. Then

$$\begin{aligned}
(1-\varepsilon)^{A_{i_0} x(t_0)/b_{i_0}} &\leqslant \sum_{i \in L(t_0)} (1-\varepsilon)^{A_i x(t_0)/b_{i_0}} \\
&\leqslant \|p(t_0)\|_1 e^{-\varepsilon\delta(t_0)\frac{p(t_0)^T}{\|p(t_0)\|_1} B^{-1} A \mathbf{1}_{j(t_0)}} \\
&\leqslant \|p(0)\|_1 e^{-\varepsilon \sum_{1 \leqslant t \leqslant t_0} \delta(t)\frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}},
\end{aligned}$$

where the second inequality uses (2.11) for $t = t_0$ and the third inequality uses (2.13) for $0 \leqslant t < t_0$. Taking logs and using $\|p(0)\|_1 = m$, we conclude that

$$A_{i_0} x(t_0)/b_{i_0} \cdot \log(1-\varepsilon) \leqslant \log m - \varepsilon \sum_{1 \leqslant t \leqslant t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)} \qquad (2.14)$$

We next relate the objective value $c^T x(t_0) = \sum_{1 \leqslant t \leqslant t_0} c_{j(t)} \delta(t)$ at time t_0 to the optimal value z^* by the following claim.

Claim 1. $\sum_{1 \leqslant t \leqslant t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)} \geqslant \frac{\kappa \cdot c^T x(t_0)}{z^*}$.

Proof. Let $x^* \in \mathbb{R}^n_{\geqslant 0}$ be an optimal solution to (2.6). Since x^* is feasible, $B^{-1}Ax^* \geqslant \mathbf{1}$, and thus for any t,

$$p(t)^T B^{-1} A x^* \geqslant p(t)^T \mathbf{1} = \|p(t)\|_1.$$

By the choice of the index $j(t)$, we have that $\frac{1}{c_{j(t)}} p(t)^T B^{-1} A \mathbf{1}_{j(t)} \geqslant \frac{1}{c_j} \kappa p(t)^T B^{-1} A \mathbf{1}_j$ for all $j \in [n]$. Since $z^* = c^T x^*$, we conclude further

$$\begin{aligned}
z^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} &= \sum_{j \in [n]} c_j x_j^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} \\
&= \sum_{j \in [n]} c_j \cdot \frac{c_{j(t)}}{c_{j(t)}} x_j^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} \\
&\geqslant \sum_{j \in [n]} c_j \cdot \frac{c_{j(t)}}{c_j} x_j^* \kappa p(t)^T B^{-1} A \mathbf{1}_j \\
&= \kappa c_{j(t)} p(t)^T B^{-1} A x^* \\
&\geqslant \kappa c_{j(t)} \|p(t)\|_1.
\end{aligned}$$

Multiplying both sides of this inequality by $\delta(t)/\|p(t)\|_1$ and summing up over $1 \leqslant t \leqslant t_0$ finishes the proof of the claim. □

Using the claim above, we deduce from (2.14)

$$A_{i_0} x(t_0)/b_{i_0} \cdot \log(1-\varepsilon) \leqslant \log m - \varepsilon \cdot \frac{\kappa \cdot c^T x(t_0)}{z^*}$$

Dividing both sides by $M(t_0)$, arranging, and using $M(t_0) \geqslant T = (\log m)/\varepsilon^2$ and $A_{i_0} x(t_0)/b_{i_0} \leqslant A_{i_0} x(t_0 - 1)/b_{i_0} + 1 \leqslant T + 1$, and $\frac{\log \frac{1}{1-\varepsilon}}{\varepsilon} \leqslant 1 + 2\varepsilon$, valid for all $\varepsilon \in (0, \frac{1}{2}]$, we obtain

$$\begin{aligned}
\frac{\kappa \cdot c^T \hat{x}}{z^*} = \frac{\kappa \cdot c^T x(t_0)}{M(t_0) z^*} &\leqslant \frac{\log \frac{1}{1-\varepsilon}}{\varepsilon} \cdot \frac{A_{i_0} x(t_0)/b_{i_0}}{M(t_0)} + \frac{\log m}{\varepsilon \cdot M(t_0)} \\
&\leqslant (1 + 2\varepsilon) \frac{T+1}{T} + \varepsilon \leqslant 1 + 4\varepsilon.
\end{aligned}$$

□

We observe that the proof of Theorem 2.3.2 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 $z \in \mathbb{R}^m_{\geqslant 0}$, such that $\mathbf{1}^T z = 1$, the oracle finds a column j of A such that $z^T A \mathbf{1}_j \geqslant 1$.*

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

Proof. Recall (2.14):

$$A_{i_0}x(t_0)/b_{i_0} \cdot \log(1-\varepsilon) \leqslant \log m - \varepsilon \sum_{1 \leqslant t \leqslant t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}.$$

With assumption $b = \mathbf{1}$, we have,

$$A_{i_0}x(t_0) \cdot \log(1-\varepsilon) \leqslant \log m - \varepsilon \sum_{1 \leqslant t \leqslant t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} A \mathbf{1}_{j(t)}.$$

The vector $z_t = p(t)/\|p(t)\|_1$ satisfies $\mathbf{1}^T z_t = 1$. Apply oracle $\mathcal{O}'$ with input vector z_t, it returns index $j(t)$ such that we have $\frac{p(t)^T}{\|p(t)\|_1} A \mathbf{1}_{j(t)} \geqslant 1$. Thus, we have

$$A_{i_0}x(t_0) \cdot \log(1-\varepsilon) \leqslant \log m - \varepsilon \cdot \mathbf{1}^T x(t_0).$$

Proceeding as in the proof of Theorem 2.3.2, we get the result. □

2.3.2. 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 $O(s\varepsilon^{-2}\log s)$ calls to the oracle, where s is the support of x^*.

Theorem 2.3.3. *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*

$$\frac{\alpha}{1+4\varepsilon} \cdot x^* \leqslant \bar{x} = \sum_{l \in \mathcal{N}} \lambda_l x^l.$$

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

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

$$\begin{aligned} \min \quad & \sum_{l \in \mathcal{N}} \lambda_l \\ s.t. & \sum_{l \in \mathcal{N}} \lambda_l x_j^l \geqslant \alpha \cdot x_j^* \quad \text{for all } j, \\ & \sum_{l \in \mathcal{N}} \lambda_l \geqslant 1, \\ & \lambda_l \geqslant 0. \end{aligned} \tag{2.15}$$

Clearly, we can restrict our attention to the $j \in S^+ := \{j : x_j^* > 0\}$ and rewrite the constraint for $j \in S^+$ as $\sum_{l \in \mathcal{N}} \lambda_l x_j^l/(\alpha \cdot x_j^*) \geqslant 1$. 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):

$$a_{j,i} := \begin{cases} x_j^l/(\alpha x_j^*) & 1 \leqslant j \leqslant s, l \in \mathcal{N} \\ 1 & j = s+1, l \in \mathcal{N} \end{cases}$$

Thus we can apply Corollary 3.4.2 of Section 2.3.1, 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 $w \in \mathbb{R}^s_{\geqslant 0}$, $z \in \mathbb{R}_{\geqslant 0}$, and $\sum_{j=1}^{j=s} w_j + z = 1$. Oracle $\mathcal{O}'$ needs to find a column l such that $\tilde{z}^T A\mathbf{1}_l \geqslant 1$. In our case $\tilde{z}^T A\mathbf{1}_l = \sum_{j=1}^s w_j x_j^l/\alpha x_j^* + z$, and we need to find a column l 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 $x^l \in \mathcal{Q}_\mathcal{I}$ such that

$$V^T x^l = \sum_{j \in S^+} \frac{w_j}{\alpha x_j^*} x_j^l \geqslant \alpha \cdot V^T x^* = \sum_{j \in S^+} w_j,$$

and hence,

$$\sum_{j \in S^+} \frac{w_j}{\alpha x_j^*} x_j^l + z \geqslant \sum_{j \in S^+} w_j + z = 1.$$

Thus i is the desired column of A.

It follows by Corollary 3.4.2 that Algorithm 1 finds a feasible solution $\lambda' \in \mathbb{R}^{|\mathcal{N}|}_{\geqslant 0}$ to the covering LP (2.15), 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 $l \in \mathcal{N}'$, where $\mathcal{N}'$ is the index set returned by oracle $\mathcal{O}'$ and $|\mathcal{N}'| \leqslant s\lceil \varepsilon^{-2} \log s \rceil$ also $\Lambda := \sum_{l \in \mathcal{N}'} \lambda'_i \leqslant (1 + 4\varepsilon)$. Scaling λ'_i by Λ, we obtain a set of multipliers $\{\lambda_l = \lambda'_l/\Lambda : \ l \in \mathcal{N}'\}$, such that $\sum_{l \in \mathcal{N}'} \lambda_l = 1$ and

$$\sum_{l \in \mathcal{N}'} \lambda_l x^l \geqslant \frac{\alpha}{1 + 4\varepsilon} x^*. \tag{2.16}$$

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

2.3.3. From Dominating Convex Combination to Exact Convex Decomposition

In this subsection, 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 [LS05], but was not made explicit. Kraft, Fadaei, and Bichler [KFB14] describe an alternative construction. Their

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 $\sum_{i\in\mathcal{N}} \lambda_l x^l$ of integral points in $\mathcal{Q}_\mathcal{I}$ dominating x^*.
Ensure: A convex decomposition $x^* = \sum_{i\in\mathcal{N}'} \lambda_l x^l$ with $x^l \in \mathcal{Q}_\mathcal{I}$.
1: **while** there is an $i \in \mathcal{N}$ and a j such that $\lambda_l x^l_j > 0$ and $\sum_{h\in\mathcal{N}} \lambda_h x^h - \lambda_l \mathbf{1}_j \geqslant x^*$ **do**
2: replace x^l by $x^l - \mathbf{1}_j$.
3: **end while**
4: **while** $\Delta_j := \sum_{i\in\mathcal{N}} \lambda_l x^l - x^*_j > 0$ for some j **do**
5: {for all $i \in \mathcal{N}$ and all j: if $\lambda_l x^l_j > 0$ then $\sum_{h\in\mathcal{N}} \lambda_h x^h - \lambda_l \mathbf{1}_j < x^*$}
6: Let j be such that $\Delta_j > 0$ and let i be such that $\lambda_l x^l_j > 0$. Let $B = \{j \in S^+ : x^l_j \neq 0 \text{ and } \Delta_j > 0\}$ and let $b = |B|$. Renumber the coordinates such that $B = \{1, \ldots, b\}$ and $\Delta_1/x^l_1 \leqslant \ldots \leqslant \Delta_b/x^l_b$.
7: For $k \in \{0, \ldots, b\}$ define a vector y^k by $y^k_j = x^l_j$ for $j \leqslant k$ and $y^k_j = 0$ for $j > k$.
8: Change the left-hand side of (2.17) as follows: replace λ_l by $\lambda_l - \Delta_b/x^l_b$; for $1 \leqslant k < b$, increase the coefficient of y^k by $\Delta_{k+1}/x^l_{k+1} - \Delta_k/x^l_k$; and increase the coefficient of y^0 by Δ_1/x^l_1.
9: **end while**

construction may increase the size of the convex decomposition (= number of non-zero λ_l) 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 2.3.4. *Let $x^* \in \mathcal{Q}$ be dominated by a convex combination $\sum_{i\in\mathcal{N}} \lambda_l x^l$ of integral points in $\mathcal{Q}_\mathcal{I}$, i.e.,*

$$\sum_{i\in\mathcal{N}} \lambda_l x^l \geqslant x^*. \tag{2.17}$$

Then Algorithm 2 achieves equality in (2.17). 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 $\lambda_l x^l_j > 0$ and replacing x^l by $x^l - \mathbf{1}_j$ maintains feasibility, i.e., satisfies constraint (2.17), we perform this replacement. Note that x^l is an integer vector in $\mathcal{Q}_\mathcal{I}$, therefore $x^l - \mathbf{1}_j$ 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 $l \in \mathcal{N}$ and $j \in S^+$ with $\lambda_l x^l_j > 0$

$$\sum_{h\in\mathcal{N}} \lambda_h x^h_j - \lambda_l \mathbf{1}_j < x^*_j \tag{2.18}$$

We will establish (2.18) as an invariant of the second while-loop.
For $j \in S^+$, let $\Delta_j = \sum_{l\in\mathcal{N}} \lambda_l x^l_j - x^*_j$. Then $\Delta_j \geqslant 0$ and, by (2.18), for every $j \in S^+$ and $l \in \mathcal{N}$, with $\lambda_l \neq 0$ either $x^l_j = 0$ or $\Delta_j < \lambda_l \leqslant \lambda_l x^l_j$. If $\Delta_j = 0$ for all $j \in S^+$, we are done. Otherwise, choose j and $l \in \mathcal{N}$ such that $\Delta_j > 0$ and $x^l_j > 0$. Let

$B = \{j \in S^+ : x_j^l \neq 0 \text{ and } \Delta_j > 0\}$ be the indices in the support of x^l for which Δ_j is non-zero. We will change the left-hand side of (2.17) 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 $\Delta_1/x_1^l \leqslant \ldots \leqslant \Delta_b/x_b^l$. For $j \in [b]$, we clearly have

$$\lambda_l - \frac{\Delta_j}{x_j^l} = \lambda_l - \frac{\Delta_b}{x_b^l} + \frac{\Delta_b}{x_b^l} - \frac{\Delta_{b-1}}{x_{b-1}^l} + \ldots + \frac{\Delta_{j+1}}{x_{j+1}^l} - \frac{\Delta_j}{x_j^l}.$$

Multiplying by x_j^l and adding zero a few times, we obtain

$$\begin{aligned}\lambda_l x_j^l - \Delta_j = &\left(\lambda_l - \frac{\Delta_b}{x_b^l}\right) x_j^l + \sum_{k=j}^{b-1} \left(\frac{\Delta_{k+1}}{x_{k+1}^l} - \frac{\Delta_k}{x_k^l}\right) x_j^l \\ &+ \sum_{k=1}^{j-1} \left(\frac{\Delta_{k+1}}{x_{k+1}^l} - \frac{\Delta_k}{x_k^l}\right) 0 + \frac{\Delta_1}{x_1^l} 0.\end{aligned}$$

For $k \in \{0, \ldots, b-1\}$ define a vector y^k by $y_j^k = x_j^l$ for $j \leqslant k$ and $y_j^k = 0$ for $j > k$. Then $x_j^l = y_j^k$ for $k \geqslant j$ and $0 = y_j^k$ for $k < j$. Hence for all $j \leqslant b$

$$\begin{aligned}\lambda_l x_j^l - \Delta_j = &\left(\lambda_l - \frac{\Delta_b}{x_b^l}\right) x_j^l \\ &+ \sum_{k=1}^{b-1} \left(\frac{\Delta_{k+1}}{x_{k+1}^l} - \frac{\Delta_k}{x_k^l}\right) y_j^k + \frac{\Delta_1}{x_1^l} y_j^0. \qquad (2.19)\end{aligned}$$

Note that the coefficients on the right-hand side of (2.19) are non-negative and sum up to λ_l. Also, by the packing property of $\mathcal{Q}$, $y^k \in \mathcal{Q}_\mathcal{I}$ for $0 \leqslant k < b$. We now change the left-hand side of (2.17) as follows: we replace λ_l by $\lambda_l - \Delta_b/x_b^l$; for $1 \leqslant k < b$, we increase the coefficient of y^k by $\Delta_{k+1}/x_{k+1}^l - \Delta_k/x_k^l$; and we increase the coefficient of y^0 by Δ_1/x_1^l. 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 (2.18) holds for the vectors y^k, $0 \leqslant k < b$, that have a non-zero coefficient in the convex combination. Note first that $y_j^k > 0$ implies $j \in B$. Also (2.17) holds with equality for all $j \in B$. Thus (2.18) 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 $|S^+|$. □

2.3.4. Fast Convex Decomposition

We are now ready to prove Theorem 2.3.1.

Proof of Theorem 2.3.1. Theorem 2.3.3 yields a convex combination of integer points of $\mathcal{Q}_\mathcal{I}$ dominating $\alpha x^*/(1+4\varepsilon)$. Theorem 2.3.4 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. □

Algorithm 3. Packing($\mathcal{O}_\kappa$)

Require: a packing system (A, b, V) given by a κ-approximation-oracle $\mathcal{O}_\kappa$, where $A \in \mathbb{R}^{m\times n}_{\geqslant 0}$, $b \in \mathbb{R}^m_{>0}$, $V \in \mathbb{R}^n_{>0}$, $b > 0$, and an accuracy parameter $\varepsilon \in (0, 1)$
Ensure: A feasible solution $\hat{x} \in \mathbb{R}^n_{\geqslant 0}$ to (2.24) s.t. $V^T\hat{x} \geqslant \frac{1-4\varepsilon}{\kappa} z^*$
1: $x(0) := 0$; $t := 0$; and $T := \frac{\log m}{\varepsilon^2}$
2: **while** $M(t) < T$ **do**
3: $\quad t := t + 1$
4: $\quad$ Let $j(t) := \mathcal{O}_\kappa(p(t))$
5: $\quad x_{j(t)}(t) := x_{j(t)}(t-1) + \delta(t)$
6: **end while**
7: **return** $\hat{x} = \frac{x(t)}{M(t)}$

2.4. Approximately Truthful-in-Expectation 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 we show how to construct a randomized mechanism that satisfies the following assumptions for some $\varepsilon > 0$ and $\varepsilon_0 > 0$,

$$\text{No positive transfer.} \tag{2.20}$$
$$\text{Individually rational with probability } 1 - \varepsilon_0. \tag{2.21}$$
$$(1 - \varepsilon_0)\text{-truthful-in-expectation.} \tag{2.22}$$
$$\Upsilon\text{-social efficiency, where } \Upsilon \text{ depends on } \varepsilon_0 \text{ and } \varepsilon. \tag{2.23}$$

Our mechanism is based on constructing a randomized fractional mechanism and then converting the mechanism into an integral mechanism that satisfies in the above conditions. We state the main result of this section as the following theorem:

Theorem 2.4.1. *Let* $\varepsilon_0 \in (0, 1/2]$, $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$ *and* $\Upsilon = \alpha(1-\varepsilon)(1-\varepsilon_0)/(1+4\varepsilon)$. *Then we obtain a randomized integral mechanism satisfying Conditions (2.20) to (2.23).*

This section is structured as follows. In Subsection 2.4.1, we generalize the FPTAS given by Garg and Könemann to solve LP-relaxation of SWM. Then in Subsection 2.4.2, we apply the FPTAS and design a randomized fractional mechanism and finally in Subsection 2.4.3 we convert the fractional mechanism into a randomized integral mechanism and prove the main result of this section.

2.4.1. An FPTAS for Packing Linear Programs

In this subsection we present FPTAS $\mathcal{A}$ for a packing linear program (Algorithm 3) including LP-relaxation of SWM and show an upper bound for its runtime. Consider a packing linear program:

$$\begin{aligned} \max \quad & c^T x \\ s.t. \quad & Ax \leqslant b \\ & x \geqslant 0 \end{aligned} \tag{2.24}$$

where $A \in \mathbb{R}^{m\times n}_{\geqslant 0}$ is an $m \times n$ matrix with non-negative entries and $c \in \mathbb{R}^n_{>0}$, $b \in \mathbb{R}^m_{>0}$ are positive vectors. We may assume that each column of A contains a non-zero entry as otherwise the problem is trivially unbounded. For every $\kappa \geqslant 1$ and $z \in \mathbb{R}^m_{\geqslant 0}$, let $\mathcal{O}_\kappa(z)$ denote an κ-approximation oracle that returns a j such that

$$\frac{1}{c_j}\sum_{i=1}^{m}\frac{z_i a_{ij}}{b_i} \leqslant \kappa \cdot \min_{j'\in[n]}\frac{1}{c_{j'}}\sum_{i=1}^{m}\frac{z_i a_{ij'}}{b_i}. \tag{2.25}$$

It is easy to see that when $\kappa = 1$, the oracle returns a j minimizing $\frac{1}{c_{j'}}\sum_{i=1}^{m} z_i a_{ij'}/b_i$. Garg and Könemann assume the availability of $\mathcal{O}_1(z)$, $z \in \mathbb{R}^m_{\geqslant 0}$, and present an FPTAS. Here, we assume the availability of oracle $\mathcal{O}_\kappa$, for some $\kappa \geqslant 1$, and observe that their algorithm and proof also works with κ-approximation-oracles with $\kappa > 1$.

Theorem 2.4.2 (Extension of [GK98] for $\kappa > 1$)**.** *Let $T = \varepsilon^{-2}\log m$ and let z^* be value of the optimal solution to (2.24). Procedure Packing($\mathcal{O}_\kappa$) in Algorithm 3 terminates in at most mT iterations with a feasible solution $\hat{x}$ of at most mT positive components such that*

$$c^T\hat{x} \geqslant \frac{(1-3\varepsilon)}{\kappa}z^*. \tag{2.26}$$

It makes at most mT calls to the κ-approximation oracle $\mathcal{O}_\kappa$.

Proof. Consider Algorithm 3. It constructs a sequence $x(1)$, $x(2)$, …, $x(t_0)$. Let $M(t) = \max_{i\in[m]} A_i x(t)/b_i$, where A_i is the i-th row of A. For $i \in [m]$ let

$$p_i(t) := (1+\varepsilon)^{A_i x(t-1)/b_i}. \tag{2.27}$$

At each time t, the algorithm calls the oracle with the vector $z_t = p(t)$, and increases the variable $x_{j(t)}$ by $\delta(t) := \min_{i\in[m],a_{i,j(t)}\neq 0}\{\frac{b_i}{a_{i,j(t)}}\}$, where $j(t)$ is the index returned by the oracle. Note that there is always an i such that $a_{i,j(t)} > 0$ by our assumption on A. At the end, we scale $x(t)$ by $M(t)$ to guarantee feasibility. As in the proof of Theorem 2.3.2, we can show that for all $i \in [m]$

$$A_i x(t_0)/b_i \cdot \log(1+\varepsilon) \leqslant \log m + \varepsilon\sum_{t=1}^{t_0}\delta(t)\frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_{j(t)}. \tag{2.28}$$

We will relate the objective value $c^T x(t_0) = \sum_{t=1}^{t_0} c_{j(t)}\delta(t)$ at time t to the optimal value z^* by the following claim.

Claim 2. $\sum_{t=1}^{t_0}\delta(t)\frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_{j(t)} \leqslant \frac{\kappa\cdot c^T x(t_0)}{z^*}$.

Proof. Let $x^* \in \mathbb{R}^n_{\geqslant 0}$ be an optimal solution to (2.24). Then by feasibility of x^*, $B^{-1}Ax^* \leqslant \mathbf{1}$, thus we have for any t,

$$\frac{p(t)^T}{\|p(t)\|_1}B^{-1}Ax^* \leqslant 1.$$

By the choice of the index $j(t)$, we have that $\frac{1}{c_{j(t)}}\frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_{j(t)} \leqslant \frac{1}{c_j}\kappa\frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_j$ for all $j \in [n]$. Thus

$$\begin{aligned} z^*\frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_{j(t)} &= \sum_{j\in[n]} c_j x_j^* \frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_{j(t)} \\ &= \sum_{j\in[n]} c_j \frac{c_{j(t)}}{c_{j(t)}} x_j^* \kappa \frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_j \\ &\leqslant \sum_{j\in[n]} c_j \frac{c_{j(t)}}{c_j} x_j^* \kappa \frac{p(t)^T}{\|p(t)\|_1}B^{-1}A\mathbf{1}_j \\ &= \kappa c_{j(t)} \frac{p(t)^T}{\|p(t)\|_1}B^{-1}Ax^* \leqslant \kappa c_{j(t)}. \end{aligned}$$

Multiplying both sides of this inequality by $\delta(t)$ and summing up over $1 \leqslant t \leqslant t_0$ finishes the proof. □

Using the above claim, we can deduce from (3.3) that

$$A_i x(t_0)/b_i \cdot \log(1+\varepsilon) \leqslant \log m + \varepsilon \cdot \frac{\kappa \cdot c^T x(t_0)}{z^*} \quad \text{for all } i \in [m].$$

Dividing both sides by $M(t_0)$, arranging, and noting that at termination $M(t_0) \geqslant T$, we get that for all $i \in [m]$

$$\begin{aligned} \frac{\kappa \cdot c^T x(t_0))}{M(t_0)z^*} &\geqslant \frac{\log(1+\varepsilon)}{\varepsilon} \cdot \frac{A_i x(t_0)}{b_i M(t_0)} - \frac{\log m}{\varepsilon \cdot M(t_0)} \\ &\geqslant \frac{\log(1+\varepsilon)}{\varepsilon} \cdot \frac{A_i x(t_0)}{b_i M(t_0)} - \frac{\log m}{\varepsilon T}. \end{aligned}$$

In particular for the index $i_0 \in [m]$ such that $M(t_0) = A_{i_0}x(t_0)/b_{i_0}$, we have at termination

$$\frac{\kappa \cdot c^T \hat{x}}{z^*} = \frac{\kappa \cdot c^T x(t_0)}{M(t_0)z^*} \geqslant \frac{\log(1+\varepsilon)}{\varepsilon} - \frac{\log m}{\varepsilon T}.$$

Using $T = \frac{\log m}{\varepsilon^2}$, we finally get

$$\frac{\kappa \cdot c^T \hat{x}}{z^*} \geqslant \frac{\log(1+\varepsilon)}{\varepsilon} - \varepsilon \geqslant 1 - 3\varepsilon,$$

where the last inequality follows from the fact that $\frac{\log(1+\varepsilon)}{\varepsilon} \geqslant 1 - 2\varepsilon$, valid for all $\varepsilon \in (0, \frac{1}{2}]$.

The bound on the number of iterations follows from the fact that $\sum_i A_i x(t)$ grows by at least one in each iteration. □

Note that Theorem 2.4.2 can be extended to a certain class of packing convex programs, we will show this result in Theorem A.4.8.

2.4.2. Approximately Truthful-in-Expectation fractional Mechanisms

In this subsection we show how to construct a randomized fractional mechanism (Algorithm 4) and prove the following theorem:

Theorem 2.4.3. *Suppose that* $\varepsilon_0 \in (0, 1/2]$, $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$ *and* $\Upsilon = (1-\varepsilon)(1-\varepsilon_0)$. *Given an* ε*-approximation algorithm for* $\mathcal{Q}$*, Algorithm 4 defines a fractional randomized mechanism satisfying Conditions (2.20) to (2.23).*

In what follows we present two useful lemmas related to the mechanism and prove the main theorem. In order to present the mechanism, 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 $d = d_1 + \ldots + d_n$. [2] 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.,

$$v_i(u^i) = \max_{z \in \mathcal{Q}} v_i(z), \tag{2.29}$$

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

$$L_i := \sum_{j \neq i} v_j(u^j) \quad \text{and} \quad \beta_i := \varepsilon L_i. \tag{2.30}$$

Note that L_i does not depend on the valuation of player i. Let $\mathcal{A}$ be an FPTAS for the 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'). Note that x, $\bar{x}$, x' are fractional allocations. Now we bound the maximal change in social welfare induced by a change of the valuation of the i-th player.

Lemma 2.4.4. *Let* $\varepsilon \geqslant 0$ *and let* $\mathcal{A}$ *be an* ε*-approximation algorithm which returns allocation* x *on input vector* v*. Let* $\hat{x} \in \mathcal{Q}$ *be an arbitrary point, then*

$$v(x) \geqslant v(\hat{x}) - \beta_i - \varepsilon \cdot v_i(\hat{x}). \tag{2.31}$$

[2] 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 $x_{i,S}$.

Proof. We have

$$\begin{aligned} v(x) &\geqslant (1-\varepsilon)\max_{x\in\mathcal{Q}} v(x) \\ &\geqslant (1-\varepsilon)v(\hat{x}) \\ &= v(\hat{x}) - \varepsilon\cdot\sum_{j\neq i} v_j(\hat{x}) - \varepsilon\cdot v_i(\hat{x}) \\ &\geqslant v(\hat{x}) - \beta_i - \varepsilon\cdot v_i(\hat{x}), \end{aligned}$$

where the first inequality follows from the fact that $\mathcal{A}$ is an ε-approximation algorithm, and the last inequality follows from $\varepsilon\sum_{j\neq i} v_j(\hat{x}) \leqslant \varepsilon\sum_{j\neq i} v_j(u^j) = \beta_i$. □

We use the following payment rule:

$$p_i(v) := \max\{p_i^{VCG}(v) - \beta_i, 0\} \tag{2.32}$$

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 ($x = \mathcal{A}(v)$).

Let $U_i(v) = \bar{v}_i(x) - p_i(v)$ be the utility of player i for bid vector v. Note that the value of the allocation $x = \mathcal{A}(v)$ is evaluated with the true valuation $\bar{v}_i$ of player i. Consider $U_i(\bar{v}) = \bar{v}_i(\bar{x}) - p_i(\bar{v})$ be the utility of player i for valuation vector $\bar{v} = (\bar{v}_i, v_{-i})$.

Lemma 2.4.5. *Let $\varepsilon \geqslant 0$ and let $\mathcal{A}$ be an ε-approximation algorithms. Let M_0 be the mechanism with allocation function $\mathcal{A}(v)$ and the payment rule above. M_0 is an individually rational mechanism with no positive transfer, such that for all i,*

$$U_i(\bar{v}) \geqslant U_i(v) - \varepsilon\cdot\bar{v}_i(x) - 3\beta_i. \tag{2.33}$$

Proof. By definition, $p_i(v) \geqslant 0$ for all v and all x; so the mechanism has no positive transfer.

We next address individual rationality. We have

$$\begin{aligned} U_i(\bar{v}) &= \bar{v}_i(\bar{x}) - p_i(\bar{v}) \\ &= \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) + \beta_i \\ &= \bar{v}_i(\bar{x}) + \bar{v}_{-i}(\bar{x}) - \bar{v}_{-i}(x') + \beta_i \\ &= \bar{v}(\bar{x}) - \bar{v}(x') + \bar{v}_i(x') + \beta_i \\ &\geqslant (1-\varepsilon)\bar{v}_i(x') \geqslant 0, \end{aligned}$$

where the first inequality follows from Lemma 2.4.4 with $v = \bar{v}$ and $\hat{x} = x'$.

Finally, we address truthfulness. We have $v'(x') = v_{-i}(x')$, $v'(x) = v_{-i}(x)$, and $v'_i(x) = 0$. Thus

$$\begin{aligned} p_i^{VCG}(v) &= v_{-i}(x') - v_{-i}(x) \\ &= v'(x') - v'(x) + \varepsilon \cdot v'_i(x). \end{aligned}$$

Applying Lemma 2.4.4 for $v = v'$ and $\hat{x} = x$, we obtain

$$v'(x') - v'(x) + \varepsilon \cdot v'_i(x) \geqslant -\beta_i.$$

Therefore

$$p_i^{VCG}(v) + \beta_i \geqslant 0. \tag{2.34}$$

To see (2.33), we consider two cases:

Case 1: $p_i(v) = 0$. Then using (2.34)

$$\begin{aligned} U_i(\bar{v}) &= \bar{v}_i(\bar{x}) - 0 \\ &\geqslant \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i. \end{aligned}$$

Case 2: $p_i(v) = p_i^{VCG}(v) - \beta_i$.

$$\begin{aligned} U_i(\bar{v}) &= \bar{v}_i(\bar{x}) - p_i(\bar{v}) \\ &= \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) + \beta_i \\ &\geqslant \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i, \end{aligned}$$

where the last inequality follows from $\beta_i \geqslant 0$. Therefore, in both cases we have

$$U_i(\bar{v}) \geqslant \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i.$$

Now by using the definition of p_i^{VCG} and Lemma 2.4.4, we get

$$\begin{aligned} U_i(\bar{v}) &\geqslant \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i \\ &= \bar{v}_i(\bar{x}) + \bar{v}_{-i}(\bar{x}) - \bar{v}_{-i}(x') - \beta_i \\ &= \bar{v}(\bar{x}) - \bar{v}_{-i}(x') - \beta_i \\ &\geqslant \bar{v}(x) - \beta_i - \varepsilon\bar{v}_i(x) - \bar{v}_{-i}(x') - \beta_i \\ &= \bar{v}_i(x) - p_i^{VCG}(v) - \varepsilon\bar{v}_i(x) - 2\beta_i \\ &\geqslant \bar{v}_i(x) - p_i(v) - \beta_i - \varepsilon\bar{v}_i(x) - 2\beta_i \\ &= U_i(v) - \varepsilon\bar{v}_i(x) - 3\beta_i. \end{aligned}$$

□

Now we present Algorithm 4 that uses mechanism M_0 to give a fractional randomized mechanism which is approximately truthful-in-expectation.

Algorithm 4. The mechanism M of Theorem 2.4.3. The vectors u^i are defined as in (2.29) and the quantities L_i are defined in (2.30). The definitions of q_0, q_j, active and inactive players are given in the proof of Theorem 2.4.3.

Require: A valuation vector $v \in \mathcal{V}$, 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 proof of Theorem 2.4.3.
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 (2.32). For all inactive i, change x_i and p_i to zero.
5: **else**
6: For every $1 \leqslant i \leqslant n$, set

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

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

Proof of Theorem 2.4.3. Define $q_0 = (1 - \frac{\varepsilon_0}{n})^n$, $\bar{\varepsilon} = \varepsilon_0/2$, and $q_j = (1 - q_0)/n$ for $1 \leqslant j \leqslant n$. Let $\gamma = \bar{\varepsilon}(1 - q_0)^2/n^3$, $\gamma' = \gamma/q_j$, and $\varepsilon = \gamma\bar{\varepsilon}(1 - q_0)/(8n)$. Then using $q_0 = (1 - \frac{\varepsilon_0}{n})^n \geqslant 1 - \varepsilon_0$ and $q_0 = (1 - \frac{\varepsilon_0}{n})^n \leqslant 1 - \varepsilon_0/2$, we get

$$\frac{\varepsilon_0^5}{128n^4} = \frac{\bar{\varepsilon}^2(\varepsilon_0/2)^3}{8n^4} \leqslant \varepsilon = \gamma\bar{\varepsilon}(1 - q_0)/(8n) = \frac{\bar{\varepsilon}^2(1 - q_0)^3}{8n^4} \leqslant \frac{\bar{\varepsilon}^2\varepsilon_0^3}{8n^4} = \frac{\varepsilon_0^5}{16n^4},$$

as stated in the Theorem. Let $U_i(v)$ be the utility of player i obtained by the mechanism M_0 of Lemma 2.4.5. Following [DRVY11], we call player i *active* if the following two conditions hold:

$$U_i(v) + \frac{\bar{\varepsilon} q_i}{q_0} v_i(u^i) \geqslant \frac{q_i}{q_0}\gamma' L_i, \tag{2.35}$$

$$v_i(u^i) \geqslant \gamma L_i. \tag{2.36}$$

We denote by $T = T(v)$ the set of active players when the valuation is $v = (v_1, \ldots, v_n)$. Note that L_i does not depend on v_i. Thus when we refer to conditions (2.35) and (2.36) for $\bar{v}$, we replace v and x by $\bar{v}$ and $\bar{x}$ on the left side and keep the right side unchanged. Non-negativity of payments is immediate from the definition of mechanism M and Lemma 2.4.5. Moreover, the utility of a truth-telling bidder i can be negative only if it is allocated in step 5, i.e., at most with probability q_i. It follows that the mechanism is individually rational with probability at least $1 - \sum_{i=1}^n q_i = q_0 = (1 - \frac{\varepsilon_0}{n})^n \geqslant 1 - \varepsilon_0$.

Now we address truthfulness. Let us denote the expected utility of player i obtained from the mechanism in Algorithm 4 on input $v \in \mathcal{V}$ by $\mathbb{E}[U_i'(v)]$. Assume $j = 0$

in Algorithm 4. 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. Let x' be the allocation obtained in this way. The value for player i is $v_i(x')$. When the i-th player is active, this value is equal to $v_i(x)$ because v_i depends only on the valuation in the i-th group (separability property). Therefore in this case his utility is $U_i(v)$. So we have that

$$\mathbb{E}[U_i'(v)] = \begin{cases} q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \gamma' L_i) & \text{if } i \in T(v), \\ q_i \bar{v}_i(u^i) & \text{if } i \notin T(v). \end{cases} \tag{2.37}$$

We first observe

$$\mathbb{E}[U_i'(\bar{v})] \geqslant (1 - \bar{\varepsilon}) q_i \cdot \bar{v}_i(u^i). \tag{2.38}$$

Indeed, the inequality is trivially satisfied if $i \notin T(\bar{v})$. On the other hand, if $i \in T(\bar{v})$, then (2.35) implies $U_i(\bar{v}) \geqslant \frac{q_i}{q_0}\left(\gamma' L_i - \bar{\varepsilon}\bar{v}_i(u^i)\right)$, therefore

$$\begin{aligned} \mathbb{E}[U_i'(\bar{v})] &= q_0 \cdot U_i(\bar{v}) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\ &\geqslant q_0 \cdot \frac{q_i}{q_0}\left(\gamma' L_i - \bar{\varepsilon}\bar{v}_i(u^i)\right) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\ &= (1 - \bar{\varepsilon}) q_i \cdot \bar{v}_i(u^i). \end{aligned}$$

We now consider four cases:

Case 1: $i \in T(\bar{v}) \cap T(v)$. Note that (2.36) for $\bar{v}$ implies $\beta_i = \varepsilon L_i \leqslant \frac{\varepsilon \bar{v}_i(u^i)}{\gamma}$. Thus, by Lemma 2.4.5, and using assumption (2.29) that $\bar{v}_i(x) \leqslant \bar{v}_i(u^i)$, we have

$$U_i(\bar{v}) \geqslant U_i(v) - \varepsilon(1 + \frac{3}{\gamma})\bar{v}_i(u^i) \geqslant U_i(v) - \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i). \tag{2.39}$$

Hence by using (2.37) and (2.39), we have

$$\begin{aligned} \mathbb{E}[U_i'(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\ &\leqslant q_0(U_i(\bar{v}) + \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i)) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\ &= \underbrace{q_0 U_i(\bar{v}) + q_i(\bar{v}_i(u^i) - \gamma' L_i)}_{\mathbb{E}[U_i'(\bar{v})]} + \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i) \\ &= \mathbb{E}[U_i'(\bar{v})] + q_0 \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i). \end{aligned}$$

Now applying (2.38) in above inequality, we get

$$\begin{aligned} \mathbb{E}[U_i'(v)] &\leqslant \mathbb{E}[U_i'(\bar{v})] + q_0 \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i) \\ &\leqslant \left(1 + \frac{q_0}{(1 - \bar{\varepsilon}) q_i} \frac{4\varepsilon}{\gamma}\right) \mathbb{E}[U_i'(\bar{v})] \\ &\leqslant (1 + \bar{\varepsilon})\mathbb{E}[U_i'(\bar{v})], \end{aligned}$$

where the last inequality follows from the definition of ε. Note that (since $q_0 \leqslant 1$ and $\bar{\varepsilon} \leqslant 1/2$)

$$\varepsilon \frac{q_0}{(1 - \bar{\varepsilon}) q_i} \frac{4}{\gamma} \leqslant \varepsilon \frac{1}{(1 - \bar{\varepsilon}) q_i} \frac{4}{\gamma} \leqslant \frac{\gamma \bar{\varepsilon}(1 - q_0)}{8n} \frac{8}{q_i \gamma} = \bar{\varepsilon}.$$

Case 2: $i \notin T(v)$. By (2.38), we have

$$\mathbb{E}[U_i'(v)] = q_i \bar{v}_i(u^i) \leqslant \frac{1}{1-\bar{\varepsilon}} \mathbb{E}[U_i'(\bar{v})] \leqslant (1+\varepsilon_0)\mathbb{E}[U_i'(\bar{v})].$$

Since,

$$\frac{1}{1-\bar{\varepsilon}} = 1 + \bar{\varepsilon}(1 + \bar{\varepsilon} + \bar{\varepsilon}^2 + \ldots) \leqslant 1 + 2\bar{\varepsilon} = 1 + \varepsilon_0.$$

Case 3: $i \in T(v) \setminus T(\bar{v})$ and (2.36) does not hold for $\bar{v}$. Since $U_i(v) \leqslant \bar{v}_i(u^i)$, we have

$$\begin{aligned}
\mathbb{E}[U_i'(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\
&\leqslant (q_0 + q_i)\bar{v}_i(u^i) - q_i \gamma' L_i \\
&< (q_0 + q_i - 1)\bar{v}_i(u^i) \\
&\leqslant 0 \\
&\leqslant q_i \bar{v}_i(u^i) \\
&= \mathbb{E}[U_i'(\bar{v})],
\end{aligned}$$

where the second inequality holds because (2.36) does not hold for $\bar{v}$ and $q_i\gamma'/\gamma = 1$. *Case 4:* $i \in T(v) \setminus T(\bar{v})$ and (2.36) holds for $\bar{v}$. Then (2.35) does not hold for $\bar{v}$ and hence

$$U_i(\bar{v}) < \frac{q_i}{q_0}\left(\gamma' L_i - \bar{\varepsilon}\bar{v}_i(u^i)\right). \tag{2.40}$$

Since (2.36) holds for $\bar{v}$, we have (2.39). Hence by (2.38), (2.39) and (2.40) we have

$$\begin{aligned}
\mathbb{E}[U_i'(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\
&\leqslant q_0 \left(U_i(\bar{v}) + \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i) \right) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\
&\leqslant q_i \gamma' L_i - q_i \bar{\varepsilon}\bar{v}_i(u^i) + \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i) + q_i(\bar{v}_i(u^i) - \gamma' L_i) \\
&= (1-\bar{\varepsilon})q_i \cdot \bar{v}_i(u^i) + \frac{4\varepsilon}{\gamma}\bar{v}_i(u^i) \\
&\leqslant \left(1 + \frac{1}{(1-\bar{\varepsilon})q_i}\frac{4\varepsilon}{\gamma}\right)\mathbb{E}[U_i'(\bar{v})] \\
&\leqslant (1+\bar{\varepsilon})\mathbb{E}[U_i'(\bar{v})],
\end{aligned}$$

where the last inequality follows from the definition of ε (see Case 1). We finally argue about the approximation ratio. Note that for $i \notin T(v)$, one of the inequalities (2.35) or (2.36) does not hold. Also, $U_i(v) \geqslant 0$ by the individual rationality of the mechanism M_0 and hence $v_i(u^i) < \max\{\gamma, \gamma'/\bar{\varepsilon}\}L_i = \gamma' L_i/\bar{\varepsilon}$. Since $\mathcal{A}$ returns allocation x that is

$(1-\varepsilon)$-social efficiency and [3] $q_i - q_0 n\frac{\gamma'}{\bar{\varepsilon}} \geqslant 0$, it follows that for any $v \in \mathcal{V}$,

$$\begin{aligned}
\mathbb{E}[v(x)] &= q_0 \sum_{i\in T(v), x=\mathcal{A}(v)} v_i(x) + \sum_{i\in[n]} q_i v_i(u^i) \\
&\geqslant q_0 \sum_{i\in[n], x=\mathcal{A}(v)} v_i(x) - q_0 \sum_{i\notin T(v), x=\mathcal{A}(v)} v_i(x) + \sum_{i\in[n]} q_i v_i(u^i) \\
&\geqslant q_0 v(x) - q_0 \sum_{i\notin T(v)} v_i(u^i) + \sum_{i\in[n]} q_i v_i(u^i) \\
&\geqslant q_0 v(x) - q_0 \frac{\gamma'}{\bar{\varepsilon}} \sum_{i\notin T(v)} L_i + \sum_{i\in[n]} q_i v_i(u^i) \\
&= q_0 v(x) - q_0 \frac{\gamma'}{\bar{\varepsilon}} \sum_{i\notin T(v)} \sum_{j\neq i} v_j(u^j) + \sum_{i\in[n]} q_i v_i(u^i) \\
&\geqslant q_0 v(x) - q_0 \frac{\gamma'}{\bar{\varepsilon}} n \sum_{j\neq i} v_j(u^j) + \sum_{i\in[n]} q_i v_i(u^i) \\
&\geqslant q_0 v(x) + \sum_{i\in[n]} \left(q_i - q_0 n \frac{\gamma'}{\bar{\varepsilon}}\right) v_i(u^i) \\
&\geqslant q_0(1-\varepsilon)\cdot \max_{z\in Q} v(z) \\
&\geqslant (1-\varepsilon_0)(1-\varepsilon)\cdot \max_{z\in Q} v(z).
\end{aligned}$$

□

2.4.3. Proof of Theorem 2.4.1

In this subsection we prove the main result of this section. Let us define randomized mechanism M' which returns an integral allocation. Let $\varepsilon > 0$ be arbitrary. First run Algorithm 4 to obtain x and $p(v)$. Then compute a convex decomposition of $\frac{\alpha}{1+4\varepsilon}x$, i.e.,

$$\frac{\alpha}{1+4\varepsilon}x = \sum_{l\in\mathcal{N}} \lambda_l^x x^l,$$

and finally with probability λ_l^x (we used the superscript to distinguish the convex decompositions of x) return the allocation x^l and charge i-th player, the price $p_i(v)\frac{v_i(x^l)}{v_i(x)}$, if $v_i(x) > 0$, and zero otherwise.

Proof of Theorem 2.4.1. Let M be a fractional randomized mechanism obtained in Theorem 2.4.3. Since M has no positive transfer, M' does neither. M is individually rational with probability $1-\varepsilon_0$, therefore for any allocation $\bar{x}$, we have $\bar{v}_i(\bar{x}) - p_i(\bar{v}) \geqslant 0$ with probability $1-\varepsilon_0$. So

$$\bar{v}_i(x^l) - p_i(\bar{v})\frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})} = (\bar{v}_i(\bar{x}) - p_i(\bar{v}))\frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})} \geqslant 0,$$

[3] $q_0 n\frac{\gamma'}{\bar{\varepsilon}} \leqslant \frac{n\gamma}{q_i\bar{\varepsilon}} = \frac{n^2\bar{\varepsilon}(1-q_0)^2}{(1-q_0)\bar{\varepsilon}n^3} = \frac{1-q_0}{n} = q_i$.

hence M' is individually rational with probability $1-\varepsilon_0$. Now we prove truthfulness. Let $\mathbb{E}[U_i^{''}(\bar{v})]$ be the expected utility of player i when she inputs her true valuation and let $\mathbb{E}[U_i^{''}(v)]$ be her expected utility when she inputs v_i. Then by definition of $\mathbb{E}[U_i^{''}(\bar{v})]$, we have

$$\begin{aligned}
\mathbb{E}[U_i^{''}(\bar{v})] &= \mathbb{E}_{\bar{x}\sim M(\bar{v})}\left[\sum_{l\in\mathcal{N}}\lambda_l^{\bar{x}}\left(\bar{v}_i(x^l)-p_i(\bar{v})\frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})}\right)\right]\\
&= \mathbb{E}_{\bar{x}\sim M(\bar{v})}\left[\left(\bar{v}_i(\sum_{l\in\mathcal{N}}\lambda_l^{\bar{x}}x^l)-p_i(\bar{v})\frac{\bar{v}_i(\sum_{l\in\mathcal{N}}\lambda_l^{\bar{x}}x^l)}{\bar{v}_i(\bar{x})}\right)\right]\\
&= \mathbb{E}_{\bar{x}\sim M(\bar{v})}[\frac{\alpha}{1+4\varepsilon}\bar{v}_i(\bar{x})-\frac{\alpha}{1+4\varepsilon}p_i(\bar{v})]\\
&= \frac{\alpha}{1+4\varepsilon}\mathbb{E}_{\bar{x}\sim M(\bar{v})}[\bar{v}_i(\bar{x})-p_i(\bar{v})]\\
&= \frac{\alpha}{1+4\varepsilon}\mathbb{E}[U'(\bar{v})]\\
&\geqslant (1-\varepsilon_0)\frac{\alpha}{1+4\varepsilon}\mathbb{E}[U'(v)]\\
&= (1-\varepsilon_0)\frac{\alpha}{1+4\varepsilon}\mathbb{E}_{x\sim M(v)}[\bar{v}(x)-p_i(v)]\\
&= (1-\varepsilon_0)\mathbb{E}_{x\sim M(v)}[\frac{\alpha}{1+4\varepsilon}\bar{v}(x)-p_i(v)\frac{\alpha}{1+4\varepsilon}\cdot\frac{v_i(x)}{v_i(x)}]\\
&= (1-\varepsilon_0)\mathbb{E}\left[\sum_{l\in\mathcal{N}}\lambda_l^{x}\left(\bar{v}_i(x^l)-p_i(v)\frac{v_i(x^l)}{v_i(x)}\right)\right]\\
&= (1-\varepsilon_0)\mathbb{E}[U_i^{'}(v)].
\end{aligned}$$

Taking expectation with respect to x shows that the mechanism is $\frac{\alpha(1-\varepsilon_0)(1-\varepsilon)}{1+4\varepsilon}$-socially efficient. This completes the proof of Theorem 2.4.1.

$$\begin{aligned}
\mathbb{E}[v(x)] &= \mathbb{E}_{x\sim M(v)}\left[\sum_{l\in\mathcal{N}}\lambda_l^{x}v(x^l)\right]\\
&= \mathbb{E}_{x\sim M(v)}\left[v(\sum_{l\in\mathcal{N}}\lambda_l^{x}x^l)\right]\\
&= \mathbb{E}_{x\sim M(v)}\left[v(\frac{\alpha}{1+4\varepsilon}x)\right]\\
&= \frac{\alpha}{1+4\varepsilon}\mathbb{E}_{x\sim M(v)}[v(x)]\\
&\geqslant \frac{\alpha}{1+4\varepsilon}(1-\varepsilon_0)(1-\varepsilon)
\end{aligned}$$

□

3

Randomized Fictitious Play for Approximating Saddle Points

Approximating saddle points is an overarching and fundamental problem that has a wide range of applications, from economics and game theory to dynamical systems, optimization, and analytic combinatorics, for example. In this chapter, we mainly focus on saddle points from a game theoretical point of view. Let us consider a particular class of games called matrix games. In this game, there are two players: one, called row player, selects a row, say $i \in [m]$, and the other called column player, selects a column, say $j \in [n]$, of a given matrix $A \in \mathbb{R}^{m \times n}$, and each is unaware of the choice of the other. Then, their choices are made known and the row player pays the amount of $A(i, j)$ to the column player, which is the (i, j)-th entry of A. Assuming the players are rational, the row player wants to minimize his loss, and the column player wants to maximize his gain. In such a game, entry $A(i^*, j^*)$ is a *saddle point* when it is a minimum number in the i^*-th row, and it is simultaneously a maximum number in the j^*-th column. It is readily checked that if A contains a saddle point, then the game has an *equilibrium*, because no player increases his benefit by changing his chosen column (or row). The saddle points can also be considered in a larger class of games called the 2-player zero-sum game in which the payoff function is specified by a function $F : X \times Y \to \mathbb{R}$, where X and Y are *strategy sets* for the players and for a chosen strategy $(x, y) \in X \times Y$, and the minimizer pays amount $F(x, y)$ to the maximizer. Pair (x^*, y^*) is called a saddle point if

$$F(x^*, y^*) = \inf_{x \in X} F(x, y^*) = \sup_{y \in Y} F(x^*, y).$$

Since the class of 2-player zero-sum games with a specified payoff function is a quite general framework, it has received considerable attention. For instance, in social and economical systems, there are plenty of situations where two decision makers interact with each other. Understanding and predicting the behavior of such systems can

usually be reduced to finding a pair of *optimal strategies* (or saddle points) for a corresponding 2-player zero-sum game. It turns out that computing saddle points is a long-standing problem from both the theoretical and the practical perspective, and hence, there is extensive literature on the existence of saddle points in this class of games and its applications, see, e.g., [Dan63, Gro67, tKP90, McL84, Vor84, Roc70, Sha58, Ter72, Wal45, Seb90, Bel97, DKR91, Was03]. In the 50s, Brown [Bro51] introduced an iterative procedure, so-called *fictitious play*, for computing the equilibrium of a 2-player zero-sum game, with a payoff function $F(x, y) = x^T Ay$, where $A \in \mathbb{R}^{m\times n}$, and the strategy sets are probability distributions over rows and columns, known as *mixed strategies*. The procedure proceeds in rounds. In each round: each player updates his strategy by applying the best response strategy to the current opponent's strategy. (See Section 3.1 for more details.) Robinson [Rob51] showed that if the number of rounds goes to infinity, then the fictitious play converges to the optimal strategies for players. Very recently, Daskalakis and Pan [DP14] showed an exponential lower bound for the number of iterations in which the fictitious play converges to an equilibrium.

For every $\varepsilon \in (0, 1)$, point $(x^*, y^*) \in X \times Y$ is an ε-approximation saddle point (or equivalently, an *ε-optimal strategy*) of function $F : X \times Y \to \mathbb{R}$ if we have

$$\sup_{y\in Y} F(x^*, y) \leqslant \inf_{x\in X} F(x, y^*) + \varepsilon.$$

Based on fictitious play, Grigoriadis and Khachiyan [GK95] presented a randomized algorithm, so-called *randomized fictitious play* for computing the *ε-optimal strategies* for the 2-player zero-sum games with a payoff function $x^T Ay$, where $A \in [-1, 1]^{m\times n}$. Generally speaking, in randomized fictitious play, each player chooses the response update randomly from a distribution that gives large weight to good responses, but concentrates not only on the best response. They also proved an expected upper bound $O((m + n) \log(n + m)/\varepsilon^2)$ for the running time of the algorithm.

When the strategy sets are convex sets, one can easily see that computing the ε-approximation saddle point (or ε-optimal strategies) can be reformulated as a convex minimization problem over a convex set, and hence, any algorithm for solving this class of problems, e.g., the Ellipsoid method, can be used to compute them in time polynomial in the input size and polylog$(\frac{1}{\varepsilon})$ (see, e.g., [GLS93]).

Our Results. Our main contribution is the design of a randomized algorithm for computing an ε-approximation saddle point of convex-concave functions over the product of two convex bounded sets. Our algorithm is based on combining a technique developed by Grigoriadis and Khachiyan [GK95], which is a randomized variant of Brown's fictitious play [Bro51], with the recent results on random sampling from convex sets (see, e.g., [LV06, Vem05]). The algorithm finds an ε-approximation saddle point in an expected number of $O\left(\frac{\rho^2(n+m)}{\varepsilon^2} \log \frac{R}{\varepsilon}\right)$ iterations, where in each iteration, two points are sampled from log-concave distributions over strategy sets. In particular, the algorithm requires $O^*\left(\frac{\rho^2(n+m)^6}{\varepsilon^2} \log R\right)$ oracle calls, where $O^*(\cdot)$ suppresses polylogarithmic factors that depend on n, m, and ε.

Even though the bounds for the running time of the algorithm are stated for general

convex sets, note that these bounds may be improved for classes of convex sets for which faster sampling procedures could be developed.

Our algorithm is superior to known methods when the width parameter ρ is small, where $\rho = \max_{x\in X, y\in Y} |F(x,y)|$ and $\varepsilon \in (0,1)$ is a fixed but arbitrarily small constant; see the comparison with sampling-based algorithms in Section 3.2. We believe that our method presented in this paper will be useful for developing algorithms for computing approximate equilibria for other classes of games.

Outline. In Section 3.1, we formally define the saddle point and approximation saddle point and state some preliminary results. In Section 3.2, we briefly explain the results that are most closely related to our work. In Section 3.3, we formally state our results and the algorithm. In Section 3.4, we show its correctness and analyze the runtime of the algorithm. Finally, in Section 3.5, we present some applications of our algorithm in combinatorial optimization.

3.1. Definitions, Notations, and Preliminaries

In this section, we formally define the saddle point and ε-approximation saddle point, some classical definitions in game theory, and the original fictitious play. Then, we state some notation and technical assumptions.

Definition 3.1.1. *For a given function $F : X \times Y \to \mathbb{R}$, pair (x^*, y^*) is a saddle point if we have*

$$F(x^*, y^*) = \inf_{x\in X} \sup_{y\in Y} F(x,y) = \sup_{y\in Y} \inf_{x\in X} F(x,y). \tag{3.1}$$

Furthermore, for every $\varepsilon \in (0,1)$, pair (x^*, y^*) is an ε-approximation saddle point if we have

$$\sup_{y\in Y} F(x^*, y) \leqslant \inf_{x\in X} F(x, y^*) + \varepsilon. \tag{3.2}$$

Let $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ be a real-valued function, where X is a convex set. We say f is a *convex function* if for every $t \in [0,1]$ and $x, y \in X$ we have

$$f(t \cdot x + (1-t) \cdot y) \leqslant tf(x) + (1-t)f(y).$$

And, f is a *concave function* if we have

$$f(t \cdot x + (1-t) \cdot y) \geqslant tf(x) + (1-t)f(y).$$

We have that f is a log-concave if $\log f$ is concave. Let $X \subseteq \mathbb{R}^m$ and $Y \subseteq \mathbb{R}^n$ be two bounded convex sets. Moreover, we say that $F : X \times Y \to \mathbb{R}$ is a *convex-concave function*, that is, $F(\cdot, y) : X \to \mathbb{R}$ is convex for all $y \in Y$, and $F(x, \cdot) : Y \to \mathbb{R}$ is concave for all $x \in X$. The following theorem, known as Saddle-Point Theorem, guarantees the existence of the saddle points (see, e.g., [Roc70]).

Theorem 3.1.2 (Saddle-Point Theorem)**.** *Suppose that $F : X \times Y \to \mathbb{R}$ is a continuous convex-concave function, where X and Y are closed convex sets. Then a saddle point for F exists.*

A *2-player zero-sum game* is an important class of games, in which the gain of one player is exactly the loss of other player. Now, let us consider a 2-player zero-sum game with payoff function $F(\cdot,\cdot)$ where one player, the minimizer, choosing his strategy from a convex domain X, while the other player, the maximizer, choosing his strategy from a convex domain Y. For a pair of strategies $x \in X$ and $y \in Y$, $F(x,y)$ denotes the corresponding payoff, which is the amount that the minimizer pays to the maximizer. We note that sets X and Y are called the strategy sets. If both strategy sets are finite, then the game is called finite and hence the payoff function can be represented as a matrix, let say $A \in \mathbb{R}^{m\times n}$, in which each entry corresponds to the payoff of a pair of strategies. In this setting the minimizer, row player, picks a row, say $i \in [m]$, and the maximizer, column player, picks a column, say $j \in [n]$. Then the minimizer pays amount $A(i,j)$ to the maximizer. We refer to elements of X and Y as pure strategies. In a more general setting, selecting the pure strategies can be randomized, which means each player pick his own strategy according to a probability distribution over his pure strategy set, called *mixed strategy*. In a matrix game with payoff matrix $A \in \mathbb{R}^{m\times n}$, if the row and column players select mixed strategies x and y, respectively, which are probability distributions over rows and columns, then the *expected payoff* is computed as $x^T Ay$, where x^T denotes the transpose of vector x.

Definition 3.1.3. *An* equilibrium point*, say (x^*, y^*), is a pair of strategies in which no player increases his utility, by changing his strategy assuming the opponent does not change his strategy.*

Clearly, every saddle point for F corresponds to a equilibrium point for the game. Let us assume that (x^*, y^*) is a saddle point. Then the *value* of the game is defined as $v^* = F(x^*, y^*)$. Now, in what follows we define the *optimal strategies*.

Definition 3.1.4. *Suppose that v^* be the value a 2-player zero-sum game with payoff function $F : X \times Y \to \mathbb{R}$. Then $x^* \in X$ is an optimal strategy for minimizer if and only if*

$$F(x^*, y) \leqslant v^*, \text{ for every } y \in Y.$$

Also $y^* \in Y$ is an optimal strategy for the maximizer if and only if

$$F(x, y^*) \geqslant v^*, \text{ for every } x \in X.$$

*When both players apply the optimal strategies, then the minimizer pays v^* to maximizer. When an approximate solution suffices or at least one of the sets X or Y is open, the appropriate notion is that of ε-*optimal strategies*. We have that $x^* \in X$ is an ε-optimal strategy for the minimizer if and only if*

$$F(x^*, y) \leqslant v^* + \varepsilon, \text{ for every } y \in Y.$$

Also $y^* \in Y$ is an ε-optimal strategy for the maximizer if and only if

$$F(x, y^*) \geqslant v^* - \varepsilon, \text{ for every } x \in X.$$

The following famous theorem due to Von Neumann states the existence of a saddle point in matrix games.

Theorem 3.1.5 (Min-Max Theorem)**.** *Let A be a given $m \times n$ matrix over real numbers. Then there is a pair of mixed strategies (x^*, y^*) forming a saddle point:*

$$\min_{x \in X} \max_{y \in Y} x^T A y = \max_{y \in Y} \min_{x \in X} x^T A y = x^{*T} A y^*.$$

3.1.1. Technical Assumptions

In this subsection we present some general conditions for function F and sets $X \subset \mathbb{R}^m$ and $Y \subset \mathbb{R}^n$. Throughout, we assume that sets X and Y are specified by a membership oracle, that is an algorithm which for a given $x \in \mathbb{R}^m$ (respectively, $y \in \mathbb{R}^n$) determines, in polynomial time in m (respectively, n), whether or not $x \in X$ (respectively, $y \in Y$).

We define the width parameter of our problem ρ as

$$\rho = \max_{x \in X, y \in Y} |F(x, y)|.$$

Let $B^k(x^0, r)$ denote the k-dimensional ball for radius r centered at $x^0 \in \mathbb{R}^k$:

$$B^k(x^0, r) = \{x \in \mathbb{R}^k : \ \|x - x^0\|_2 \leqslant r\}.$$

We now state two technical assumptions regarding the function F and sets X and Y.

Assumption A1. There exist $\xi^0 \in X$, $\eta^0 \in Y$, and strictly positive numbers r_X, R_X, r_Y, and R_Y such that

$$B^m(\xi^0, r_X) \subseteq X \subseteq B^m(\mathbf{0}, R_X)$$
$$B^n(\eta^0, r_Y) \subseteq Y \subseteq B^n(\mathbf{0}, R_Y).$$

In particular, since the sets contain the respective balls, they are full-dimensional in their respective spaces. With regard to this assumption, we define parameter R as follows

$$R = \max\{R_X, R_Y, \frac{1}{r_X}, \frac{1}{r_Y}\}.$$

Assumption A2. Width parameter ρ corresponding to F is bounded by 1.

Assumption (A1) is standard for algorithms that deal with convex sets defined by membership oracles (see, e.g., [GLS93]), and will be required by the sampling algorithms. Assumption (A2) can be made without loss of generality, since the original game can be converted to an equivalent one satisfying (A2) by scaling the function F by $\frac{1}{\rho}$. However, in order to get an ε-approximation saddle point, we also need to replace ε by $\frac{\varepsilon}{\rho}$. We note that such dependence on the width is unavoidable in most known algorithms that obtain ε-approximate solutions and whose running time is proportional to $poly(\frac{1}{\varepsilon})$ (see e.g. [AHK06, PST91]). In Theorem 3.3.1 we show that the running time of our algorithm increases by factor ρ^2. We assume that ε is a positive constant less than 1. In case the payoff function is bilinear, then we can prove the following upper bound for ρ.

Proposition 3.1.6. *If $F(x,y) = x^T Ay$, where A is a given $m \times n$ matrix and x^T denotes the transpose of vector x, then*

$$\rho \leqslant \sqrt{m \cdot n} \cdot R_X \cdot R_Y \cdot \max_{i,j} |a_{ij}|.$$

Proof. Let A_i denote the i-th row of matrix A. By applying Cauchy-Schwarz inequality we have that

$$\begin{aligned} ||Ay|| &= \left[\sum_{i=1}^{m} (A_i y)^2\right]^{1/2} \\ &\leqslant \left[\sum_{i=1}^{m} ||A_i||^2 \cdot ||y||^2\right]^{1/2} \\ &\leqslant \sqrt{m \cdot n \cdot (\max_{i,j} |a_{ij}|)^2 \cdot R_Y^2}. \end{aligned}$$

Another application of Cauchy-Schwarz inequality gives that for every $(x,y) \in X \times Y$,

$$|x^T Ay| \leqslant ||x^T|| \cdot ||Ay|| \leqslant \sqrt{m \cdot n} \cdot R_X \cdot R_Y \cdot \max_{i,j} |a_{ij}|.$$

Therefore, ρ is bounded as above. □

3.2. Relation to Previous Works

As we already mentioned, computing the ε-approximation saddle point is a quite general task and hence there are a large number of works that consider the problem. We devote this section to explain some of them which are mostly related and compare them with our result.

Fictitious Play. Fictitious play refers to an earliest learning rule, originally proposed by Brown [Bro51], for computing the optimal strategies of a matrix game. In this method each player updates his strategy by applying the best response, given the opponent's current strategy. More precisely, the minimizer and the maximizer initialize, respectively, $x(0) = 0$ and $y(0) = 0$, and for $t = 1, 2, \ldots$, update $x(t)$ and $y(t)$ by

$$x(t+1) = \frac{t}{t+1} x(t) + \frac{1}{t+1} e_{i(t)}, \text{ where } i(t) = \operatorname{argmin}_{i \in [m]} e_i{}^T Ay(t), \quad (3.3)$$

$$y(t+1) = \frac{t}{t+1} y(t) + \frac{1}{t+1} e_{j(t)}, \text{ where } j(t) = \operatorname{argmax}_{j \in [n]} x(t)^T Ae_j, \quad (3.4)$$

where $e_j \in \mathbb{R}^n$ ($e_i \in \mathbb{R}^m$) denotes the unit vector whose j-th (i-th) coordinate is 1. Robinson [Rob51] showed the convergence of fictitious play to the optimal strategies, that is,

$$\begin{aligned} x^* &= \lim_{t \to \infty} x(t), \\ y^* &= \lim_{t \to \infty} y(t). \end{aligned}$$

Shapiro [Sha58] derived a bound of $(2^{m+n}/\varepsilon)^{m+n-2}$ on the number of rounds needed for convergence of fictitious play to an ε-optimal strategies (or ε-approximation saddle point). Very recently, Daskalakis and Pan [DP14] obtained an exponential lower bound for fictitious play. One might notice that in the original matrix game, the players pick their own strategies, known as *mixed strategies*, from simplices X and Y, that are,

$$X = \{x \in \mathbb{R}^m \mid \sum_i x_i = 1, x \geqslant 0\}$$

and

$$Y = \{y \in \mathbb{R}^n \mid \sum_j y_j = 1, y \geqslant 0\}.$$

In a more general setting, many authors consider matrix games where the strategy set for each player is a polytope (or more generally, a polyhedron), which is known as polyhedral games (e.g., see [Was03]). Even though, each polyhedral game can be reduced to a matrix game by using the vertex representation of each polytope (see e.g. [Sch86]), this transformation may be (and is typically) not algorithmically efficient since the number of vertices may be exponential in the number of facets by which each polytope is given. In a more recent paper, Hofbauer and Sorin [HS06] showed the convergence of the fictitious play for 2-player zero-sum games in which the payoff function is convex-concave and each set X (or Y) is compact convex set. Regardless of the convergence of the fictitious play, let us write it for every 2-player zero-sum games with payoff $F(x, y) : X \times Y \to \mathbb{R}$ as follows,

$$x(t+1) = \frac{t}{t+1}x(t) + \frac{1}{t+1}\xi(t), \text{ where } \xi(t) = \operatorname{argmin}_{\xi \in X} F(\xi, y(t)), \quad (3.5)$$

$$y(t+1) = \frac{t}{t+1}y(t) + \frac{1}{t+1}\eta(t), \text{ where } \eta(t) = \operatorname{argmax}_{\eta \in Y} F(x(t), \eta), \quad (3.6)$$

where $\xi(t)$ and $\eta(t)$ denote the best responses of minimizer and the maximizer, respectively.

Randomized Fictitious Play. In [GK95], Grigoriadis and Khachiyan introduced a randomized variant of fictitious play for matrix games. Their algorithm replaces the minimum and maximum selections (3.3)-(3.4) by a smoothed version, in which, at each time step t, the minimizing player selects a strategy $i \in [m]$ with probability proportional to $\exp\left\{-\frac{\varepsilon}{2}e_i A y(t)\right\}$. Similarly, the maximizing player chooses strategy $j \in [n]$ with probability proportional to $\exp\left\{\frac{\varepsilon}{2}x(t)Ae_j\right\}$. Grigoriadis and Khachiyan proved that, if $A \in [-1, 1]^{m \times n}$, then this algorithm converges, with high probability, to an ε-saddle point in $O(\frac{\log(m+n)}{\varepsilon^2})$ iterations. Each iteration takes $O(n + m)$ time to select strategies i and j, for the players.

The Multiplicative Weights Update Method. In a similar line of work, Freund and Schapire [FS99] used a method, originally developed by Littlestone and Warmuth [LW94], to give a procedure for computing ε-approximation saddle points for matrix games. Their procedure can be thought of as a derandomization of the randomized fictitious play described above. A number of similar algorithms have also been developed for approximately solving special optimization problems, such as general linear

programming [PST91], multicommodity flow problems [GK98], packing and covering linear programming [PST91, GK98, GK04, KY07, You01], some classes of convex programming [Kha04], and semidefinite programming [AHK05, AK07]. Arora, Hazan and Kale [AHK06] gave a meta algorithm that puts many of these results under one umbrella. In particular, they consider the following scenario: given a set X of decisions, a finite set Y of outputs, and a payoff matrix $M \in \mathbb{R}^{X \times Y}$ such that $M(x, y)$ is the penalty that would be paid if decision $x \in X$ was made and output $y \in Y$ was the result, the objective is to develop a decision making strategy that tends to minimize the total payoff over many rounds of such decision making. Suppose that an oracle for finding $\max_{y \in Y} \sum_{i \in [m]} \lambda_i f_i(y)$ exists, where $f_1, \ldots, f_m$ are real-valued convex functions from convex set Y and for any non-negative vector $\lambda \in \mathbb{R}^m$ we have $\sum_{i=1}^m \lambda_i = 1$. Then, Arora et al. [AHK06, Kal07] show how to apply this framework and approximately compute the objective

$$\max_{y \in Y} \min_{i \in [m]} f_i(y).$$

There are two reasons why this method cannot be (directly) used to solve our problem (3.2). First, the number of decisions m is infinite in our case, and second, we do not assume to have access to an oracle of the type described above; we assume only a (weakest possible) membership oracle on Y. Our algorithm extends the multiplicative update method to the computation of ε-approximation saddle points.

Hazan's Work. In his Ph.D. Thesis [Haz06, Chapters 4 and 5], Hazan gave an algorithm, based on multiplicative weights updates method, for approximating the minimum of a convex function within an absolute error of ε. This algorithm can be written in the same form as our algorithm (see Subsection 3.3.1), except that it chooses respectively the points $\xi(t) \in X$ and $\eta(t) \in Y$, at each time step $t = 1, \ldots, T$, as the (approximate) centroids of the corresponding sets with respect to densities

$$p_\xi(t) = \exp\left\{\sum_{\tau=1}^{t-1} \log(e - F(\xi, \eta(\tau)))\right\}$$

and

$$q_\eta(t) = \exp\left\{\sum_{\tau=1}^{t-1} \log(e + F(\xi(\tau), \eta))\right\},$$

where each of them is a log-concave distribution, and outputs $(\sum_{t=1}^T x(t)/T, \sum_{t=1}^T y(t)/T)$ at the end. However, no claim was given regarding the running time or even the convergence for such an extension, and in fact, the proof technique used in [Haz06, Theorem 4.14] does not seem to extend to this case since the function $\log(e - F(\xi, \eta(\tau)))$ (respectively, $\log(e + F(\xi(\tau), \eta))$) is not concave in $\eta(\tau)$ (respectively, not convex in $\xi(\tau)$).

Optimization via Random Sampling. Our algorithm makes use of known algorithms for sampling from a given log-concave distribution over a convex set $X \subseteq \mathbb{R}^m$. First note that we can write (3.1) as the convex minimization problem

$$\inf_{x \in X} F(x),$$

where $F(x) = \sup_{y \in Y} F(x, y)$ is a convex function. Thus, it is worth comparing the bounds we obtain (see next section Theorem 3.3.1) with the bounds that one could obtain by applying the random sampling techniques for convex optimization. Several algorithms for convex optimization based on sampling have been recently proposed. In what follows we briefly mention some of them. It is easy to see that any minimization problem is reducible in polytime to checking whether for a given v, the corresponding set K_v is empty or not, where K_v is defined as

$$K_v = \{x \in X : F(x) \leqslant v\}.$$

If an algorithm is able to check set K_v, for every v, then binary search over all possible values of v and finding a nonempty K_v with minimum v solves the problem. It is readily checked that if F is a convex function, then K_v, for every v, is a convex set. Bertsimas and Vempala [BV04] proposed an algorithm based on random sampling of points from a convex body to minimize F over X, assuming a membership oracle exists. For a given K_v, their algorithm outputs for a given v, either K_v is empty or a point that belongs K_v. They also showed that running time is bounded by $O^*((m^5T+m^7)\log R)$, where T is the time required by a single oracle call. Later on, Kalai and Vempala [KV06] applied a method known as *simulated annealing* to design an algorithm for minimizing a linear function over an arbitrary convex set which is specified only by a membership oracle. They showed an upper bound of $O^*(m^{4.5}T \log R)$, which is an improvement over the previous one. In fact, their algorithm samples points from a convex set according to a family of *Boltzman-Gibbs* distributions. Recall that we define $= F(x) = \sup_{y \in Y} F(x, y)$. So, if one wants to apply the aforementioned technique to approximate the saddle point, then each membership call involves another application of these techniques (to check if $\sup_{y \in Y} F(x, y) \leqslant v$). Therefore, the running time of the algorithm is bounded by $O^*(n^{4.5}(m^5T + m^7)\log^{O(1)} R)$, which is significantly greater than the bound in our result.

Ellipsoid Method. A popular method for solving convex optimization problems is the Ellipsoid method. Under Assumption (A1), the Ellipsoid method can be used to minimize a linear function over a convex set $X \subseteq \mathbb{R}^m$, which is given by a membership oracle, in time $O(m^{10}T \log R + m^{12} \log R)$ (see [GLS93] and Table 1 in [BV04]). By a similar argument as the one given above, in the special case when $F(x, y)$ is linear in y, the method takes a total running time of $O^*((n^{10}(m^{10}T+m^{12})+n^{12})\log^{O(1)} R)$ to find a saddle point, which is significantly greater than the bound stated in Theorem 3.3.1.

3.3. Main Results

In this section we formally state our main theorem and approximation algorithm. A main ingredient of our algorithm is a sampling algorithm from a log-concave density $f(\cdot)$ over a convex set $X \subseteq \mathbb{R}^m$. The currently best known result achieving this is due to Lovász and Vempala (see, e.g., [LV07, Theorem 2.1]). In this work, the authors introduced a random walk on X called *ball walk* that proceeds in rounds. Let us fix a positive number $r > 0$ and let $x \in X$ be the current position of the walk. Then it picks $y \in B^m(x, r)$ uniformly at random and moves to y with probability $\min(1, f(x)/f(y))$ and stays at x with remaining probability. They showed that after $O^*(\frac{m^5}{\varepsilon^4})$ rounds,

Algorithm 5. Randomized fictitious play

Require: Two convex bounded sets X, Y and a function $F(x, y)$ such that $F(\cdot, y) : X \to \mathbb{R}$ is convex for all $y \in Y$ and $F(x, \cdot) : Y \to \mathbb{R}$ is concave for all $x \in X$, satisfying assumptions (A1) and (A2)

Ensure: A pair of ε-optimal strategies

1: $t := 0$; choose $x(0) \in X$; $y(0) \in Y$, arbitrarily
2: **while** $t \leqslant T$ **do**
3: Pick $\xi \in X$ and $\eta \in Y$, independently, from X and Y with densities $\frac{p_\xi(t)}{\|p(t)\|_1}$ and $\frac{q_\eta(t)}{\|q(t)\|_1}$, respectively
4: $x(t+1) := \frac{t}{t+1}x(t) + \frac{1}{t+1}\xi$; $y(t+1) := \frac{t}{t+1}y(t) + \frac{1}{t+1}\eta$; $t := t+1$;
5: **end while**
6: **return** $(\mathrm{x(t)}, y(t))$

the walk distribution over X converges to f within a total variation distance of ε with high probability.

Recall that, for every two probability distributions ν and τ on $\mathcal{F}$, the *total variation distance* between ν and τ is defined as follows

$$d_{TV}(\nu, \tau) = \sup_{A \subseteq \mathcal{F}} (\nu(A) - \tau(A)).$$

Note that, we apply random sampling as a black-box for each iteration independently; it might be possible to improve the running time if we utilize the fact that the distributions are slightly modified from an iteration to the next.

Theorem 3.3.1 (Main Theorem)**.** *Assume X and Y satisfy assumption (A1). Then there is a randomized algorithm that finds a pair of ε-optimal strategies in an expected number of $O\left(\frac{\rho^2(n+m)}{\varepsilon^2}\log\frac{R}{\varepsilon}\right)$ iterations, each computing two samples from log-concave distributions. In particular, the algorithm requires $O^*\left(\frac{\rho^2(n+m)^6}{\varepsilon^2}\log R\right)$ oracle calls, where $O^*(\cdot)$ suppresses polylogarithmic factors that depend on n, m and ε.*

In order to prove the theorem, we present Algorithm 5 and show that the algorithm outputs an ε-approximation saddle point. We also derive an upper bound for its running time as stated in the theorem. For a complete analysis and correctness see Section 3.4.

3.3.1. The Algorithm

In this subsection, we present Algorithm 5 that is an adaptation of the algorithms in [GK95] and [FS99]. It proceeds in steps $t = 0, 1, \ldots$, updating the pair of *accumulative* strategies $x(t)$ and $y(t)$. Given the current pair $(x(t), y(t))$, define

$$p_\xi(t) = e^{-\frac{\varepsilon t F(\xi, y(t))}{2}} \quad \text{for } \xi \in X, \tag{3.7}$$

$$q_\eta(t) = e^{\frac{\varepsilon t F(x(t), \eta)}{2}} \quad \text{for } \eta \in Y, \tag{3.8}$$

and let

$$\|p(t)\|_1 = \int_{\xi \in X} p_\xi(t) d\xi \qquad \text{and} \qquad \|q(t)\|_1 = \int_{\eta \in Y} q_\eta(t) d\eta$$

be the respective normalization factors. The parameter T will be specified later (see Lemma 3.4.5).

3.4. The Analysis of Algorithm 5

In this section we show that the algorithm outputs an ε-approximation saddle point and derive an upper bound for its running time. The analysis is composed of three parts, which we briefly explain here. For a complete proof see the following respective subsections.

I. **Bounding the Potential Increase.** Following [GK95], we use potential function

$$\Phi(t) = \|p(t)\|_1 \|q(t)\|_1 = \int_{\xi \in X, \eta \in Y} e^{\frac{\varepsilon}{2} t(F(x(t),\eta) - F(\xi, y(t)))} d\xi d\eta, \tag{3.9}$$

to bound the number of iterations required by the algorithm to reach an ε-approximation saddle point. This is a generalization of the argument in [GK95] (and [KY07]): we show that the potential function increases, on the average, only by a factor of $e^{O(\varepsilon^2)}$, implying that after t iterations the potential is at most $e^{O(\varepsilon^2)t}$ factor of the initial potential. The analysis would end, if we could argue that after t iterations the function under integral is also bounded by $e^{O(\varepsilon^2)t}$ and hence after t iterations an ε-approximation algorithm is achieved. In case X and Y are simplices and the potential is a sum over all vertices of the simplices [GK95], then the upper bound for the potential function is sufficient to bound the summands. However, we know that if a definite integral of a non-negative function over a given region Q is bounded by some τ, then it does not imply that the function is also bounded by τ, at any point in Q. So, we still need to argue that the function cannot largely deviate from the τ. Basically, we solve this problem in the next part.

II. **Bounding the Number of Iterations.** In this part of the analysis, we overcome the aforementioned difficulty by showing that, due to concavity of the exponent of the function under the integral in (3.9), the change in the function around a given point cannot be too large. Thus, the value at a given point cannot be large unless there is a sufficiently large fraction of the volume of the sets X and Y over which the integral is also too large.

III. **Approximate Distributions.** Finally, we show that the same bound on the running time holds when the sampling distributions in line 3 of the algorithm are replaced by sufficiently close approximate distributions. We then show that the algorithm converges to a ε-optimal strategies after a certain number of rounds.

3.4.1. Bounding the Potential Increase

In this subsection, we establish an upper bound for the potential function $\Phi(t)$, which was defined in (3.9). In the next lemma, we estimate the expected increase of $\Phi(t)$ after execution of each round of the algorithm and then, applying law of iterated expectations, we show that

$$\Phi(t) = \int_{\xi\in X,\eta\in Y} e^{\frac{\varepsilon}{2}t(F(x(t),\eta)-F(\xi,y(t)))}d\xi d\eta \leqslant 2e^{\frac{\varepsilon^2}{3}t}\Phi(0).$$

Lemma 3.4.1. *For $t = 0, 1, 2, \ldots,$*

$$\mathbb{E}[\Phi(t+1)] \leqslant \mathbb{E}[\Phi(t)]\left(1+\frac{\varepsilon}{6}^2\right)^2.$$

Proof. Conditional on the values of $x(t)$ and $y(t)$, we have

$$\begin{aligned}
\|p(t+1)\|_1 &= \int_{\xi\in X} e^{-\frac{\varepsilon(t+1)F(\xi,y(t+1))}{2}}d\xi \\
&= \int_{\xi\in X} e^{-\frac{\varepsilon(t+1)F(\xi,\frac{t}{t+1}y(t)+\frac{1}{t+1}\eta)}{2}}d\xi \\
&\leqslant \int_{\xi\in X} e^{-\frac{\varepsilon tF(\xi,y(t))}{2}}e^{-\frac{\varepsilon F(\xi,\eta)}{2}}d\xi \\
&= \int_{\xi\in X} p_\xi(t)e^{-\frac{\varepsilon F(\xi,\eta)}{2}}d\xi \\
&\leqslant \int_{\xi\in X} p_\xi(t)\left[1+\frac{\varepsilon^2}{6}-\frac{\varepsilon}{2}F(\xi,\eta)\right]d\xi \\
&= \|p(t)\|_1\left(1+\frac{\varepsilon^2}{6}-\frac{\varepsilon}{2}\frac{\int_{\xi\in X}p_\xi(t)F(\xi,\eta)d\xi}{\|p(t)\|_1}\right),
\end{aligned} \tag{3.10}$$

where it follows from assumption (A2), concavity of $F(\xi,\cdot): Y \to \mathbb{R}$ and the inequality $e^\delta \leqslant 1+\delta+\frac{2}{3}\delta^2$, which is valid for all $\delta \in [-\frac{1}{2}, \frac{1}{2}]$. Taking the expectation with respect to η (with density proportional to $q_\eta(t)$), we get

$$\mathbb{E}_q[\|p(t+1)\|_1] \leqslant \|p(t)\|_1\left[1+\frac{\varepsilon^2}{6}-\frac{\varepsilon}{2}\frac{\int_{\eta\in Y}q_\eta(t)\int_{\xi\in X}p_\xi(t)F(\xi,\eta)d\xi d\eta}{\|q(t)\|_1\|p(t)\|_1}\right]. \tag{3.11}$$

Similarly, by taking the expectation with respect to ξ (with density proportional to $p_\xi(t)$), we can derive

$$\mathbb{E}_p[\|q(t+1)\|_1] \leqslant \|q(t)\|_1\left[1+\frac{\varepsilon^2}{6}+\frac{\varepsilon}{2}\frac{\int_{\xi\in X}p_\xi(t)\int_{\eta\in Y}q_\eta(t)F(\xi,\eta)d\eta d\xi}{\|p(t)\|_1\|q(t)\|_1}\right]. \tag{3.12}$$

Now, using independence of ξ and η, we have

$$\begin{aligned}\mathbb{E}[\Phi(t+1)|x(t),y(t)] \;\leqslant\; & \Phi(t)\left[\left(1+\frac{\varepsilon^2}{6}\right)^2\right. \\ & + \frac{\varepsilon}{2}\left(1+\frac{\varepsilon^2}{6}\right)\left(\frac{\int_{\xi\in X} p_\xi(t)\int_{\eta\in Y} q_\eta(t)F(\xi,\eta)d\eta d\xi}{\|p(t)\|_1\|q(t)\|_1} - \frac{\int_{\eta\in Y} q_\eta(t)\int_{\xi\in X} p_\xi(t)F(\xi,\eta)d\xi d\eta}{\|q(t)\|_1\|p(t)\|_1}\right) \\ & \left. -\frac{\varepsilon^2}{4}\frac{\int_{\xi\in X} p_\xi(t)\int_{\eta\in Y} q_\eta(t)F(\xi,\eta)d\eta d\xi}{\|p(t)\|_1\|q(t)\|_1}\cdot\frac{\int_{\eta\in Y} q_\eta(t)\int_{\xi\in X} p_\xi(t)F(\xi,\eta)d\xi d\eta}{\|q(t)\|_1\|p(t)\|_1}\right].\end{aligned}$$

By interchanging the order of integration, we get that the second part of the sum on the right-hand side is zero, and third part is non-positive. Hence

$$\mathbb{E}[\Phi(t+1)|x(t),y(t)] \leqslant \Phi(t)\left(1+\frac{\varepsilon^2}{6}\right)^2. \tag{3.13}$$

The lemma follows by taking the expectation of (3.13) with respect to $x(t)$ and $y(t)$. □

Corollary 3.4.2. *With probability at least $\frac{1}{2}$, after t iterations*

$$\Phi(t) \leqslant 2e^{\frac{\varepsilon^2}{3}t}\Phi(0). \tag{3.14}$$

Proof. By applying Lemma 3.4.1 we have $\mathbb{E}[\Phi(t+1)] \leqslant \mathbb{E}[\Phi(t)]\left(1+\frac{\varepsilon}{6}^2\right)^2$. Using the law of iterated expectations, we have

$$\mathbb{E}[\Phi(t)] \leqslant \Phi(0)\cdot\left(1+\frac{\varepsilon}{6}^2\right)^{2t} \leqslant \Phi(0)\cdot e^{\frac{\varepsilon^2}{3}t},$$

where the last inequality follows from $1+\delta \leqslant e^\delta$, for every $\delta \geqslant 0$. Now, applying Markov's inequality we conclude that

$$\Pr[\Phi(t) \geqslant 2e^{\frac{\varepsilon^2}{3}t}\Phi(0)] \leqslant \mathbb{E}[\Phi(t)]/2e^{\frac{\varepsilon^2}{3}t}\Phi(0) \leqslant 1/2.$$

□

3.4.2. Bounding the Number of Iterations

In this subsection we show that after desired number of iterations, say T, the algorithm outputs an ε-optimal strategies. For convenience, define $Z = X \times Y$, and concave function $g_t : Z \to \mathbb{R}$ for every $z = (\xi,\eta) \in Z$ as

$$g_t(\xi,\eta) := \frac{\varepsilon}{2}t\left(F(x(t),\eta) - F(\xi,y(t))\right).$$

Note that, by our assumptions, Z is a full-dimensional and bounded convex set in $\mathbb{R}^N$, where $N = n+m$. Moreover, we have that

$$\Phi(0) = \int_{\xi\in X,\eta\in Y} 1 d\xi d\eta = vol(X)\cdot vol(Y),$$

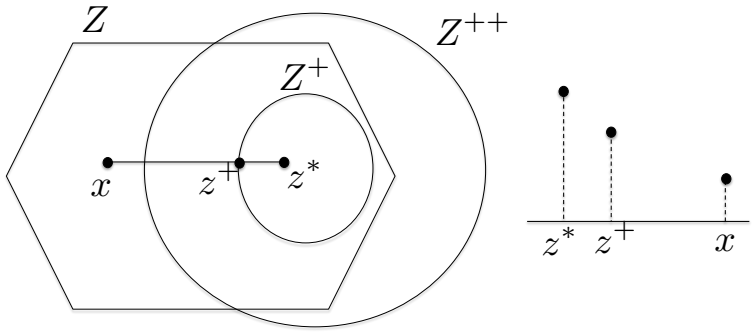

Figure 3.1. The drawing on the left illustrates the notation used in the proof of Lemma 3.4.3. We assume for the sake of a contradiction that there is a point $x \in Z \setminus Z^{++}$. Observe that Z^{++} is a scaled version of Z^{+}. The drawing on the right illustrates the contradiction. A function that drops by $\alpha/2$ from z^* to z^+ and by at most $2\varepsilon t$ from z^* to x cannot be concave.

By assumption (A2), we have that for all $z \in Z$

$$|g_t(z)| = |\frac{\varepsilon}{2} t \left(F(x(t), \eta) - F(\xi, y(t))\right)| \leqslant \varepsilon t/2(|F(x(t), \eta)| + |F(\xi, y(t))|) \leqslant \varepsilon t. \quad (3.15)$$

In the following lemma, we provide a sufficient condition for the convergence of the algorithm to an ε-approximate saddle point.

Lemma 3.4.3. *Suppose that (3.14) holds and there exists α such that*

$$0 < \alpha < 4\varepsilon t, \quad (3.16)$$

$$e^{\frac{1}{2}\alpha} \left(\frac{\alpha}{4\varepsilon t}\right)^N vol(Z) > 1. \quad (3.17)$$

Then we have that, for every $z \in Z$,

$$e^{g_t(z)} \leqslant 2e^{\frac{\varepsilon^2}{3}t + \alpha} \Phi(0). \quad (3.18)$$

Proof. Toward a contradiction assume that, there is $z^* \in Z$ with

$$g_t(z^*) > \frac{\varepsilon^2}{3} t + \alpha + \log(2\Phi(0)).$$

Let $\lambda^* = \alpha/(4\varepsilon t) < 1$,

$$Z^+ = \{z \in Z | g_t(z) \geqslant g_t(z^*) - \alpha/2\}, \text{ and } Z^{++} = \{z^* + \frac{1}{\lambda^*}(z - z^*) | z \in Z^+\}.$$

To have a clear picture of the sets, defined here, see Figure 3.1. Concavity of g_t implies convexity of Z^+. Thus, for every $z \in Z^+$ and every λ', $0 \leqslant \lambda' \leqslant 1/\lambda^*$, we have $\lambda^*\lambda' z + (1 - \lambda^*\lambda')z^* \in Z^+$, and hence

$$z^* + \frac{1}{\lambda*}(\lambda^*\lambda' z + (1 - \lambda^*\lambda')z^* - z^*) = z^* + \lambda'(z - z^*) \in Z^{++}.$$

Thus, for every $z \in Z^+$, the entire ray $\{z^* + \lambda'(z - z^*) : 0 \leqslant \lambda' \leqslant 1/\lambda^*\}$ belongs to Z^{++}. In particular, $Z^+ \subseteq Z^{++}$. We next show $Z \subseteq Z^{++}$. Toward a contradiction assume that $x \in Z \setminus Z^{++}$ (and hence $x \in Z \setminus Z^+$). Let us define

$$\lambda^+ = \sup\{\lambda \mid z^* + \lambda(x - z^*) \in Z^+\} \text{ and } z^+ = z^* + \lambda^+(x - z^*).$$

By continuity of g_t, $z^+ \in Z^+$ and $g_t(z^*) - \alpha/2 = g_t(z^+)$. By definition of z^+, we have $x - z^* = \frac{1}{\lambda^+}(z^+ - z^*)$ and hence

$$\frac{1}{\lambda^+} > \frac{1}{\lambda^*}. \tag{3.19}$$

But $z^+ = \lambda^+ x + (1 - \lambda^+)z^*$ and hence

$$g_t(z^*) - \alpha/2 = g_t(z^+) = g_t(\lambda^+ x + (1 - \lambda^+)z^*) \geqslant \lambda^+ g_t(x) + (1 - \lambda^+)g_t(z^*).$$

Thus, we have

$$\frac{\alpha}{2} \leqslant \lambda^+(g_t(z^*) - g_t(x)) \leqslant \lambda^+(|g_t(z^*)| + |g_t(x)|) \leqslant 2\varepsilon t\lambda^+,$$

where the last inequality comes from (3.15) because $z^*, x \in Z$. Therefore, we have

$$\lambda^+ \geqslant \frac{\alpha}{4\varepsilon t} = \lambda^*,$$

which contradicts (3.19). We have now established $Z \subseteq Z^{++}$. By definition, we have $Z^{++} = \frac{1}{\lambda^*}Z^+ + (1 - \frac{1}{\lambda^*})z^*$. Since the volume of a body is invariant under translation, we have

$$vol(Z) \leqslant vol(Z^{++}) = vol\left(\frac{1}{\lambda^*}Z^+\right) = \left(\frac{1}{\lambda^*}\right)^N vol(Z^+)$$

and further

$$\begin{aligned}
\Phi(t) &= \int_{z \in Z} e^{g_t(z)} dz \\
&\geqslant \int_{z \in Z^+} e^{g_t(z)} dz \\
&\geqslant 2\Phi(0) e^{\frac{\varepsilon^2}{3}t + \frac{1}{2}\alpha} vol(Z^+) \\
&\geqslant 2\Phi(0) e^{\frac{\varepsilon^2}{3}t + \frac{1}{2}\alpha} \left(\frac{\alpha}{4\varepsilon t}\right)^N vol(Z) \\
&> 2\Phi(0) e^{\frac{\varepsilon^2}{3}t},
\end{aligned}$$

which is a contradiction to (3.14). We therefore conclude that for every $z \in Z$,

$$e^{g_t(z)} \leqslant 2e^{\frac{\varepsilon^2}{3}t + \alpha}\Phi(0).$$

□

We can now derive an upper bound for the number of iterations needed to converge to ε-optimal strategies..

Lemma 3.4.4. *Assume that there exists α satisfying (3.16), (3.17). If we choose t so that*

$$t \geqslant \frac{6}{\varepsilon^2}(\alpha + \max\{0, \log(2\, vol(Z))\}), \tag{3.20}$$

then $(x(t), y(t))$ is an ε-optimal pair and (3.16) holds.

Proof. By (3.18) we have that for all $z \in Z$,

$$g_t(z) \leqslant \frac{\varepsilon^2}{3}t + \alpha + \log(2\Phi(0)) = \frac{\varepsilon^2}{3}t + \alpha + \log(2\, vol(Z)).$$

Or equivalently,

$$\frac{\varepsilon}{2}t(F(x(t), \eta) - F(\xi, y(t))) \leqslant \frac{\varepsilon^2}{3}t + \alpha + \log(2\, vol(Z)) \quad \text{for all } \xi \in X \text{ and } \eta \in Y.$$

Dividing the both sides by $\varepsilon t/2$ gives

$$F(x(t), \eta) \leqslant F(\xi, y(t)) + \frac{2\varepsilon}{3} + \frac{2}{\varepsilon t}(\alpha + \log(2\, vol(Z)),$$

Now, applying the assumption about t in the lemma (i.e., (3.20)) implies that

$$F(x(t), \eta) \leqslant F(\xi, y(t)) + \varepsilon \quad \text{for all } \xi \in X \text{ and } \eta \in Y.$$

Finally, (3.16) holds since $4\varepsilon t \geqslant 24\alpha/\varepsilon > \alpha$. □

In the following lemma, we show there exist α and t satisfying the assumptions in Lemma 3.4.3.

Lemma 3.4.5. *For every $\varepsilon \in (0, 1)$, there exist α and $t = O\left(\frac{N}{\varepsilon^2}\log\frac{R}{\varepsilon}\right)$ satisfying (3.16), (3.17) and (3.20).*

Proof. To show the lemma, we consider two cases:

Case 1. $vol(Z) \leqslant \frac{1}{2}$. Let us set $t = 6\alpha/\varepsilon^2$ and take the logarithm from both sides of inequality (3.17), then we get

$$\frac{\alpha}{2} + N\log\left(\frac{\alpha}{4\varepsilon t}\right) + \log(vol(Z)) > 0.$$

Now, if we chose $\frac{\alpha}{2} = N\log(\frac{25}{\varepsilon}) - \log(vol(Z)))$, then the α and t would satisfy the inequality (3.17) . Hence, we have that

$$t = 6\alpha/\varepsilon^2 = O\left(\frac{N}{\varepsilon^2}\log\frac{1}{\varepsilon} + \frac{1}{\varepsilon^2}\log\frac{1}{vol(Z)}\right) = O\left(\frac{N}{\varepsilon^2}\log\frac{R}{\varepsilon}\right),$$

where the last inequality follows since $1/\, vol\, Z \leqslant R^N$.

Case 2. $vol(Z) > \frac{1}{2}$. Then we have

$$e^{\frac{\alpha}{2}} \left(\frac{\alpha}{4\varepsilon t}\right)^N vol(Z) > \frac{1}{2} e^{\frac{\alpha}{2}} \left(\frac{\alpha}{4\varepsilon t}\right)^N,$$

In order to satisfy inequality (3.17), it is sufficient to find α and t satisfying

$$\frac{1}{2} e^{\frac{\alpha}{2}} \left(\frac{\alpha}{4\varepsilon t}\right)^N > 1.$$

To satisfy (3.20), let us simply choose $t = 6\alpha/\varepsilon^2 + (6/\varepsilon^2)\log(2\,vol(Z))$ and demand that

$$\frac{1}{2} e^{\frac{\alpha}{2}} \left(\frac{\alpha}{4\varepsilon t}\right)^N = \frac{1}{2} e^{\frac{\alpha}{2}} \left(\frac{\alpha}{\frac{24\alpha}{\varepsilon} + \frac{24}{\varepsilon}\log(2\,vol(Z))}\right)^N > 1,$$

or equivalently

$$2\left(\frac{24}{\varepsilon}\right)^N \left(1 + \frac{\log(2\,vol(Z))}{\alpha}\right)^N < e^{\frac{\alpha}{2}}.$$

Thus, it is enough to select

$$\alpha = \max\left\{4(\log 2 + N\log(\frac{24}{\varepsilon})), 2\sqrt{N\log(2\,vol(Z))}\right\},$$

which satisfies

$$2\left(\frac{24}{\varepsilon}\right)^N \leqslant e^{\frac{\alpha}{4}} \quad \text{and} \quad \left(1 + \frac{\log(2\,vol(Z))}{\alpha}\right)^N < e^{\frac{\log(2\,vol(Z))}{\alpha} N} \leqslant e^{\frac{\alpha}{4}}.$$

It follows that

$$t = \max\left\{\frac{24}{\varepsilon^2}(\log 2 + N\log(\frac{24}{\varepsilon})), \frac{12}{\varepsilon^2}\sqrt{N\log(2\,vol(Z))}\right\} + \frac{6}{\varepsilon^2}\log(2\,vol(Z)).$$

Since $vol(Z) \leqslant R^N$, we have that $t = O\left(\frac{N}{\varepsilon^2}\log\frac{R}{\varepsilon}\right)$

As we see in both cases (3.16) holds by the preceding lemma. □

Corollary 3.4.6. *Assume X and Y satisfy assumptions (A1) and (A2). Then Algorithm 5, when run with T satisfying the bound in Lemma 3.4.5, computes a pair of ε-approximation saddle point in expected $O(\frac{n+m}{\varepsilon^2}\log\frac{R}{\varepsilon})$ iterations.*

3.4.3. Using Approximate Distributions

We now consider the (realistic) situation when we can only sample approximately from the convex sets. In this case we assume the existence of approximate sampling routines that, upon the call in step 3 of the algorithm, return vectors $\xi \in X$, and (independently) $\eta \in Y$, with densities $\hat{p}_\xi(t)$ and $\hat{q}_\eta(t)$, such that

$$\sup_{X' \subseteq X} \left|\frac{\hat{p}_{X'}(t)}{\hat{p}_X(t)} - \frac{p_{X'}(t)}{p_X(t)}\right| \leqslant \delta \quad \text{and} \quad \sup_{Y' \subseteq Y} \left|\frac{\hat{q}_{Y'}(t)}{\hat{q}_Y(t)} - \frac{q_{Y'}(t)}{q_Y(t)}\right| \leqslant \delta, \tag{3.21}$$

where

$$\hat{p}_{X'}(t) = \int_{\xi \in X'} \hat{p}_\xi d\xi.$$

Similarly, define $p_{X'}(t), \hat{q}_{Y'}(t), q_{Y'}(t)$, and δ is a given desired accuracy. We next prove an approximate version of Lemma 3.4.1.

Lemma 3.4.7. *Suppose that we use approximate sampling routines with $\delta = \varepsilon/4$ in step 3 of Algorithm 5. Then, for $t = 0, 1, 2, \ldots$, we have*

$$\mathbb{E}[\Phi(t+1)] \leqslant \mathbb{E}[\Phi(t)](1 + \frac{43}{36}\varepsilon^2).$$

Proof. The argument up to Equation (3.10) remains the same. Taking the expectation with respect to η (with density proportional to $\hat{q}_\eta(t)$), we get

$$\mathbb{E}_{\hat{q}}[\|p(t+1)\|_1] \leqslant \|p(t)\|_1 \left[1 + \frac{\varepsilon^2}{6} - \frac{\varepsilon}{2} \frac{\int_{\eta \in Y} \hat{q}_\eta(t) \int_{\xi \in X} p_\xi(t) F(\xi, \eta) d\xi d\eta}{\|\hat{q}(t)\|_1 \|p(t)\|_1}\right]. \quad (3.22)$$

Similarly,

$$\mathbb{E}_{\hat{p}}[\|q(t+1)\|_1] \leqslant \|q(t)\|_1 \left[1 + \frac{\varepsilon^2}{6} + \frac{\varepsilon}{2} \frac{\int_{\xi \in X} \hat{p}_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{\|\hat{p}(t)\|_1 \|q(t)\|_1}\right]. \quad (3.23)$$

Thus, by independence of ξ and η, we have

$$\begin{aligned}
\mathbb{E}[\Phi(t+1)|x(t), y(t)] \quad \leqslant \quad & \Phi(t) \left[\left(1 + \frac{\varepsilon^2}{6}\right)^2 \right. \\
& + \frac{\varepsilon}{2}\left(1 + \frac{\varepsilon^2}{6}\right)\left(\frac{\int_{\xi \in X} \hat{p}_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{|\hat{p}(t)| \|q(t)\|_1} - \frac{\int_{\eta \in Y} \hat{q}_\eta(t) \int_{\xi \in X} p_\xi(t) F(\xi, \eta) d\xi d\eta}{\|\hat{q}(t)\|_1 \|p(t)\|_1}\right) \\
& \left. - \frac{\varepsilon^2}{4} \frac{\int_{\xi \in X} \hat{p}_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{\|\hat{p}(t)\|_1 \|q(t)\|_1} \cdot \frac{\int_{\eta \in Y} \hat{q}_\eta(t) \int_{\xi \in X} p_\xi(t) F(\xi, \eta) d\xi d\eta}{\|\hat{q}(t)\|_1 \|p(t)\|_1}\right].
\end{aligned}$$

□

We will make use of the following proposition.

Proposition 3.4.8. *If we set $\delta = \varepsilon/4$ in (3.21), then*

$$\left|\frac{\int_{\xi \in X} \hat{p}_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{|\hat{p}(t)| \|q(t)\|_1} - \frac{\int_{\eta \in Y} \hat{q}_\eta(t) \int_{\xi \in X} p_\xi(t) F(\xi, \eta) d\xi d\eta}{\|\hat{q}(t)\|_1 \|p(t)\|_1}\right| \leqslant \varepsilon. \quad (3.24)$$

Proof. Since

$$\frac{\int_{\xi \in X} p_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{\|p(t)\|_1 \|q(t)\|_1} = \frac{\int_{\eta \in Y} q_\eta(t) \int_{\xi \in X} p_\xi(t) F(\xi, \eta) d\xi d\eta}{\|q(t)\|_1 \|p(t)\|_1},$$

we can bound the L.H.S. of (3.24) by

$$\left|\frac{\int_{\xi \in X} p_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{\|p(t)\|_1 \|q(t)\|_1} - \frac{\int_{\xi \in X} \hat{p}_\xi(t) \int_{\eta \in Y} q_\eta(t) F(\xi, \eta) d\eta d\xi}{|\hat{p}(t)| \|q(t)\|_1}\right| +$$

$$\left| \frac{\int_{\eta\in Y} q_\eta(t) \int_{\xi\in X} p_\xi(t)F(\xi,\eta)d\xi d\eta}{\|q(t)\|_1\|p(t)\|_1} - \frac{\int_{\eta\in Y} \hat{q}_\eta(t) \int_{\xi\in X} p_\xi(t)F(\xi,\eta)d\xi d\eta}{\|\hat{q}(t)\|_1\|p(t)\|_1} \right|. \tag{3.25}$$

Thus it is enough to show that each term in (3.25) is at most $\frac{\varepsilon}{2}$. Since the two terms are similar, we only consider the first term. Define $X' = \{\xi \in X \ : \ \frac{p_\xi(t)}{p_X(t)} \geqslant \frac{\hat{p}_\xi(t)}{\hat{p}_X(t)}\}$ and $X'' = X \setminus X'$.

$$\begin{aligned}
&\frac{1}{\|q(t)\|_1} \left| \int_{\xi\in X} \int_{\eta\in Y} q_\eta(t)F(\xi,\eta) \left(\frac{p_\xi(t)}{\|p(t)\|_1} - \frac{\hat{p}_\xi(t)}{|\hat{p}(t)|} \right) d\eta d\xi \right| \\
&\qquad \leqslant \frac{1}{q_Y(t)} \int_{\xi\in X} \int_{\eta\in Y} q_\eta(t)|F(\xi,\eta)| \left| \frac{p_\xi(t)}{p_X(t)} - \frac{\hat{p}_\xi(t)}{\hat{p}_X(t)} \right| d\eta d\xi \\
&\qquad \leqslant \frac{1}{q_Y(t)} \int_{\xi\in X} \int_{\eta\in Y} q_\eta(t) \left| \frac{p_\xi(t)}{p_X(t)} - \frac{\hat{p}_\xi(t)}{\hat{p}_X(t)} \right| d\eta d\xi \quad \text{(by (A2))} \\
&\qquad = \int_{\xi\in X} \left| \frac{p_\xi(t)}{p_X(t)} - \frac{\hat{p}_\xi(t)}{\hat{p}_X(t)} \right| d\xi \\
&\qquad = \int_{\xi\in X'} \left(\frac{p_\xi(t)}{p_X(t)} - \frac{\hat{p}_\xi(t)}{\hat{p}_X(t)} \right) d\xi + \int_{\xi\in X''} \left(\frac{\hat{p}_\xi(t)}{\hat{p}_X(t)} - \frac{p_\xi(t)}{p_X(t)} \right) d\xi \\
&\qquad = \left(\frac{p_{X'}(t)}{p_X(t)} - \frac{\hat{p}_{X'}(t)}{\hat{p}_X(t)} \right) + \left(\frac{\hat{p}_{X''}(t)}{\hat{p}_X(t)} - \frac{p_{X''}(t)}{p_X(t)} \right) \leqslant \frac{\varepsilon}{2} \text{ (by(3.21)).}
\end{aligned}$$

□

Proposition 3.4.8 implies that

$$\mathbb{E}[\Phi(t+1)|x(t),y(t)] \leqslant \Phi(t) \left[\left(1+\frac{\varepsilon^2}{6}\right)^2 + \frac{\varepsilon^2}{2}\left(1+\frac{\varepsilon^2}{6}\right) + \frac{\varepsilon^4}{4} \right] \leqslant \Phi(t)\left(1+\frac{43}{36}\varepsilon^2\right).$$

The rest of the proof is as in Lemma 3.4.1. □

Proof of Theorem 3.3.1. Combining the currently known bound on the mixing time for sampling (see [LV04, LV06, LV07] and also Section 3.2) with the bounds on the number of iterations from Corollary 3.4.6 gives Theorem 3.3.1. □

3.5. Applications in Combinatorial Optimization

In this section we give some examples for which the width parameter ρ is small. For such examples with $\rho \leqslant 1$, we see that our algorithm is superior to the previous known algorithms.

3.5.1. Mixed Popular Matchings

Let $\mathcal{S}, \mathcal{T}$ be two families (say, of combinatorial objects), and $\mathcal{A} \in [-1,1]^{\mathcal{S}\times\mathcal{T}}$ be a given matrix. We assume that these families have exponential size in some input parameter and hence, the matrix is given by an oracle that specifies for each $S \in \mathcal{S}$ and $T \in \mathcal{T}$ the

value of $\mathcal{A}(S,T)$. The objective is to find a saddle point for the matrix game defined by $\mathcal{A}$ and mixed strategy sets, which are defined as,

$$\Delta_{\mathcal{S}} = \{p \in \mathbb{R}^{\mathcal{S}} \mid \sum_{S\in\mathcal{S}} p_S = 1, p \geqslant 0\}$$

and

$$\Delta_{\mathcal{T}} = \{q \in \mathbb{R}^{\mathcal{T}} \mid \sum_{T\in\mathcal{T}} q_T = 1, q \geqslant 0\}.$$

In general, the optimal strategies might have exponential support (i.e., an exponential number of non-zero entries). However, if the families arise from combinatorial objects in a natural way, then the supports of optimal strategies may be polynomially bounded. More precisely, let E and F be two sets of sizes m and n respectively, such that each element $S \in \mathcal{S}$ (respectively, $T \in \mathcal{T}$), is characterized by a vector $x(S) \in \{0,1\}^m$ indexed by the elements of E (respectively, $y(T) \in \{0,1\}^n$ indexed by the elements of F). Le X denote the convex hull of set $\{x(S) : S \in \mathcal{S}\}$ and similarly Y denotes the convex hull of set $\{x(T) : T \in \mathcal{T}\}$. Let us assume that X and Y have explicit linear descriptions, and furthermore that there exists an $m \times n$ matrix A such that $\mathcal{A}(S,T) = x(S)^T Ay(T)$, for all $S \in \mathcal{S}$ and $T \in \mathcal{T}$. Then it follows from Von Neumann's saddle point theorem [Dan63] (which is a special case of Theorem (3.1.2)) that

$$\min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{T}}} p^T \mathcal{A} q = \min_{x\in X} \max_{y\in Y} x^T A y. \tag{3.26}$$

Indeed,

$$\begin{aligned}
\min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{T}}} p^T \mathcal{A} q &= \min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{T}}} \sum_{S\in\mathcal{S},T\in\mathcal{T}} p_S q_T \mathcal{A}(S,T) \\
&= \min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{T}}} \sum_{S\in\mathcal{S},T\in\mathcal{T}} p_S q_T x(S)^T A y(T) \\
&= \min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{T}}} \sum_{S\in\mathcal{S}} p_S x(S)^T A \sum_{T\in\mathcal{T}} q_T y(T) \\
&= \min_{p\in\Delta_{\mathcal{S}}} \max_{y\in Y} \sum_{S\in\mathcal{S}} p_S x(S)^T A y \\
&= \max_{y\in Y} \min_{p\in\Delta_{\mathcal{S}}} \sum_{S\in\mathcal{S}} p_S x(S)^T A y \\
&= \max_{y\in Y} \min_{x\in X} x^T A y = \min_{x\in X} \max_{y\in Y} x^T A y,
\end{aligned}$$

see, e.g., [KMN09]. Thus the original matrix game corresponds to a problem of the form (3.1). To have a more clear picture of above framework, let us consider the following problem, which was studied in [KMN09] under the name of *mixed popular matchings*.

Let $G = (U \cup V, E)$ be a bipartite graph, and $r : E \subset U \times V \to \mathbb{Z}$ be a rank function that captures preferences of any vertex of U over the vertices in V that means for every $(u,v_1),(u,v_2) \in E$, $r(u,v_1) < r(u,v_2)$ if and only if u prefers v_1 to v_2. A U-matching $M : U \to V$ is an injective mapping such that $\{(u,M(u)) : u \in U\} \subseteq E$. Let

$$\mathcal{S} = \mathcal{T} = \{\{(u,M(u)) : u \in U\} : M \text{ is a } U\text{-matching of G}\} \subseteq 2^E.$$

Given $S, T \in \mathcal{S}$, define the quantity $\phi(S,T)$ as follows,

$$\phi(S,T) = |\{u \in U \ : \ r(u,S(u)) < r(u,T(u))\}|/|U|,$$

that is the fraction of the vertices of U that "prefers" S to T, and define $\mathcal{A}(S,T) = \phi(S,T) - \phi(T,S)$. It is well-known (see e.g. [GLS93]) that the convex hull of U-matchings has the linear description

$$X = Y = \{x \in \mathbb{R}_+^E : \sum_{(u,v)\in E} x_{u,v} = 1 \ \ \forall u \in U, \sum_{(u,v)\in E} x_{u,v} \leqslant 1 \ \ \forall v \in V\}.$$

Furthermore, if we define $A \in \mathbb{R}^{E\times E}$ to be the matrix with entries

$$a_{(u,v),(u',v')} = \begin{cases} \frac{1}{|U|} & \text{if } u = u' \text{ and } r(u,v) < r(u',v'), \\ -\frac{1}{|U|} & \text{if } u = u' \text{ and } r(u,v) > r(u',v'), \\ 0 & \text{otherwise,} \end{cases}$$

then for any $S, T \in \mathcal{S}$, we can write $\mathcal{A}(S,T) = x(S)^T A y(T)$, where $x(S),\ y(T) \in \{0,1\}^E$ are the characteristic vectors of S and T, respectively.

A matching S is said *popular* if $\phi(S,T) \geqslant \phi(T,S)$ for every matching $T \in \mathcal{S}$ and a *mixed matching* is simply a probability distributions over matchings in graph G. The function ϕ that compares two matchings generalizes in a natural manner to mixed matchings by taking expectation. A mixed matching p is popular if $\phi(p,q) \geqslant \phi(q,p)$ for all mixed matchings q. In [KMN09] they show that the given graph has a popular mixed matching if and only if

$$\min_{p\in\Delta_{\mathcal{S}}} \max_{q\in\Delta_{\mathcal{S}}} p^T \mathcal{A} q \leqslant 0,$$

where p and q are two mixed matching and $\Delta_{\mathcal{S}}$ is a set of probability distributions over a set of matchings in graph G. Note that in the above example, the problem can be written as a linear program of *polynomially* bounded size [KMN09]. However, this is not the case when the known linear descriptions of X and Y are not polynomially bounded, e.g., when in the above example G is a general *nonbipartite* graph. In this case finding a saddle-point may require the use of the Ellipsoid method, the sampling techniques of [BV04, KV06], or the use of our algorithm.

3.5.2. Linear Relaxation for Submodular Set Cover

Let us consider set-function $f : 2^{[n]} \to [0,1]$, which is defined power set of $[n]$. We say f is a *submodular* function, if for every $S, T \subseteq [n]$, we have that

$$f(S \cup T) + f(S \cap T) \leqslant f(S) + f(T).$$

Also we say f is *monotone*, if for every $S \subseteq T \subseteq [n]$, we have that $f(S) \leqslant f(T)$. Let us assume that f is a monotone submodular function and $S_1, \ldots, S_n \subseteq E$ are given subsets of a finite set E. With respect to sets S_i, $1 \leqslant i \leqslant n$, and E we define polytope P as follows,

$$P = \{x \in [0,1]^n : \sum_{i:S_i \ni e} x_i \geqslant 1 \text{ for all } e \in E\},$$

For every $X \subseteq [n]$, let $e(X) \in \{0,1\}^n$ denote an n-dimensional vector whose i-th coordinate is 1 if $i \in X$ and zero otherwise. *Submodular set covering problem* is defined as minimizing $f(X)$ subject to the constraint that the characteristic vector $e(X)$ belongs to a polytope $P \subseteq \mathbb{R}^n$. In what follows, we show how the problem can reduced to computing a saddle point for a corresponding function whit $\rho \leqslant 1$..

We define *polymatroid* P_f with respect to function f, as

$$P_f = \{y \in \mathbb{R}^n_+ : \ e(X)^T y \leqslant f(X) \text{ for all } X \subseteq [n]\}.$$

It is easy to see that that

$$f(X) = \max_{y \in P_f} e(X)^T y.$$

Thus we arrive at the following saddle point computation which provides a lower bound on the optimum submodular set cover,

$$\min_{x \in P} \max_{y \in P_f} x^T y.$$

Bibliography

[AHK05] S. Arora, E. Hazan, and S. Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *FOCS*, pages 339–348, 2005.

[AHK06] S. Arora, E. Hazan, and S. Kale. Multiplicative weights method: a meta-algorithm and its applications. Technical report, Princeton University, USA, 2006.

[AK07] S. Arora and S. Kale. A combinatorial, primal-dual approach to semidefinite programs. In *STOC*, pages 227–236, 2007.

[BBR04] Y. Bartal, J.W. Byers, and D. Raz. Fast, distributed approximation algorithms for positive linear programming with applications to flow control. *SIAM J. Comput.*, 33(6):1261–1279, 2004.

[Bel97] A. S. Belenky. A 2-person game on a polyhedral set of connected strategies. *Computers & Mathematics with Applications*, 33(6):99 – 125, 1997.

[BGT81] Robert G. Bland, Donald Goldfarb, and Michael J. Todd. Feature article-the ellipsoid method: A survey. *Operations Research*, 29(6):1039–1091, 1981.

[BI06] D. Bienstock and G. Iyengar. Approximating fractional packings and coverings in $O(1/\epsilon)$ iterations. *SIAM J. Comput.*, 35(4):825–854, 2006.

[Bro51] G.W. Brown. Iterative solution of games by fictitious play. *In: T.C. Koopmans, Editor, Activity Analysis of Production and Allocation*, pages 374–376, 1951.

[BV04] D. Bertsimas and S. Vempala. Solving convex programs by random walks. *J. ACM*, 51(4):540–556, 2004.

[CEF10] G. Christodoulou, K. Elbassioni, and M. Fouz. Truthful mechanisms for exhibitions. In *WINE*, pages 170–181, 2010.

[CV02] Robert D. Carr and Santosh Vempala. Randomized metarounding. *Random Struct. Algorithms*, 20(3):343–352, 2002.

[Dan63] G.B. Dantzig. *Linear Programming and extensions.* Princeton University Press, 1963.

[DJ07] F. Diedrich and K. Jansen. Faster and simpler approximation algorithms for mixed packing and covering problems. *Theor. Comput. Sci.*, 377:181–204, 2007.

[DKR91] A. Darte, L. Khachiyan, and Y. Robert. Linear scheduling is nearly optimal. *Parallel Processing Letters*, 1(2):73–81, 1991.

[DP14] Constantinos Daskalakis and Qinxuan Pan. A counter-example to karlin's strong conjecture for fictitious play. In *FOCS*, pages 11–20, 2014.

[DRVY11] Shaddin Dughmi, Tim Roughgarden, Jan Vondrák, and Qiqi Yan. An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries. *CoRR*, abs/1109.1053, 2011.

[EJ15] K. Elbassioni and M. Jha. On the power of combinatorial bidding in web display ads. In *ICIW*, 2015. To appear.

[EMMR14] K. Elbassioni, K. Makino, K. Mehlhorn, and F. Ramezani. On randomized fictitious play for approximating saddle points over convex sets. *Algorithmica*, 70(2):53–78, 2014.

[EMR15] K. Elbassioni, K. Mehlhorn, and F. Ramezani. Towards more practical linear programming-based techniques for algorithmic mechanism design. In *SAGT, to appear*, 2015.

[FS99] Y. Freund and R.E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[GK92] M.D. Grigoriadis and L.G. Khachiyan. Approximate solution of matrix games in parallel. In *Advances in Optimization and Parallel Computing*, pages 129–136, 1992.

[GK95] M.D. Grigoriadis and L.G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53 – 58, 1995.

[GK96] M.D. Grigoriadis and L.G. Khachiyan. Coordination complexity of parallel price-directive decomposition. *Math. Oper. Res*, 21(2):321–340, 1996.

[GK98] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *FOCS*, pages 300–309, 1998.

[GK04] N. Garg and R. Khandekar. Fractional covering with upper bounds on the variables: Solving lps with negative entries. In *ESA*, pages 371–382, 2004.

[GKPV01] M.D. Grigoriadis, L.G. Khachiyan, L. Porkolab, and J. Villavicencio. Approximate max-min resource sharing for structured concave optimization. *SIAM Journal on Optimization*, 41:1081–1091, 2001.

[GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, New York, 1988.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, second corrected edition, 1993.

[Gro67] H. Groemer. On the min-max theorem for finite two-person zero-sum games. *Probability Theory and Related Fields*, 9(1):59–61, 1967.

[Haz06] E. Hazan. *Efficient Algorithms for Online Convex Optimization and Their Application*. PhD thesis, Princeton University, USA, 2006.

[HKV11] M. Hoefer, T. Kesselheim, and B. Vöcking. Approximation algorithms for secondary spectrum auctions. In *SPAA*, pages 177–186, 2011.

[HS06] J. Hofbauer and S. Sorin. Best response dynamics for continuous zero-sum games. *Discrete and Continuous Dynamical Systems – Series B*, 6(1):215–224, 2006.

[Kal07] S. Kale. *Efficient Algorithms using the Multiplicative Weights Update Method*. PhD thesis, Princeton University, USA, 2007.

[KFB14] D. Kraft, S. Fadaei, and M. Bichler. Fast convex decomposition for truthful social welfare approximation. In *WINE*, pages 120–132, 2014.

[Kha79] L. G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk*, SSSR 244:1093–1096, 1979. translated in Soviet Mathematics Doklady 20, 191-194, 1979.

[Kha04] R. Khandekar. *Lagrangian Relaxation Based Algorithms for convex Programming Problems.* PhD thesis, Indian Institute of Technology, Delhi, India, 2004.

[KMN09] T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. In *ICALP (1)*, pages 574–584, 2009.

[KV06] A. Kalai and S. Vempala. Simulated annealing for convex optimization. *Math. Oper. Res.*, 31(2):253–266, 2006.

[KY07] C. Koufogiannakis and N.E. Young. Beating simplex for fractional packing and covering linear programs. In *FOCS*, pages 494–504, 2007.

[LN93] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *STOC*, pages 448–457, 1993.

[LS05] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, pages 595–604, 2005.

[LS11] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. *J. ACM*, 58(6):25, 2011.

[LV04] L. Lovász and S. Vempala. Hit-and-run from a corner. In *STOC*, pages 310–314, 2004.

[LV06] L. Lovász and S. Vempala. Fast algorithms for logconcave functions: Sampling, rounding, integration and optimization. In *FOCS*, pages 57–68, 2006.

[LV07] L. Lovász and S. Vempala. The geometry of logconcave functions and sampling algorithms. *Random Struct. Algorithms*, 30(3):307–358, 2007.

[LW94] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

[McL84] L. McLinden. A minimax theorem. *Mathematics of Operations Research*, 9(4):576–591, 1984.

[NN94] Yurii Nesterov and Arkadii Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* Society for Industrial and Applied Mathematics, 1994.

[NRTV07] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory.* Cambridge University Press, 2007.

[PST91] S.A. Plotkin, D.B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.

[Rob51] J. Robinson. An iterative method of solving a game. *The Annals of Mathematics*, 54(2):296–301, 1951.

[Roc70] R.T. Rockafellar. *Convex Analysis (Princeton Mathematical Series).* Princeton University Press, 1970.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.

[Seb90] Z. Sebestyén. A general saddle point theorem and its applications. *Acta Mathematica Hungarica*, 56(3-4):303–307, 1990.

[Sha58] H.N. Shapiro. Note on a computation method in the theory of games. *Communications on Pure and Applied Mathematics*, 11(4):587–593, 1958.

[Ter72] F. Terkelsen. Some minimax theorems. *Mathematica Scandinavica*, 31:405–413, 1972.

[tKP90] In-sook Kim and S. Park. Saddle point theorems on generalized convex spaces. *Journal of Inequalities and Applications*, 5(4):397–405, 1990.

[Vem05] S.S. Vempala. Geometric random walks: A survey. *Combinatorial and Computational Geometry, MSRI Publications*, 52:573–612, 2005.

[Vor84] N.N. Vorob'ev. *Foundations of game theory: noncooperative games*, volume 2. Birkhäuser, 1984.

[Wal45] A. Wald. Generalization of a theorem by v. Neumann concerning zero sum two person games. *The Annals of Mathematics*, 46(2):281–286, 1945.

[Was03] A.R. Washburn. *Two-Person Zero-Sum Games*. INFORMS, 2003.

[You01] N.E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.

A Multiplicative Weights Update Algorithms

The Multiplicative Weights Update Method (MWUM) is a general framework that has been widely applied in machine learning, optimization, and game theory. The method is based on an iterative and simple idea and is usually explained as a solution to a general problem, called *stock market prediction*. In this problem, there is a decision maker who wants to predict the stock market every day and n experts who announce their predictions at the beginning of each day. The decision maker's goal is to follow the prediction of an expert while minimizing the cost incurred by the chosen prediction. MWUM is a learning process that initially sets an equal weight for each expert and then, according to the value of each prediction, updates the weights at the end of the day. For example, if an expert gives a correct prediction, then his weight gets updated by a multiplicative factor $(1 + \epsilon)$, or similarly, by a multiplicative factor $(1 - \epsilon)$ for making a wrong prediction, where $0 < \epsilon < 1$ is a given parameter to the algorithm. Based on these weights, at the beginning of the next day, the decision maker follows the prediction of experts with higher weights. Intuitively, the method should perform well. For instance, consider a situation where there is a best expert with few mistakes. Then, in a long run, his weight mostly increases, and it is more likely that the decision maker follows him. In this chapter, we survey and review recent algorithms that are based on MWUW, which can also be found in [Kha04, Kal07]. In the last section, we also show that there exists an efficient algorithm for convex fractional packing. For the sake of completeness, we also present the correctness proofs for most of the algorithms.

Outline. In Sections A.1 and A.2, we explain algorithms for some variations of the stock market prediction problem. In Section A.3, we consider the feasibility check problem, which is a basic problem in optimization and show how to apply the MWUM for this problem. In Section A.4, we will see how MWUM is applied to get fast and efficient algorithm for the mixed packing and covering problem, which is a generaliza-

tion of packing and covering linear programming. In Subsections A.4.3, we apply the algorithm for the mixed problem to convex fractional packing problem and present an efficient algorithm for it. Throughout this chapter, we denote the inner product of two vectors a and b by $\langle a, b\rangle$.

A.1. The Weighted Majority Algorithm

In this section, we briefly discuss the weighted majority algorithm and the results concerning it. Consider a variant of the stock market prediction in which the prediction has the binary value, say Yes/No, and the quality of the algorithm is measured by the number of mistakes that are made over time. The weighted majority algorithm assigns equal weights $w_i = 1$ to each expert $i \in \{1, \ldots, n\}$. Then, at the end of each day, the algorithm updates the weight of each expert who predicts wrongly by a factor of $(1-\epsilon)$ and the decision maker follows the prediction with higher total weight. Formally, the algorithm proceeds as follows:

Algorithm 6. Weighted majority algorithm

Initialization: Fix an $\epsilon \leqslant \frac{1}{2}$. For each expert i, associate the weight $w_i^1 = 1$.
For $t = 1, 2, \ldots, T$:
1. Make the prediction that is the weighted majority of the experts' predictions based on the weights $w_1^t, \ldots, w_n^t$. That is, predict Yes or No depending on which prediction has a higher total weight of experts advising it (breaking ties arbitrarily).
2. For every expert i who predicts incorrectly, decrease his weight for the next round by multiplying it by a factor of $(1 - \epsilon)$:

$$w_i^{t+1} = (1 - \epsilon)w_i^t.$$

The following theorem derives an upper bound for mistakes m^T that are made by the algorithm after T rounds.

Theorem A.1.1. *[Kal07, Chapter 2] After T steps, let m_i^T denote the number of mistakes of expert i, and let m^T be the number of mistakes that Algorithm 6 has made. Then, we have the following bound for every i:*

$$m^T \leqslant \frac{2 \log n}{\epsilon} + 2(1 + \epsilon)m_i^T.$$

In particular, this holds for i, which is the best expert, i.e., having the least m_i^T.

The proof of the theorem is based on a potential function in t, which is defined as

$$\phi(t) = \sum_{i=1}^{n} w_i^t.$$

It is clear that $\phi(1) = n$, and if the algorithm makes a mistake on the $(T + 1)$-st day, then at least half of the total weight decreases by a factor of $(1 - \epsilon)$, and we have

$$\phi(T + 1) \leqslant \phi(T)/2 + \phi(T)(1 - \epsilon)/2 = \phi(T)(1 - \epsilon/2).$$

Hence, by an induction argument we have

$$\phi(T+1) \leqslant n(1-\epsilon/2)^{m^T}.$$

On the other hand, we have that for every expert $i \in \{1,\dots,n\}$, $w_i^T = (1-\epsilon)^{m_i^T} \leqslant \phi(T+1)$. We therefore have the following inequality proving the theorem.

$$(1-\epsilon)^{m_i^T} \leqslant n(1-\epsilon/2)^{m^T}.$$

A.2. The Multiplicative Weights Update Algorithm

In this section, we consider a more general model for the stock market prediction. We assume that the experts predict a real number in range $[-1,1]$ and also the prediction of expert i at the beginning of day $t \in \{1,\dots,T\}$ causes cost m_i^t in range $[-1,1]$, which is specified by the environment at the end of the day. Since each expert predicts a unique number which might be translated to an action, we cannot follow a weighted majority algorithm and we have to select exactly one expert and follow his advise. As before, the goal is to design an algorithm to help the decision maker to minimize his expenses. Here, we present a randomized algorithm which is similar to a weighted majority algorithm with two differences: First, we follow the advise of one expert that is selected according to a probability distribution over the experts. Note that, the probability destiny corresponding to each expert is proportional to his weight. Second, since our algorithm is randomized the quantity we are dealing with is the expected cost of the decision maker. Suppose that P^t denotes the probability distribution over the experts for t-th day and let $m^t \in [-1,1]^n$ be a vector whose i-th component indicating the cost caused by the i-th expert. Then the expected cost incurred by the decision maker in the t-th day, denoted by C_t, is

$$\mathbb{E}[C_t] = \langle m^t, P^t \rangle.$$

In the following we present the algorithm and a theorem that establishes an upper bound for the total expected cost.

Algorithm 7. Multiplicative Weights Algorithm

Initialization: Fix an $\epsilon \leqslant \frac{1}{2}$. For each expert i, associate the weight $w_i^t := 1$.
For $t = 1, 2, \dots, T$:
1. Choose expert i with probability proportional to his weight w_i^t, that is , distribution $P^t = \{w_1^t/\phi(t), \dots, w_n^t/\phi(t)\}$ where $\phi(t) = \sum_i w_i^t$.
2. Observe the costs of the experts m^t.
3. Penalize the costly experts by updating their weights as follows: for every expert i,

$$w_i^{t+1} := \begin{cases} w_i^t(1-\epsilon)^{m_i^t} & \text{if } m_i^t \geqslant 0 \\ w_i^t(1+\epsilon)^{-m_i^t} & \text{if } m_i^t < 0. \end{cases}$$

Theorem A.2.1. *[Kal07, Chapter 2] In the given setup, the Multiplicative Weights algorithm guarantees that after T rounds, for any expert i, we have*

$$\sum_{t=1}^{T}\langle m^t, P^t\rangle \leqslant \sum_{t=1}^{T} m_i^t + \epsilon\sum_{t=1}^{T}|m_i^t| + \frac{\log n}{\epsilon}.$$

Proof. Let us first mention some facts, which follow immediately from the convexity of the exponential function:

$$(1-\epsilon)^x \leqslant (1-\epsilon x) \text{ if } x \in [0,1],$$
$$(1+\epsilon)^{-x} \leqslant (1-\epsilon x) \text{ if } x \in [-1,0].$$

Let us define the potential function $\phi(t) = \sum_i w_i^t$. Since $m_i^t \in [-1,1]$, using the above facts we have

$$\begin{aligned}\phi(t+1) = \sum_i w_i^{t+1} &= \sum_{i:m_i^t \geqslant 0} w_i^t(1-\epsilon)^{m_i^t} + \sum_{i:m_i^t<0} w_i^t(1+\epsilon)^{-m_i^t}\\ &\leqslant \sum_i w_i^t(1-\epsilon m_i^t) = \phi(t) - \epsilon\phi(t)\sum_i m_i^t P_i^t\\ &= \phi(t)(1-\epsilon\langle m^t, P^t\rangle) \leqslant \phi(t)e^{-\epsilon\langle m^t, P^t\rangle},\end{aligned}$$

where, we used the fact that $P_i^t = \frac{w_i^t}{\phi(t)}$. Thus, by induction, after T rounds we have

$$\phi(T+1) \leqslant \phi(1)e^{-\epsilon\sum_{t=1}^{T}\langle m^t, P^t\rangle} = n\cdot e^{-\epsilon\sum_{t=1}^{T}\langle m^t, P^t\rangle}$$

Furthermore, for every expert i,

$$\phi(T+1) \geqslant w_i^{T+1} = (1-\epsilon)^{\sum_{t=2, m_i^t \geqslant 0}^{T} m_i^t}\cdot(1+\epsilon)^{-\sum_{t=2, m_i^t \leqslant 0}^{T} m_i^t}.$$

Now, by taking logarithms and applying the facts that $\log(1-\epsilon) \geqslant -(\epsilon+\epsilon^2)$, $\log(1+\epsilon) \geqslant \epsilon - \epsilon^2$, we get the desired inequality. □

A.3. The Approximation Algorithm for Feasibility Check

In this section, we apply MWUM for a fundamental problem in optimization that is the feasibility check of a given convex set; by either finding a feasible point in the set or reporting that the set is empty. Let us consider convex set $Q \subseteq \mathbb{R}^n$ which is defined as

$$\mathcal{Q} := \{x \in \mathcal{P} \subset \mathbb{R}^n : Ax \geqslant b\}, \tag{A.1}$$

where A is an $m \times n$ matrix, $b \in \mathbb{R}^m$ and $\mathcal{P}$ is a convex set in $\mathbb{R}^n$. Usually, set $\mathcal{P}$ is considered as a set of "easy" constraints, for example non-negativity. In order to approximately solve the feasibility check problem, we slightly relax the constraints by allowing an additive small error $\delta > 0$ for each constraint, that is, point $x \in \mathbb{R}^n$ is approximately feasible if and only if for all i,

$$x \in \mathcal{P} \text{ and } A_i x \geqslant b_i - \delta,$$

where A_i is the i-th row of A. Our goal is to design an algorithm that either outputs an approximately feasible point in $\mathcal{Q}$ with small error or shows that $\mathcal{Q}$ is infeasible. Let us assume that an oracle exists so that for a given a probability vector p on the m constraints, it either finds an x satisfying the following constraint

$$x \in \mathcal{P} \text{ and } \langle p^T, Ax \rangle \geqslant \langle p^T, b \rangle, \tag{A.2}$$

or outputs there is no such x. It is reasonable to expect such an optimization procedure to exist since we only need to check the feasibility of one constraint rather than m. It is clear that if $x \in \mathcal{Q}$, then for every probability distribution p we have $p^T Ax \geqslant p^T b$. So if we can find a probability distribution p so that there is no x satisfying (A.2), then we conclude that Q is an empty set. In what follows we define an oracle with some certain properties which is useful in design of our algorithm.

Definition A.3.1. *[Kal07, Chapter 2] An (l, ρ)-bounded oracle, for parameters $0 \leqslant l \leqslant \rho$, is an algorithm which given a probability vector p over the constraints, solves the feasibility problem (A.2). Furthermore, there is a fixed subset $I \subseteq [m]$ of constraints such that whenever the oracle manages to find a point $x \in \mathcal{P}$ satisfying (A.2), the following holds:*

$$\forall i \in I : A_i x - b_i \in [-l, \rho]$$

$$\forall i \notin I : A_i x - b_i \in [-\rho, l]$$

The value ρ is called the **width** *of the problem.*

Now, we adjust Multiplicative Weights Update Algorithm 7 for this problem and show the following theorem.

Theorem A.3.2. *[Kal07, Chapter 2] Let $\delta > 0$ be a given error parameter. Suppose there exists an (l, ρ)-bounded oracle for the feasibility problem (A.2). Assume that $l \geqslant \frac{\delta}{2}$. Then there is an algorithm which either solves the problem up to an additive error of δ, or correctly concludes that the system is infeasible, making only $O\left(\frac{l\rho \log m}{\delta^2}\right)$ calls to the oracle.*

Proof. The condition $l \geqslant \frac{\delta}{2}$ is only technical, and if it is not met we can just redefine l to be $\frac{\delta}{2}$. To prove the theorem, we adopt Algorithm 7. Let us assume that each expert represents one of the m constraints and events correspond to vectors in $\mathcal{P}$. The loss of the expert corresponding to i-th constraint for event x is $(A_i x - b_i)/\rho$, which is a number lying in the range $[-1, 1]$. In each round t, given a distribution over the experts (i.e., the constraints) p^t, we run the oracle with p^t. If the oracle declares that there is no $x \in \mathcal{Q}$ such that $\langle p^{t^T}, Ax \rangle \geqslant \langle p^{t^T}, b \rangle$, then we stop, because now p^t proves that (A.1) is infeasible. So let us assume that this does not happen, i.e., in each round t, the oracle manages to find a solution x^t such $\langle p^{t^T}, Ax \rangle \geqslant \langle p^{t^T}, b \rangle$. Recall that the the cost vector was specified to be $m^t := (Ax^t - b)/\rho$, so we conclude that the expected cost in each round is non-negative and is computed as

$$\langle m^t, p^t \rangle = \frac{1}{\rho} \langle Ax^t - b, p^t \rangle = \frac{1}{\rho} (p^{t^T} Ax^t - p^{t^T} b) \geqslant 0.$$

Let us fix an arbitrary $i \in I$. Then Theorem A.2.1 tells us that after T rounds,

$$\begin{aligned} 0 \leqslant \langle m^t, p^t \rangle &\leqslant \sum_{t=1}^{T} (A_i x^t - b_i)/\rho + \epsilon \sum_{t=1}^{T} |A_i x^t - b_i|/\rho + \frac{\log m}{\epsilon} \\ &= (1+\epsilon) \sum_{t=1}^{T} (A_i x^t - b_i)/\rho + 2\epsilon \sum_{<0} |A_i x^t - b_i|/\rho + \frac{\log m}{\epsilon} \\ &\leqslant (1+\epsilon) \sum_{t=1}^{T} (A_i x^t - b_i)/\rho + \frac{2\epsilon l}{\rho} T + \frac{\log m}{\epsilon} \end{aligned}$$

Here, the subscript <0 refers to the rounds in which $A_i x^t - b_i < 0$. The last inequality follows because if $A_i x^t - b_i < 0$, then $|A_i x^t - b_i| \leqslant l$. Dividing by T, multiplying by ρ, and letting $\bar{x} = \frac{1}{T} \sum_{t=1}^{T} x^t$ (note that $\bar{x} \in \mathcal{Q}$ since $\mathcal{Q}$ is a convex set), we get that

$$0 \leqslant (1+\epsilon)(A_i \bar{x} - b_i) + 2\epsilon l + \frac{\rho \log m}{\epsilon T}$$

Now, if we choose $\epsilon = \frac{\delta}{4l}$ (note that $\epsilon \leqslant \frac{1}{2}$ since $l \geqslant \frac{\delta}{2}$), and $T = \lceil \frac{8l\rho \log m}{\delta^2} \rceil$, we get that

$$0 \leqslant (1+\epsilon)[A_i \bar{x} - b_i] + \delta \rightarrow A_i \bar{x} \geqslant b_i - \delta.$$

Reasoning similarly for $i \notin I$, we get the same inequality. Putting both together, we conclude that $\bar{x}$ satisfies the feasibility problem $A.1$ up to an additive δ factor, as desired. □

A.3.1. Covering Linear Programming

In this subsection, we show how MWUM can be applied for solving covering linear programming is a basic optimization problem that is usually formulated as follows:

$$\begin{array}{lrcl} \min & c^T x & & \\ s.t & Ax & \geqslant & b \\ & x & \geqslant & 0, \end{array}$$

where entries in matrix A, b and c are non-negative. It is not hard to see that by adding the constraint $c^T x \geqslant b'$, for some real number $b' \geqslant 0$ to the set of constraints, we can reduce the covering problem to a feasibility check problem. Note that by performing binary search over the possible values of b' we can find the optimal b'. Then finding a solution to the covering problem corresponds to the feasibility check of the following set:

$$\mathcal{Q} := \{x \in \mathcal{P} \subset \mathbb{R}^n : c^T x \geqslant b' \text{ and } Ax \geqslant b\},$$

where $\mathcal{P}$ denote the non-negativity constraints. By appropriate scaling of the inequalities, we assume that for every i, $b_i = 1$. Since each entry of A is non-negative, for every $x \in \mathcal{P}$ we have $Ax \geqslant 0$ and hence $A_i x - b_i \geqslant -1$. Let us assume that there is a $(1, \rho)$-bounded oracle for this problem. Now, applying Theorem A.3.2, we get the following theorem:

Theorem A.3.3. *[Kal07, Chapter 2] Suppose there exists a $(1, \rho)-$bounded oracle for the program $Ax \geqslant b$ with $x \in \mathcal{P}$. Given an error parameter $\delta > 0$, there is an algorithm which computes a δ-approximate solution to the program, or correctly concludes that it is infeasible, using $O(\rho \log m/\delta^2)$ calls to the oracle.*

Since reducing our covering problem to the corresponding feasibility check problem requires a binary search, we have to repeat Algorithm 7 as many as times the binary search needs.

A.4. The Width-Independent Algorithm for Optimization Problems

In this section, we present a width-independent algorithm for several class of optimization problems which appeared in Khandekar's Thesis [Kha04]. Before we show the results, we need to have some notations and definitions:

Definition A.4.1. *[Kha04, Chapter 5] (Mixed Problem). Given a non-empty convex set $\mathcal{Q} \subseteq \mathbb{R}^n$, non-negative continuous convex functions $f_i : \mathcal{Q} \to \mathbb{R}_+$ for $1 \leqslant i \leqslant m$, and non-negative continuous concave functions $g_i : \mathcal{Q} \to \mathbb{R}_+$ for $1 \leqslant i \leqslant k$, a mixed problem seeks either to find an $x \in \mathcal{Q}$ such that*

$$\max_{1 \leqslant i \leqslant m} f_i(x) \leqslant \min_{1 \leqslant j \leqslant k} g_j(x),$$

or to prove that no such $x \in \mathcal{Q}$ exists.

Note that, this problem is called mixed as it contains both a packing component $\max_i f_i(x)$ and a covering component $\min_j g_j(x)$. For a non-zero vector $v \geqslant 0$, the vector $\underline{v}$ denotes v after normalization, that is , $\underline{v} = v/\sum_i v_i$. In order to solve the mixed problem, we assume that there is an oracle to solve the following problem.

Definition A.4.2. *[Kha04, Chapter 5] (Oracle). Given $y \in \mathbb{R}^m_+$ and $z \in \mathbb{R}^k_+$, the oracle either finds an $x \in \mathcal{Q}$ such that*

$$\langle \underline{y}, f(x) \rangle \leqslant \langle \underline{z}, g(x) \rangle, \tag{A.3}$$

or proves that no $x \in \mathcal{Q}$ such exists.

A linear version of the mixed problem was first considered by Young [You01] but Khandekar [Kha04] generalized the result to arbitrary convex/concave functions over convex sets.

Outline. In Subsection A.4.1, we explain a basic problem, so-called *online prediction*, and prove that there exists an algorithm that solves the problem. In Subsection A.4.2, based on the result concerning the Online Prediction problem we present an algorithm for the mixed problem which is independent of the width parameter. In Subsection A.4.3, applying the general framework for solving the mixed problem, we present an efficient algorithm for convex packing problem.

A.4.1. Online Prediction Problem

In the online prediction problem, there is an agent who has m different strategies to choose from. The agent is involved in a prediction that goes over many rounds. In round r, the agent decides a probability distribution $P^r = (P_1^r, \cdots, P_m^r)$ over the strategies. In round r, each strategy earns a profit $\mathcal{P}_i^r$ and incurs a loss $\mathcal{L}_i^r$ that is determined by the environment. The income made by the i-th strategy is then $\mathcal{I}_i^r = \mathcal{P}_i^r - \mathcal{L}_i^r$. The income earned by the agent in round r is the weighted sum

$$\langle P^r, \mathcal{I}^r \rangle = \sum_i P_i^r \mathcal{I}_i^r = \sum_i P_i^r (\mathcal{P}_i^r - \mathcal{L}_i^r).$$

We also assume that income $\mathcal{I}_i$, for every $1 \leqslant i \leqslant m$, lies in range $[-1, 1]$. The goal of the agent is to maximize her cumulative income relative to the income of the best strategy. Let N denote the total number of rounds. The agent chooses a distribution P^r in round r based on the knowledge of the vectors $\mathcal{P}^s, \mathcal{L}^s$ in all previous rounds $1 \leqslant s < r$. The agent's goal is to earn a cumulative income that is as close to the best strategy as possible. Let us define

$$\mathcal{P} = \sum_{r=1}^{N} \sum_{i=1}^{m} P_i^r \mathcal{P}_i^r \quad \text{and} \quad \mathcal{L} = \sum_{r=1}^{N} \sum_{i=1}^{m} P_i^r \mathcal{L}_i^r,$$

to be the cumulative profit and loss made by the agent in N rounds, respectively. Let $\mathcal{I} = \mathcal{P} - \mathcal{L}$ denote the net cumulative income of the agent and $\mathcal{I}_i = \sum_{r=1}^{N} (\mathcal{P}_i^r - \mathcal{L}_i^r)$ be the cumulative income of the i-th strategy in N rounds. We have the following theorem.

Theorem A.4.3. *[Kha04, Chapter 2] Given any $\epsilon \in (0, 1)$, there exists a deterministic algorithm for choosing the distributions such that*

$$\max_{i \in [m]} \mathcal{I}_i \leqslant (e^{\epsilon} \mathcal{P} - e^{-\epsilon} \mathcal{L}) + \frac{\log m}{\epsilon}$$

Proof. Let $x_i^r = \sum_{s=1}^{r-1} \mathcal{I}_i^s$ and define distribution P_i^r in each round r as follows:

$$P_i^r = e^{\epsilon x_i^r} / (e^{\epsilon x_1^r} + e^{\epsilon x_2^r} + \cdots + e^{\epsilon x_m^r}).$$

Clearly, for every r we have $P_i^r \geqslant 0$ and $\sum_{i=1}^{m} P_i^r = 1$, so this is a well-defined probability distribution. Using inequality $e^z - 1 \leqslant z e^z$, assumption $-1 \leqslant \mathcal{I}_i^r \leqslant 1$ and substituting $\mathcal{I}_i^r = \mathcal{P}_i^r - \mathcal{L}_i^r$, we have

$$\begin{aligned}
e^{\epsilon x_i^{r+1}} - e^{\epsilon x_i^r} &= e^{\epsilon \sum_{s=1}^{r} \mathcal{I}_i^s} - e^{\epsilon \sum_{s=1}^{r-1} \mathcal{I}_i^s} = e^{\epsilon \sum_{s=1}^{r-1} \mathcal{I}_i^s} (e^{\epsilon \mathcal{I}_i^r} - 1) \\
&\leqslant e^{\epsilon x_i^r} \cdot \epsilon \cdot \mathcal{I}_i^r \cdot e^{\epsilon \mathcal{I}_i^r} = \epsilon \cdot (\mathcal{P}_i^r - \mathcal{L}_i^r) e^{\epsilon \mathcal{I}_i^r} \cdot e^{\epsilon x_i^r} \\
&= \epsilon (\mathcal{P}_i^r e^{\epsilon \mathcal{I}_i^r} - \mathcal{L}_i^r e^{\epsilon \mathcal{I}_i^r}) e^{\epsilon x_i^r} \\
&\leqslant \epsilon (e^{\epsilon} \mathcal{P}_i^r - \mathcal{L}_i^r e^{-\epsilon}) e^{\epsilon x_i^r}.
\end{aligned}$$

This implies that

$$(e^{\epsilon} \mathcal{P}_i^r - e^{-\epsilon} \mathcal{L}_i^r) e^{\epsilon x_i^r} \geqslant \frac{e^{\epsilon x_i^{r+1}} - e^{\epsilon x_i^r}}{\epsilon} \tag{A.4}$$

Therefore, by substituting distribution P_i^r and using Inequality $(A.4)$, we get

$$\begin{aligned} e^{\epsilon}\mathcal{P} - e^{-\epsilon}\mathcal{L} &= \sum_{r=1}^{N}(e^{\epsilon}\sum_{i=1}^{m} P_i^r\mathcal{P}_i^r - e^{-\epsilon}\sum_{i=1}^{m} P_i^r\mathcal{L}_i^r) \\ &= \sum_{r=1}^{N}\frac{\sum_{i=1}^{m}(e^{\epsilon}\mathcal{P}_i^r - e^{-\epsilon}\mathcal{L}_i^r)\cdot e^{\epsilon x_i^r}}{e^{\epsilon x_1^r} + e^{\epsilon x_2^r} + \cdots + e^{\epsilon x_m^r}} \\ &\geqslant \sum_{r=1}^{N}\sum_{i=1}^{m}\frac{e^{\epsilon x_i^{r+1}} - e^{\epsilon x_i^r}}{\epsilon(e^{\epsilon x_1^r} + e^{\epsilon x_2^r} + \cdots + e^{\epsilon x_m^r})}. \end{aligned}$$

We also have

$$\begin{aligned} \sum_{r=1}^{N}\sum_{i=1}^{m}\frac{e^{\epsilon x_i^{r+1}} - e^{\epsilon x_i^r}}{\epsilon(e^{\epsilon x_1^r} + e^{\epsilon x_2^r} + \cdots + e^{\epsilon x_m^r})} &= \frac{1}{\epsilon}\sum_{r=1}^{N}\left(\frac{\sum_{i=1}^{m} e^{\epsilon x_i^{r+1}}}{\sum_{i=1}^{m} e^{\epsilon x_i^r}} - \frac{\sum_{i=1}^{m} e^{\epsilon x_i^r}}{\sum_{i=1}^{m} e^{\epsilon x_i^r}}\right) \\ &= \frac{1}{\epsilon}\sum_{r=1}^{N}\left(\frac{\sum_{i=1}^{m} e^{\epsilon x_i^{r+1}}}{\sum_{i=1}^{m} e^{\epsilon x_i^r}} - 1\right) \\ &\geqslant \frac{1}{\epsilon}\sum_{r=1}^{N}\log\frac{\sum_{i=1}^{m} e^{\epsilon x_i^{r+1}}}{\sum_{i=1}^{m} e^{\epsilon x_i^r}} = \frac{1}{\epsilon}\log\frac{\sum_{i=1}^{m} e^{\epsilon x_i^{N+1}}}{\sum_{i=1}^{m} e^{\epsilon x_i^1}}, \end{aligned}$$

where the last inequality follows from inequality $z - 1 \geqslant \log z$ and telescoping sum. Finally we get

$$\begin{aligned} e^{\epsilon}\mathcal{P} - e^{-\epsilon}\mathcal{L} &\geqslant \frac{1}{\epsilon}\log\frac{\sum_{i=1}^{m} e^{\epsilon x_i^{N+1}}}{m} = \frac{1}{\epsilon}\log\frac{\sum_{i=1}^{m} e^{\epsilon\mathcal{I}_i}}{m} \\ &= \frac{1}{\epsilon}\log(\sum_{i=1}^{m} e^{\epsilon\mathcal{I}_i}) - \frac{\log m}{\epsilon} \geqslant \frac{1}{\epsilon}\log\max_{i\in[m]} e^{\epsilon\mathcal{I}_i} - \frac{\log m}{\epsilon} \\ &= \frac{1}{\epsilon}\log e^{\max_{i\in[m]}\epsilon\mathcal{I}_i} - \frac{\log m}{\epsilon} = \max_{i\in[m]}\mathcal{I}_i - \frac{\log m}{\epsilon} \end{aligned}$$

□

Corollary A.4.4. *[Kha04, Chapter 2] If $\mathcal{L}_i^r = 0$ for all $i \in [m], r \in [N]$*

$$\max_{i\in[m]}\mathcal{P}_i \leqslant e^{\epsilon}\mathcal{P} + \frac{\log m}{\epsilon}$$

Proof. Follows directly from Theorem A.4.3, if we set $\mathcal{I}_i = \mathcal{P}_i$ and $\mathcal{L} = 0$. □

Corollary A.4.5. *[Kha04, Chapter 2] If $\mathcal{P}_i^r = 0$ for all $i \in [m], r \in [N]$*

$$\min_{i\in[m]}\mathcal{L}_i \geqslant e^{-\epsilon}\mathcal{L} - \frac{\log m}{\epsilon}$$

Proof. Follows directly from Theorem A.4.3, if we set $\mathcal{I}_i = -\mathcal{L}_i$ and $\mathcal{P} = 0$. □

A.4.2. A Width-Independent Algorithm for Mixed Problem

In this subsection, we mainly focus on the mixed problem and present an algorithm for it. We also show the following theorem deriving an upper bound for the number of iterations and the error.

Algorithm 8. A width independent algorithm for mixed problem.

Algorithm MIXED(width independent)
Input: Convex set $\mathcal{Q} \subseteq \mathbb{R}^n$, functions $f : \mathcal{Q} \to \mathbb{R}^m_+, g : \mathcal{Q} \to \mathbb{R}^k_+$, and $w \in (0, 1)$
Output: infeasible or $\tilde{x} \in \mathcal{Q}$

1. **Initialize**
2. $r := 0$ {initialize the round number}
   Repeat
3. $r := r + 1$ {round r begins}
4. Find $x_r \in \mathcal{Q}$ such that $\langle \underline{y}, f(x_r)\rangle \leqslant \langle \underline{z}, g(x_r)\rangle$. {oracle call}
   If there is no such x_r, then output *infeasible*
5. $w_r := 1/\max\{\max_{i\in[m]} f_i(x_r), \max_{j:z_j>e^{-\epsilon\Gamma}} g_j(x_r)\}$ {pick x_r to an extent w_r}
6. **Update** (w_r, y, z, x_r)
7. Until **Stop-condition** is True {round r ends}
8. Output $\tilde{x} = \frac{\sum_{s=1}^{r} w_s x_s}{\sum_{s=1}^{r} w_s}$

Initializing Subroutine:

Set $\epsilon = \dfrac{w}{3}$
For $i \in [m]$ do $y_i := 1$
For $j \in [k]$ do $z_j := 1$ initializing the dual variables
$\Gamma := 2\epsilon^{-2}(\log m + e^{2\epsilon} \log k)$

Update (w, y, z, x) **Subroutine:**

For $i \in [m]$ do $y_i^r := y_i.e^{\epsilon w f_i(x)}$
For $j \in [k]$ do
If $z_j > e^{-\epsilon\Gamma}$, then $z_j := z_j \cdot e^{-\epsilon w g_j(x)}$ dual variable updated

Stop-Condition Subroutine:

True, if $\max_{i\in[m]} \sum_{s=1}^{r} w_s f_i(x_s) \geqslant \Gamma$ stop if some f_i is packed enough

True, if $\min_{j\in[k]} \sum_{s=1}^{r} w_s g_j(x_s) \geqslant \Gamma$ stop if all g_j is covered enough

False, otherwise

Theorem A.4.6. *[Kha04, Chapter 5] There exist an algorithm for the mixed problem that given an error parameter $w \in (0,1)$ either computes an $x \in \mathcal{Q}$ such that*

$$\max f_i(x)_{i\in[m]} \leqslant e^w \min_{j\in[k]} g_j(x),$$

or proves that there is no x such that $f_i(x) \leqslant g_j(x)$ for all i and j. The algorithm makes $O((m+k)w^{-2}\log(m+k))$ calls to the oracle (A.3) and takes additional time that is needed to compute $f(x)$ and $g(x)$ for some $x \in \mathcal{Q}$ between successive oracle calls.

Proof. We prove that Algorithm 8 satisfies the claims in the statement of the theorem. Consider the following instance of online prediction problem. Suppose there is an agent with m strategies corresponding to functions $f_1, \ldots, f_m$ and has total N rounds. Let in round r, the i-th strategy earns a profit of $w_r f_i(x_r)$ and incurs a loss of zero, $\mathcal{L}_i^r = 0$. Assume the agent chooses a distribution $\{\underline{y}^r\}$ in round r which is exactly like distribution P_i^r in Theorem A.4.3, $y_i^r = e^{\sum_{s=1}^{r-1} \mathcal{I}_i^s} = e^{\sum_{s=1}^{r-1} \epsilon w_s f_i(x_s)}$. Therefore, by Corollary A.4.4 we have,

$$\max_{i\in[m]} \mathcal{P}_i \leqslant e^{\epsilon}\mathcal{P} + \frac{\log m}{\epsilon}.$$

Hence,

$$\begin{aligned}\max_{i\in[m]} \sum_{r=1}^{N} w_r f_i(x_r) &\leqslant e^{\epsilon} \sum_{r=1}^{N}\sum_{i=1}^{m} \underline{y}_i^r . w_r f_i(x_r) + \frac{\log m}{\epsilon} \\ &= e^{\epsilon} \sum_{r=1}^{N} w_r \langle \underline{y}^r, f(x_r)\rangle + \frac{\log m}{\epsilon}. \end{aligned} \tag{A.5}$$

Let us consider another instance of online prediction problem. There is an agent who has k strategies corresponding functions $g_1, g_2, \ldots, g_k$ and has total N round. The j-th strategy in round r incurs profit of zero and loss of $w_r g_j(x_r)$ if $z_j^r > e^{-\epsilon\Gamma}$ and zero otherwise. Agent chooses a distribution $\{\underline{z}^r\}$ in round r like distribution P_i^r in Theorem A.4.3, where

$$z_j^r = e^{\sum_{s=1}^{r-1} \mathcal{I}_j^s} = e^{-\sum_{s=1}^{r-1} \epsilon w_s g_j(x_s)}.$$

By applying Corollary A.4.5 we have

$$\begin{aligned}\min_{j\in[k]}\sum_{r=1}^{N} w_r g_j(x_r) &\geqslant e^{-\epsilon}\sum_{r=1}^{N}\langle \underline{z}^r, w_r g(x_r)\rangle - \frac{\log k}{\epsilon}\\ &= e^{-\epsilon}\sum_{r=1}^{N} w_r\langle \underline{z}^r, g(x_r)\rangle - \frac{\log k}{\epsilon}. \end{aligned} \tag{A.6}$$

Since in each round r, we choose x_r from Oracle A.3, we have $\langle \underline{y}^r, f(x_r)\rangle \leqslant \langle \underline{z}^r, g(x_r)\rangle$ and also,

$$\sum_{r=1}^{N} w_r\langle \underline{y}^r, f(x_r)\rangle \leqslant \sum_{r=1}^{N} w_r\langle \underline{z}^r, g(x_r)\rangle. \tag{A.7}$$

By Inequalities $(A.5), (A.6)$ and $(A.7)$ we have that

$$\begin{aligned}\max_i\sum_{r=1}^{N} w_r f_i(x_r) &\leqslant e^{\epsilon}\sum_{r=1}^{N} w_r\langle \underline{y}^r, f(x_r)\rangle + \frac{\log m}{\epsilon}\\ &\leqslant e^{\epsilon}\sum_{r=1}^{N} w_r\langle \underline{z}^r, g(x_r)\rangle + \frac{\log m}{\epsilon}\\ &\leqslant e^{2\epsilon}\min_j\sum_{r=1}^{N} w_r g_j(x_r) + \left(\frac{\log m + e^{2\epsilon}\log k}{\epsilon}\right)\\ &= e^{2\epsilon}\min_j\sum_{r=1}^{N} w_r g_j(x_r) + \frac{\epsilon\Gamma}{2}. \end{aligned} \tag{A.8}$$

Based on **Stop-Condition**, we may consider two cases:

1. $\max_i \sum_{r=1}^{N} w_r \cdot f_i(x_r) \geqslant \Gamma$
2. $\min_j \sum_{r=1}^{N} w_r \cdot g_j(x_r) \geqslant \Gamma$

In the first case, by using inequality $1-\epsilon/2 \geqslant e^{-\epsilon}$, for every $\epsilon \leqslant 1/2$, we have

$$\begin{aligned}\max_i\sum_{r=1}^{N} w_r f_i(x_r) - \frac{\epsilon\Gamma}{2} &\geqslant \max_i\sum_{r=1}^{N} w_r f_i(x_r) - \frac{\epsilon}{2}\max_i\sum_{r=1}^{N} w_r f_i(x_r)\\ &= (1-\epsilon/2)\max_i\sum_{r=1}^{N} w_r f_i(x_r)\\ &\geqslant e^{-\epsilon}\max_i\sum_{r=1}^{N} w_r f_i(x_r). \end{aligned} \tag{A.9}$$

In the second case, by using the inequality $1 + \epsilon \leqslant e^{\epsilon}$, we have

$$\begin{aligned} e^{2\epsilon} \min_j \sum_{r=1}^{N} w_r g_j(x_r) + \frac{\epsilon\Gamma}{2} &\leqslant e^{2\epsilon} \min_j \sum_{r=1}^{N} w_r g_j(x_r) + \frac{\epsilon}{2} \min_j \sum_{r=1}^{N} w_r g_j(x_r) \\ &\leqslant (1+\epsilon) e^{2\epsilon} \min_j \sum_{r=1}^{N} w_r g_j(x_r) \\ &\leqslant e^{3\epsilon} \min_j \sum_{r=1}^{N} w_r g_j(x_r). \end{aligned} \tag{A.10}$$

For both cases, by Inequalities (A.8), (A.9) and $(A.10)$ we get

$$\max_i \sum_{r=1}^{N} w_r f_i(x_r) \leqslant e^{3\epsilon} \min_j \sum_{r=1}^{N} w_r g_j(x_r). \tag{A.11}$$

Let $\tilde{x} := \frac{\sum_{r=1}^{N} w_r x_r}{\sum_{r=1}^{N} w_r}$ and since $\mathcal{Q}$ is convex, we have $\tilde{x} \in \mathcal{Q}$. Therefore, by Inequality (A.11), the convexity of all f_i and the concavity of all g_j we have

$$\begin{aligned} \max_i f_i(\tilde{x}) &= \max_i f_i \left(\frac{\sum_{r=1}^{N} w_r x_r}{\sum_{r=1}^{N} w_r} \right) \\ &\leqslant \max_i \frac{\sum_{r=1}^{N} w_r f_i(x_r)}{\sum_{r=1}^{N} w_r} \leqslant e^{3\epsilon} \frac{\min_j \sum_{r=1}^{N} w_r g_j(x_r)}{\sum w_r} \\ &\leqslant e^{3\epsilon} \min_j g_j \left(\frac{\sum_{r=1}^{N} w_r x_r}{\sum_{r=1}^{N} w_r} \right) = e^{3\epsilon} \min_j g_j(\tilde{x}). \end{aligned} \tag{A.12}$$

So output $\tilde{x}$ satisfies the statement of in Theorem A.4.6. The proof of the running time proceeds as follows: For each round r, let us define the following set and functions: $A := \{j \in [k] | \sum_{s=1}^{r} w_s g_j(x_s) < \Gamma\}$, which denotes the set of functions g_j that are not yet covered to an extent Γ, $F_i = \sum_{s=1}^{r} w_s f_i(x_s)$ and $G_j = \sum_{s=1}^{r} w_s g_j(x_s)$. Note that the values of F_i and G_j never decrease as the algorithm progresses. Now let us carefully consider the way we have defined the weight w_r in line (5) of Algorithm 8 and the way we update the dual variables in Update subroutine (see at the end of this subsection). If $j \in A$, we have $G_j < \Gamma$. Therefore, the current value of z_j is $e^{-\epsilon G_j} > e^{-\epsilon\Gamma}$ and it affects $w_r = 1/\max\{\max_i f_i(x_r), \max_{z_j > e^{-\epsilon\Gamma}} g_j(x_r)\}$. Therefore, in any round r we have, either A decreases in size by at least one, or at least one of $\sum_{i=1}^{m} F_i$ and $\sum_{j \in A} G_j$ increases by at least one. We exit the repeat loop as soon as either $\max_i F_i \geqslant \Gamma$ or A becomes empty (see Stop-Condition subroutine at the end of this subsection). Therefore, the maximum value that any F_i can take is $\Gamma+1$. We have added 1 to Γ, because each F_i increases in steps of at most 1. Thus the maximum value that $\sum_{i=1}^{m} F_i$ can take is $m(\Gamma+1)$. Now, as soon as $G_j \geqslant \Gamma$, the index j is removed from A. Thus for any j, the value of G_j can rise at most up to $\Gamma+1$ while $j \in A$. Therefore, the total number of rounds is at most $(m+k)(\Gamma+1) = O((m+k)w^{-2}\log(m+k))$. Since the number of oracle calls is exactly equal to the number of rounds and $w = \epsilon/3$, the proof is complete. □

A.4.3. Convex Fractional Packing Problem

In this subsection we first define convex packing problem and then by applying Theorem A.4.6 we show our result.

Definition A.4.7 (Convex Fractional Packing Problem)**.** *A convex fractional packing problem is defined as follows:*

$$\begin{aligned} \max\ & c^T x \\ s.t\quad & h_i(x) \leqslant b_i \qquad\qquad i \in [m] \\ & x \geqslant 0 \end{aligned} \tag{A.13}$$

where $h_i, 1 \leqslant i \leqslant m$ is a convex function and $c, b \geqslant 0$ are vectors with non-negative entries. We also assume that for every i, $h_i(0) = 0$.

Theorem A.4.8. *Suppose that $w \in (0,1)$ is a given error parameter. Then, there is an algorithm that computes an $x \geqslant 0$, which is a feasible solution to the convex fractional packing problem and satisfies*

$$c^T x \geqslant e^{-2w} \cdot \mu^*$$

where μ^* is the optimum value of convex packing problem. The algorithm makes $O(\tau m w^{-2} \log m)$ calls to the oracle, where m is the number constraints and τ is the time required for binary search over the all possible values taken by $c^T x$.

Proof. We first reduce convex fractional packing problem to a feasibility problem by adding one more constraint, that is, $c^T x \geqslant \mu$, where μ is a constant given by binary search.The feasibility problem is defined as follows:

$$h_i(x) \leqslant b_i \qquad i \in [m] \tag{A.14}$$

$$c^T x \geqslant \mu \tag{A.15}$$

$$x \geqslant 0$$

Now, let us set $f_i(x) := h_i(x)/b_i$ for every $1 \leqslant i \leqslant m$, $f_{m+1}(x) = 1$, $g_1(x) = 1$ and $g_2(x) = c^T x/\mu$. Clearly, we have that f_i's are convex and g_i's are concave. For a given error parameter $w \in [0,1]$, consider the following mixed problem,

$$\max_{i \in [m+1]} f_i(x) \leqslant \min_{j \in [2]} g_j(x),$$

By Theorem A.4.6, we know that Algorithm 8 either finds an x that satisfies

$$\max_{i \in [m+1]} f_i(x) \leqslant e^w \min_{j \in [2]} g_j(x), \tag{A.16}$$

or says that there is no x satisfying

$$\max_{i \in [m+1]} f_i(x) \leqslant \min_{j \in [2]} g_j(x).$$

In the following we show that Algorithm 8 solves our feasibility problem as well. So, we may consider two cases:

Case 1. The algorithm outputs an x satisfying A.16. Let us define $\bar{x} := x/e^w$. Therefore, by convexity of h_i's and the assumption $h_i(0) = 0$, we have that for every $1 \leqslant i \leqslant m$,

$$h_i(\bar{x}) = h_i(x/e^w) \leqslant h_i(x)/e^w = b_i f_i(x)/e^w \leqslant b_i,$$

where the last inequality follows from $f_i(x) \leqslant e^w$ for every i. Therefore, $\bar{x}$ satisfies A.14. On the other hand, we have that

$$c^T x/\mu \geqslant \min_j g_j(x) \geqslant e^{-w} \cdot \max_i f_i(x) \geqslant e^{-w} \cdot 1 = e^{-w}.$$

So we get

$$c^T \bar{x} = \frac{c^T x}{e^w} \geqslant e^{-2w}\mu.$$

Case 2. Algorithm 8 outputs there is no x satisfying the mixed problem. In this case we show that there is no x that satisfies constraints A.14 and A.15. Towards a contradiction, assume there exists an x such that $h_i(x) \leqslant b_i$ for every i, $i \in [m]$, and $c^T x \leqslant \mu$, therefore we have that

$$h_i(x)/b_i \leqslant 1 \leqslant \frac{c^T x}{\mu},$$

which is a contradiction.

The runtime analysis exactly follows from Theorem A.4.6 except from an additional factor τ coming for binary search.

□