From Duels to Battlefields: Computing Equilibria of Blotto and Other Games

AmirMahdi Ahmadinejad, Sina Dehghani, MohammadTaghi Hajiaghayi, Brendan Lucier, Hamid Mahini, Saeed Seddighin

informs
http://pubsonline.informs.org/journal/moor

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 44, No. 4, November 2019, pp. 1304–1325
ISSN 0364-765X (print), ISSN 1526-5471 (online)

From Duels to Battlefields: Computing Equilibria of Blotto and Other Games

AmirMahdi Ahmadinejad,[a] Sina Dehghani,[b] MohammadTaghi Hajiaghayi,[b] Brendan Lucier,[c] Hamid Mahini,[d] Saeed Seddighin[b]

[a] Department of Management Science and Engineering, Stanford University, Stanford, California 94305; [b] Department of Computer Science, University of Maryland, College Park, Maryland 20742; [c] Microsoft Research, Cambridge, Massachusetts 02142; [d] School of Electrical and Computer Engineering, University of Tehran, Tehran, Iran

Contact: ahmadi@stanford.edu (AA); sina.dehghani@gmail.com, http://orcid.org/0000-0002-4625-3199 (SD); hajiagha@cs.umd.edu (MH); brlucier@microsoft.com (BL); hamid.mahini@ut.ac.ir (HM); sseddigh@cs.umd.edu (SS)

Received: January 24, 2017
Revised: November 23, 2017; June 29, 2018
Accepted: July 15, 2018
Published Online in Articles in Advance: May 31, 2019

MSC2000 Subject Classification: Primary: 91A05; secondary: 68R05
OR/MS Subject Classifications: Game theory

https://doi.org/10.1287/moor.2018.0971

Copyright: © 2019 INFORMS

Abstract. In the well-studied Colonel Blotto game, players must divide a pool of troops among a set of battlefields with the goal of winning a majority. Despite the importance of this game, only a few solutions for special variants of the problem are known. We provide a general technique for computing equilibria of the Colonel Blotto game. Our approach applies to variations of the Colonel Blotto game as well, including an infinite-strategy variant called the General Lotto game. We also apply our technique beyond Colonel Blotto games to create the first polynomial-time algorithms for computing equilibria for a variety of other zero-sum games. Our approach is to reformulate each zero-sum game into a bilinear form, then reduce equilibrium computation to linear optimization over a game-specific polytope.

Funding: This work was supported in part by the National Science Foundation (NSF) [CAREER Award CCF-1053605, BIGDATA Grant IIS-1546108, and AF:Medium Grant CCF-1161365], the Defense Advanced Research Projects Agency (DARPA) [GRAPHS/AFOSR Grant FA9550-12-1-0423], and another DARPA SIMPLEX grant.

Keywords: algorithmic game theory • Nash equilibrium • Colonel Blotto • zero-sum games

1. Introduction

Computing a Nash equilibrium (NE) of a given game is a central problem in algorithmic game theory. It is known that every finite game admits a Nash equilibrium; that is, a profile of strategies from which no player can benefit from a unilateral deviation [38]. However, it is not necessarily obvious how to find an equilibrium. Indeed, impossibility results abound: it is widely believed that the class of polynomial parity arguments on directed graphs (PPAD) is computationally hard, and computing a Nash equilibrium of a normal-form game is known to be PPAD-complete [15, 20], even for two-player games [11]. In fact, it is PPAD-complete to find an $1/n^{O(1)}$ approximation to a Nash equilibrium [12]. These results call into question the predictiveness of a Nash equilibrium as a solution concept.

This motivates the study of classes of games for which equilibria can be computed efficiently. It has been found that many natural and important classes of games have structures that can be exploited to admit computational results [14, 19, 29, 35]. Perhaps the most well-known example is the class of zero-sum two-player games,[1] in which player 2's payoff is the negation of player 1's payoff. This natural class of games models perfect competition between two parties. The normal-form representation of a zero-sum game is a matrix A, which specifies the game payoffs for player 1. Given the payoff matrix for a zero-sum game as input, a Nash equilibrium[2] can be computed in polynomial time, and hence time polynomial in the number of pure strategies available to each player [14]. Yet even for zero-sum games, this algorithmic result is often unsatisfactory. The issue is that for many games the most natural representation is more succinct than simply listing a payoff matrix, so that the number of strategies is actually exponential in the most natural input size. In this case, the algorithm described above fails to guarantee efficient computation of equilibria, and alternative approaches are required.

1.1. The Colonel Blotto Game

A classic and important example illustrating these issues is the Colonel Blotto game, first introduced by Borel in 1921 [7, 8, 17, 18, 47]. In the Colonel Blotto game, two colonels each have a pool of troops and must fight against each other over a set of battlefields. The colonels simultaneously divide their troops between the battlefields. A colonel wins a battlefield if the number of his troops dominates the number of troops of his opponent. The final payoff of each colonel is the (weighted) number of battlefields won. An equilibrium of the game is a pair of colonels' strategies, which is a (potentially randomized) distribution of troops across battlefields, such that no colonel has incentive to change his strategy. Although the Colonel Blotto game was initially proposed to study a war situation, it has found applications in the analysis of many different forms of competition, from sports, to advertisement, to politics [37, 33, 36, 13, 30, 31], and has thus become one of the most well-known games in classic game theory.

Colonel Blotto is a zero-sum game. However, the number of strategies in the Colonel Blotto game is exponential in its natural representation. After all, there are $\binom{n+k-1}{k-1}$ ways to partition n troops among k battlefields. The classic methods for computing the equilibria of a zero-sum game therefore do not yield computationally efficient results. Moreover, significant effort has been made in the economics literature to understand the structure of equilibria of the Colonel Blotto game (i.e., by solving for equilibrium explicitly [46, 5, 6, 4, 44, 48, 41, 32, 24, 21, 31]). Despite this effort, progress remains sparse. Much of the existing work considers a continuous variation of the problem whereby troops are divisible, and for this variation, a significant breakthrough came only quite recently in the seminal work of Roberson [41], 85 years after the introduction of the game. Roberson finds an equilibrium solution for the continuous version of the game, in the special case that all battlefields have the same weight. The more general weighted version of the problem remains open, as does the original version with discrete strategies. Given the apparent difficulty of solving for equilibrium explicitly, it is natural to revisit the equilibrium computation problem for Colonel Blotto games.

1.2. An Approach: Bilinear Games

How should one approach equilibrium computation in such a game? A first observation is that, in zero-sum games, computing an equilibrium is equivalent to computing a MinMax strategy for each player. Therefore, throughout this paper, we focus on computing a MinMax strategy and use the terms MinMax strategy and Nash equilibrium interchangeably. The running time of the classic algorithm for finding a MinMax strategy depends polynomially on the size of the strategy set. However, the exponential size of the strategy set is not an impassable barrier; in certain cases, games with exponentially many strategies have an underlying structure that can be used to approach the equilibrium computation problem. For example, Koller et al. [28] show how to compute equilibria for zero-sum extensive-form games with perfect recall. Immorlica et al. [25] give an approach for solving algorithmically motivated “dueling games” with uncertainty. Letchford and Conitzer [34] compute equilibria for a variety of graphical security games. Each of these cases involves games with exponentially many strategies. In each case, a similar approach is used: reformulating the original game as a payoff-equivalent *bilinear* game. In a bilinear game, the space of strategies forms a polytope in R^n, and payoffs are specified by a matrix M: if the players play strategies x and y respectively, then the payoff to player 1 is $x^T My$. It has been observed that such bilinear games can be solved efficiently when the strategy polytope has polynomially many constraints [10, 28]. In each of the examples described above, it is shown how to map strategies from the original games to appropriate payoff-equivalent bilinear games, in which strategies are choices of marginal probabilities from the original game. In other words, the game payoffs can be formulated to depend only on the marginals of the chosen strategies (e.g., the distribution of troops assigned to each battlefield, in the Colonel Blotto game), rather than the entire mixed strategy profile itself. If one can also map a strategy in the bilinear game back to the original game, then one has a polynomial-time reduction to the (solved) problem of finding equilibria of the bilinear game. In each of these prior works, it is this latter step—mapping back to the original game—that is the most demanding; this generally requires a problem-specific way to convert a profile of marginals into a corresponding mixed strategy in the original game.

1.3. Our Contribution

We first show how to compute equilibria of the Colonel Blotto game. Like the works described above, our method is to consider a payoff-equivalent bilinear game defined over a space of appropriately selected marginals (in this case, the distribution of soldiers to a given battlefield). However, unlike those works, we do not explicitly construct a game-specific mapping to and from a polynomially sized bilinear game. We instead use a more general reduction, based on the idea that it suffices to solve linear optimization queries over strategy profiles in a (potentially exponentially sized) bilinear game. In other words, equilibrium computation reduces to the problem of finding a strategy that optimizes a given linear function over its marginal components. We apply our reduction to the Colonel Blotto game by showing how to solve these requisite optimization queries, which can be done via dynamic programming.

Our reduction follows from a repeated application of the classic equivalence of separation and optimization [22]. In more detail, we formulate the equilibrium conditions as a linear program (LP) whose feasibility region is the intersection of two polytopes: the first corresponding to the set of strategies of player 1, and the second encoding payoff constraints for player 2. To find a solution of the LP using the well-known ellipsoid method, it suffices to design a separation oracle for each polytope. However, as we show, separation oracles for the second polytope reduce to (and from) separation oracles for the set of strategies of player 2. It therefore suffices to design separation oracles for the polytope of strategies for each player, and for this it is enough to perform linear optimization over those polytopes [22]. Finally, to convert back to an equilibrium of the original game, we make use of a result from combinatorial optimization: the solution of an LP with polynomially many variables can always be expressed as a mixed strategy with a polynomial-size support, and such a mixed strategy can be computed using the separation oracles described previously [22].

The reduction described above is not specific to the Colonel Blotto game. It applies to any zero-sum game that admits a payoff-equivalent bilinear form. To the best of our knowledge, this general reduction from equilibrium computation to linear optimization has not previously been stated explicitly, although it has been alluded to in the security games literature,[3] and similar ideas have been used to compute correlated equilibria in compact games [26]. We demonstrate the generality of this approach by considering notable examples of games to which it can be applied. In each case, our approach either results in the first known polynomial-time algorithm for computing equilibria or else significantly simplifies prior analysis. Finally, we note that our approach also extends to approximations: given the ability to approximately answer separation oracle queries to within any fixed error $\epsilon > 0$, one can compute a corresponding approximation to the equilibrium payoffs.

1.3.1. Dueling Games. In a *dueling game*, introduced by Immorlica et al. [25], two competitors each try to design an algorithm for an optimization problem with an element of uncertainty, and each player's payoff is the probability of obtaining a better solution than their opponent. This framework falls within a natural class of ranking or social context games [1, 9], in which players separately play a base game and then receive ultimate payoffs determined by both their own outcomes and the outcomes of others. Immorlica et al. argue that this class of games models a variety of scenarios of competitions between algorithm designers: for example, competition between search engines (which must choose how to rank search results), or competition between hiring managers (who must choose from a pool of candidates in the style of the secretary problem).

Immorlica et al. [25] show how to compute a Nash equilibrium for certain dueling games, by developing mappings to and from bilinear games with compact representations. We extend their method and show how to expand the class of dueling games for which equilibria can be efficiently computed. As one particular example, we introduce and solve the *matching duel*. In this game, two players each select a matching in a weighted graph, and each player's payoff is the probability that a randomly selected node would have a higher-weight match in that player's matching than in the opponent's. Notably, because the matching polytope does not have a compact representation [42], the original method of Immorlica et al. [25] is not sufficient to find equilibria of this game. We also illustrate that our approach admits a significantly simplified analysis for some other dueling games previously analyzed by Immorlica et al.

1.3.2. General Lotto Game. Hart [24] considers a variant of the Colonel Blotto game, namely the *General Lotto* game. In this game, each player chooses a distribution over nonnegative real numbers, subject to the constraint that its expectation must equal a certain fixed value. A value is then drawn from each player's chosen distribution, and the players' payoffs are functions of these values. What is interesting about this game is that there are infinitely many pure strategies, which complicates equilibrium computation. Nevertheless, we show that our techniques can be applied to this class of games as well, yielding a polynomial-time algorithm for computing Nash equilibria. It is worth mentioning that the General Lotto game is an important problem by itself, and its continuous variant has been thoroughly studied in the literature (see, e.g., [3], [43], [24], and [16]).

2. Results and Techniques

We present a general method for computing Nash equilibria of a broad class of zero-sum games. Our approach is to reduce the problem of computing equilibria of a given game to the problem of optimizing linear functions over the space of strategies in a payoff-equivalent bilinear game.

Before presenting our general reduction, we will first illustrate our techniques in a high-level perspective by considering the Colonel Blotto game as a specific example in Section 2.1 (deferring the complete details to Section 3). Then in Section 2.2 we will present the general reduction. Further applications of this technique are provided in Section 4 (for dueling games) and Section 6 (for the General Lotto game).

2.1. Colonel Blotto

Here, we propose a polynomial-time algorithm for finding an equilibrium of discrete Colonel Blotto in its general form. We allow the game to be *asymmetric* across both the battlefields and the players. A game is *asymmetric across the battlefields* when different battlefields have different contributions to the outcome of the game, and a game is *asymmetric across the players* when the two players have different numbers of troops.

In the Colonel Blotto game, two players A and B simultaneously distribute a and b troops, respectively, over k battlefields. A pure strategy of player A is a k-partition $x = \langle x_1, x_2, \ldots, x_k \rangle$, where $\sum_{i=1}^{k} x_i = a$, and a pure strategy of player B is a k-partition $y = \langle y_1, y_2, \ldots, y_k \rangle$ where $\sum_{i=1}^{k} y_i = b$. Let $u_i^A(x_i, y_i)$ and $u_i^B(x_i, y_i)$ be the payoff of player A and player B from the ith battlefield, respectively, given the numbers of troops deployed to that battlefield. Note that the payoff functions of the ith battlefield, u_i^A and u_i^B, have $(a+1) \times (b+1)$ entries. This means the size of input is $\Theta(kab)$. Because Colonel Blotto is a zero-sum game, we have $u_i^A(x_i, y_i) = -u_i^B(x_i, y_i)$.[4] Note that we otherwise allow payoff functions to be arbitrary. So, for example, the payoffs might depend not only on which player wins a given battle, but by how much. We also represent the total payoff of player A and player B by $h_{\mathcal{B}}^A(x, y) = \sum_i u_i^A(x_i, y_i)$ and $h_{\mathcal{B}}^B(x, y) = \sum_i u_i^B(x_i, y_i)$, respectively.

A mixed strategy of each player would be a probability distribution over his pure strategies. For ease of reading, we will tend to distinguish pure strategies and mixed strategies by using boldface letters for mixed strategies. For example, because x and y denote pure strategies in the Colonel Blotto game, we write $\mathbf{x}$ and $\mathbf{y}$ to denote mixed strategies. We can think of $\mathbf{x}$ and $\mathbf{y}$ as random variables corresponding to the actions chosen under these mixed strategies.

Theorem 1. *One can compute an equilibrium of any Colonel Blotto game in polynomial time.*

Proof. Let $\mathcal{X}$ and $\mathcal{Y}$ be the set of all pure strategies of players A and B, respectively. That is, each member of $\mathcal{X}$ is a k-partition of a troops, and each member of $\mathcal{Y}$ is a k-partition of b troops. We will also write $\mathcal{M}(\mathcal{X})$ and $\mathcal{M}(\mathcal{Y})$ for the set of mixed strategies of players A and B, respectively.

We first write an exponentially sized LP that encodes the equilibria of the Colonel Blotto game. To that end, we can imagine representing a mixed strategy $\mathbf{x} \in \mathcal{M}(\mathcal{X})$ of player A with a vector $p \in [0,1]^{|\mathcal{X}|}$ such that $\sum_{x \in \mathcal{X}} p_x = 1$, where p_x denotes the probability that player A selects action $x \in \mathcal{X}$ under mixed strategy $\mathbf{x}$. Similarly, let vector $q \in [0,1]^{|\mathcal{Y}|}$ represent a mixed strategy of player B. Because Colonel Blotto is a zero-sum game, we leverage the MinMax theorem for finding an NE of the game. This theorem says that pair (p^*, q^*) is an NE of the Colonel Blotto game if and only if strategies p^* and q^* maximize the guaranteed payoff of players A and B, respectively [45]. Now, we are going to find strategy p^* of player A, which maximizes his guaranteed payoff. The same technique can be used for finding q^*. It is known that for each mixed strategy p, at least one of the best-response strategies to p is a pure strategy. Therefore, a solution to the following program characterizes strategy p^*:

$$\begin{aligned} \max \quad & U \\ \text{s.t.} \quad & \sum_{x \in \mathcal{X}} p_x = 1, \\ & \sum_{x \in \mathcal{X}} p_x h_{\mathcal{B}}^A(x, y) \geq U, \quad \forall y \in \mathcal{Y}. \end{aligned} \tag{1}$$

Unfortunately, LP 1 has $|\mathcal{X}|$ variables and $|\mathcal{Y}| + 1$ constraints where $|\mathcal{X}|$ and $|\mathcal{Y}| + 1$ are exponential. We therefore cannot solve LP 1 directly. We will instead solve LP 1 in three steps, described below.

Step 1 (Transferring to a New Space). We first transform the solution space to a new space in which an LP equivalent to LP 1 becomes tractable. See, for example, [2] for a similar approach. This new space will project mixed strategies onto the marginal probabilities for each (battlefield, troop count) pair. For each pure strategy $x \in \mathcal{X}$ of player A, we map it to a point in $\{0,1\}^{n(A)}$, where $n(A) = k \times (a+1)$. For convenience, we may abuse the notation and index each point $\hat{x} \in \{0,1\}^{n(A)}$ by two indices i and j such that $\hat{x}_{i,j}$ represents $\hat{x}_{(i-1)(a+1)+j+1}$. Now we map a pure strategy x to $\mathcal{G}_A(x) = \hat{x} \in \{0,1\}^{n(A)}$ such that $\hat{x}_{i,j} = 1$ if and only if $x_i = j$. In other words, if player A puts j troops in the ith battlefield, then $\hat{x}_{i,j} = 1$. Let $I_A := \{\hat{x} \in \{0,1\}^{n(A)} | \exists x \in \mathcal{X}, \mathcal{G}_A(x) = \hat{x}\}$ be the set of points in $\{0,1\}^{n(A)}$ that represent pure strategies of player A. Similarly, we map mixed strategy $\mathbf{x}$ to point $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}} \in [0,1]^{n(A)}$ such that $\hat{\mathbf{x}}_{i,j}$ represents the probability that mixed strategy $\mathbf{x}$ puts j troops in the ith battlefield. Note that we represent $\hat{\mathbf{x}}$ in boldface to indicate that it corresponds to the mixed strategy $\mathbf{x}$. Note also that the mapping $\mathcal{G}_A$ is not necessarily one-to-one nor onto; that is, each point in $[0,1]^{n(A)}$ may be mapped to zero strategies, one strategy, or more than one strategy.

Let $S_A = \{\hat{\mathbf{x}} \in [0,1]^{n(A)} | \exists \mathbf{x} \in \mathcal{M}(\mathcal{X}), \mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}\}$ be the set of points in $[0,1]^{n(A)}$ that represent at least one mixed strategy of player A. Similarly, we use function $\mathcal{G}_B$ to map each strategy of player B to a point in $[0,1]^{n(B)}$, where $n(B) = k \times (b+1)$, and define $I_B = \{\hat{y} \in \{0,1\}^{n(B)} | \exists y \in \mathcal{Y}, \mathcal{G}_B(y) = \hat{y}\}$ and $S_B = \{\hat{\mathbf{y}} \in [0,1]^{n(B)} | \exists \mathbf{y} \in \mathcal{M}(\mathcal{Y}), \mathcal{G}_B(\mathbf{y}) = \hat{\mathbf{y}}\}$. We show in Section 3 that set S_A forms a convex polygon (Lemma 1).

Now, we are ready to rewrite linear program 1 in the new space as follows:

$$\begin{aligned} \max \quad & U && (2)\\ \text{s.t.} \quad & \hat{\mathbf{x}} \in S_A && \text{(Membership constraint)}\\ & h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{y}) \geq U, \quad \forall \hat{y} \in I_B && \text{(Payoff constraints)}, \end{aligned}$$

where

$$h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{y}) = \sum_{i=1}^{k} \sum_{t_a=0}^{a} \sum_{t_b=0}^{b} \hat{\mathbf{x}}_{i,t_a} \hat{y}_{i,t_b} u^A_i(t_a, t_b)$$

is the expected payoff of player A.

Step 2 (Solving LP 2). The modified linear program, LP 2, has exponentially many constraints but only polynomially many variables. One can therefore apply the ellipsoid method to solve the LP, given a separation oracle that runs in polynomial time [23, 40]. By the equivalence of separation and optimization [22], one can implement such a separation oracle given the ability to optimize linear functions over the polytopes S_A (for the membership constraints) and S_B (for the payoff constraints).

Stated more explicitly, optimizing over the membership polytope S_A refers to the following optimization problem. Given a sequence of real numbers $c_0, c_1, \ldots, c_{k(m+1)}$, where k is the number of battlefields and m is the number of troops for a player, the required oracle must find a pure strategy $x = (x_1, x_2, \ldots, x_k) \in \mathcal{X}$ such that $\sum_{i=1}^k x_i = m$, and $\hat{x} = \mathcal{G}(x)$ minimizes the following expression:

$$c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i, \tag{3}$$

and similarly for polytope $\mathcal{Y}$. We show that one can indeed find a minimizer of Equation (3) in polynomial time, using dynamic programming. It is similarly possible to optimize linear constraints over S_B, the polytope of payoff constraints, using an oracle that can be implemented in polynomial time. We provide more details in Section 3.2 and formally describe the optimization algorithms in Section 7.

Step 3 (Transferring to the Original Space). At last we should transfer the solution of LP 2 to the original space. In particular, we are given a point $\hat{\mathbf{x}} \in S_A$ and our goal is to find a strategy $\mathbf{x} \in \mathcal{M}(\mathcal{X})$ such that $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$. To achieve this, we invoke a classic result of [22], which states that an interior point of an n-dimensional polytope P can be decomposed as a convex combination of at most $n+1$ extreme points of P, in polynomial time, given an oracle that optimizes linear functions over P. Note that this is precisely the oracle required for Step 2 above. Applying this result to the solution of LP 2 in polytope S_A, we obtain a convex decomposition of $\hat{\mathbf{x}}$ into extreme points of S_A, say $\hat{\mathbf{x}} = \sum_i \alpha_i \hat{x}^i$. Because each $\hat{x}_i$ corresponds to a pure strategy in $\mathcal{X}$, it is trivial to find strategy x^i with $\mathcal{G}_A(x^i) = \hat{x}^i$, because the marginals of each $\hat{x}^i$ lie in $\{0,1\}$. We then have that $\mathbf{x} = \sum_i \alpha_i x^i$ is the required mixed strategy profile.

Combining these three steps, we find a Nash equilibrium of the Colonel Blotto game in polynomial time, completing the proof of Theorem 1. See Section 3 for more details. □

2.2. A General Framework for Bilinear Games

In our method for finding a Nash equilibrium of the Colonel Blotto game, the main steps were to express the game as a bilinear game of polynomial dimension, solve for an equilibrium of the bilinear game, then express that point as an equilibrium of the original game. To implement the final two steps, it sufficed to show how to optimize linear functions over the polytope of strategies in the bilinear game. This suggests a general reduction, whereby the equilibrium computation problem is reduced to finding the appropriate bilinear game and implementing the required optimization algorithm. In other words, the method for computing Nash equilibria applies to a zero-sum game when:

1. One can transfer each strategy $\mathbf{x}$ of player A to $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}} \in R^{n(A)}$, and each strategy $\mathbf{y}$ of player B to $\mathcal{G}_B(\mathbf{y}) = \hat{\mathbf{y}} \in R^{n(B)}$ such that the payoff of the game for strategies $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ can be represented in a bilinear form based on $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$; that is, the payoff is $\hat{\mathbf{x}}^t M \hat{\mathbf{y}}$, where M is a $n(A) \times n(B)$ matrix.

2. For any given vector α and real number α_0 we can find, in polynomial time, whether there is a pure strategy $\hat{x}$ in the transferred space such that $\alpha_0 + \sum_i \alpha_i \hat{x}_i \geq 0$.

We refer to such a game as *polynomially separable*. A direct extension of the proof of Theorem 1 implies that Nash equilibria can be found for polynomially separable games.

Theorem 2. *There is a polynomial-time algorithm that finds a Nash Equilibrium of a given polynomially separable game.*

This general methodology can be used for finding an NE in many zero-sum games. In subsequent sections, we show how our framework can be used to find Nash equilibria for a generalization of Blotto games and for a class of dueling games introduced by Immorlica et al. [25].

2.3. General Lotto

The General Lotto game is a relaxation of the Colonel Blotto game (see [24] for details). In this game each player's strategy is a distribution of a nonnegative integer-valued random variable with a given expectation. In particular, players A and B simultaneously determine (two distributions of) two nonnegative integer-valued random variables X and Y, respectively, such that $\mathbb{E}[X] = a$ and $\mathbb{E}[Y] = b$. The payoff of player A is

$$h^A_\Gamma(X, Y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \Pr(X = i) \Pr(Y = j) u(i, j), \tag{4}$$

and again the payoff of player B is the negative of the payoff of player A, that is, $h^B_\Gamma(X, Y) = -h^A_\Gamma(X, Y)$. Hart [24] presents a solution for the General Lotto game when $u(i, j) = \operatorname{sign}(i - j)$. Here, we generalize this result and present a polynomial-time algorithm for finding an equilibrium when u is a *bounded distance function*. Function u is a bounded distance function if one can write it as $u(i, j) = f_u(i - j)$ such that f_u is a monotone function and reaches its maximum value at $u^M = f_u(u^T)$, where $u^T \in O(\operatorname{poly}(a, b))$. Note that $u(i, j) = \operatorname{sign}(i - j)$ is a bounded distance function where it reaches its maximum value at $i - j = 1$. The following theorem presents our main result regarding the General Lotto game.

Theorem 3. *There is a polynomial-time algorithm that finds an equilibrium of a General Lotto game where the payoff function is a bounded distance function.*

2.3.1. Main Challenge.
In the General Lotto game, each player has an infinite number of pure strategies, and thus one can use neither our proposed algorithm for the Colonel Blotto game nor the technique of Immorlica et al. [25] for solving the problem. To address this issue, we should prune strategies such that the problem becomes tractable. To that end, our approach is to characterize the extreme point of the polytope of all strategies and use this characterization for pruning possible strategies. To the best of our knowledge, our algorithm is the first to compute an NE of a game with an infinite number of pure strategies in this way. The full proof of Theorem 3 appears in Section 6.

3. Colonel Blotto

In this section, we provide a more detailed description of our polynomial-time algorithm for finding a Nash equilibrium of the Colonel Blotto game, which we described informally in Section 2.1. In Section 3.1, we present a procedure for mapping strategies of both players to a new space. The new space maintains the important information of each strategy and helps us to find a Nash equilibrium of the game. Next, in Section 3.2, we demonstrate how to verify the feasibility of the membership and payoff constraints in the new space. This allows us to construct a polynomial-time algorithm for determining an equilibrium of the Colonel Blotto game in the new space. At last, in Section 3.3, we present an algorithm that transfers a Nash equilibrium from the new space to the original space.

3.1. Transferring to a New Space

In this subsection we define a new notation for describing the strategies of players and discuss the properties of those strategies. Let $n(A) = k(a + 1)$ and $\mathbf{x}$ be a strategy of player A. We define the function $\mathcal{G}_A$ in the following way: $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$, where $\hat{\mathbf{x}}$ is a point in $\mathbb{R}^{n(A)}$ such that $\hat{\mathbf{x}}_{(i-1)(a+1)+j+1}$ is equal to the probability that strategy $\mathbf{x}$ puts j units in the ith battlefield, for $1 \leq i \leq k$ and $0 \leq j \leq a$. For simplicity, we may represent $\hat{\mathbf{x}}_{(i-1)(a+1)+j+1}$ by $\hat{\mathbf{x}}_{i,j}$. We define $n(B)$ and $\mathcal{G}_B$ similarly for player B. Let $n = \max\{n(A), n(B)\}$. Note that $\mathcal{G}_A$ maps each strategy of the first player to exactly one point in $\mathbb{R}^{n(A)}$. However, each point in $\mathbb{R}^{n(A)}$ may be mapped to either zero strategies, one strategy, or more than one strategy. Figure 1 illustrates an example of a mixed strategy along with its mapping into the new space.

Figure 1. A strategy of a player with three troops in a Colonel Blotto game with three battlefields.

0.3 x

Strategy	0 troops	1 troop	2 troops	3 troops
battlefield 1	0	0	0	1
battlefield 2	1	0	0	0
battlefield 3	1	0	0	0

+

0.4 x

Strategy	0 troops	1 troop	2 troops	3 troops
battlefield 1	0	1	0	0
battlefield 2	0	1	0	0
battlefield 3	0	1	0	0

+

0.3 x

Strategy	0 troops	1 troop	2 troops	3 troops
battlefield 1	1	0	0	0
battlefield 2	0	0	1	0
battlefield 3	0	1	0	0

=

Strategy	0 troops	1 troop	2 troops	3 troops
battlefield 1	0.3	0.4	0	0.3
battlefield 2	0.3	0.4	0.3	0
battlefield 3	0.3	0.7	0	0

Notes. On the left is a mixed strategy x, which plays as follows: with probability 0.3 puts all troops in the first battlefield, with probability 0.4 puts one troop in each battlefield, and with probability 0.3 puts two and one troop(s) in the second and third battlefields, respectively. The table on the right shows how $\hat{x}$ is derived from x.

Let us recall the definition of $\mathcal{M}(\mathcal{X})$, which is the set of all strategies of player A, and the definition of S_A, which is

$$S_A = \{\hat{\mathbf{x}} \in [0,1]^{n(A)} \mid \exists \mathbf{x} \in \mathcal{M}(\mathcal{X}), \mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}\}.$$

To design an algorithm for verifying the membership constraints, we first show in Lemma 1 that set S_A is a polyhedron with an exponential number of vertices and facets. Then we prove in Lemma 4 that set S_A can be formulated with $\mathbb{O}(2^{\text{poly}(n)})$ number of constraints. These results enable us to leverage the ellipsoid method for verifying the membership constraints [23].

Lemma 1. *Set S_A forms a convex polyhedron with no more than $n(A)^{n(A)}$ vertices and no more than $n(A)^{(n(A)^2)}$ facets.*

Proof. First we need Lemmas 2 and 3 for proving Lemma 1.

Lemma 2. *Let $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^t$ be t arbitrary mixed strategies of player A, $\sum_{r=1}^{t} \alpha_r = 1$, and $\mathbf{x} = \sum_{r=1}^{t} \alpha_r \mathbf{x}^r$ be a mixed strategy that plays strategy $\mathbf{x}^r$ with probability α_r; then $\mathcal{G}_A(\mathbf{x}) = \sum_{r=1}^{t} \alpha_r \mathcal{G}_A(\mathbf{x}^r)$.*

Proof. Because $\mathbf{x} = \sum_{r=1}^{t} \alpha_r \mathbf{x}^r$ and $\mathcal{G}_A(\mathbf{x}^r)_j$ represent the probability that strategy $\mathbf{x}^r$ plays j, we have $[\mathcal{G}_A(\mathbf{x})]_j = \sum_{r=1}^{t} \alpha_r [\mathcal{G}_A(\mathbf{x}^r)]_j$ for all $1 \le j \le n(A)$. Therefore, $\mathcal{G}_A(\mathbf{x}) = \sum_{r=1}^{t} \alpha_r \mathcal{G}_A(\mathbf{x}^r)$. □

Lemma 3. *S_A is a convex set.*

Proof. A set of points is convex if and only if every segment joining two of its points is completely in the set. Let $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_{n(A)})$ and $\hat{\mathbf{x}}' = (\hat{\mathbf{x}}'_1, \hat{\mathbf{x}}'_2, \ldots, \hat{\mathbf{x}}'_{n(A)})$ be two points in $\mathbb{R}^{n(A)}$. We show that if $\hat{\mathbf{x}}, \hat{\mathbf{x}}' \in S_A$, then for every $0 \le \alpha \le 1$, $(\alpha\hat{\mathbf{x}} + (1-\alpha)\hat{\mathbf{x}}') \in S_A$.

Because $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ are in S_A, there exist mixed strategies $\mathbf{x}$ and $\mathbf{x}'$ in $\mathcal{M}(\mathcal{X})$, such that $\hat{\mathbf{x}} = \mathcal{G}_A(\mathbf{x})$ and $\hat{\mathbf{x}}' = \mathcal{G}_A(\mathbf{x}')$. Let $\mathbf{x}'' = \alpha\mathbf{x} + (\mathbf{1} - \alpha)\mathbf{x}'$ be a mixed strategy of player A that plays $\mathbf{x}$ with probability α and $\mathbf{x}'$ with probability $1-\alpha$. By Lemma 2 we have $\mathcal{G}_A(\mathbf{x}'') = \alpha\hat{\mathbf{x}} + (1-\alpha)\hat{\mathbf{x}}' = \hat{\mathbf{x}}''$; hence, $\hat{\mathbf{x}}'' \in S_A$. □

Now we are ready to prove Lemma 1. By Lemma 3, we know S_A is convex. We show that we can find a finite set of points in S_A such that every point in S_A can be written as a convex combination of these points. Note that $\mathcal{X} = \{\mathcal{X}^1, \mathcal{X}^2, \ldots, \mathcal{X}^{|\mathcal{X}|}\}$ is the set of all pure strategies of player A, and $I_A = \{\hat{\mathcal{X}}^1, \hat{\mathcal{X}}^2, \ldots, \hat{\mathcal{X}}^{|\mathcal{X}|}\}$ is the set of points where $\hat{\mathcal{X}}^i = \mathcal{G}_A(\mathcal{X}^i)$. Note also that I_A and $\mathcal{X}$ are finite sets. Every strategy of player A is either a pure strategy or a mixed strategy and can be written as a convex combination of the pure strategies. Therefore, according to Lemma 2, every point in S_A is either in I_A or can be written as a convex combination of points in I_A. Hence, S_A forms a convex polyhedron in $\mathbb{R}^{n(A)}$.

Next we show that the number of vertices and facets of S_A is $\mathbb{O}(2^{\text{poly}(n)})$. Let $\hat{\mathbf{x}}$ be a vertex of polyhedron S_A and $\mathbf{x}$ be a strategy of player A such that $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$. Because $\hat{\mathbf{x}}$ is a vertex of S_A, it cannot be written as a convex combination of other vertices of S_A. If $\mathbf{x}$ is a mixed strategy, it can be written as a convex combination of other strategies, and according to Lemma 2, $\mathscr{G}_A(\mathbf{x})$ can be written as convex combination of other points of S_A. This implies that either $\mathbf{x}$ is a pure strategy or there exists a pure strategy $\mathbf{x}'$ such that $\mathscr{G}_A(\mathbf{x}') = \hat{\mathbf{x}}$. Therefore, the number of vertices of S_A is no more than the number of pure strategies of player A. Because each pure strategy of player A is a partition of a units into k battlefields, the number of pure strategies is no more than $(a+1)^k$. Thus, the number of vertices of S_A is at most $n(A)^{n(A)}$.

Let $d \le n(A)$ be the dimension of S_A. Because every facet of S_A can be uniquely determined by d vertices of S_A, the number of facets is no more than

$$\binom{|I_A|}{d} \le (|I_A|)^d \le (n(A)^{n(A)})^d \le n(A)^{(n(A)^2)}. \quad \square$$

Lemma 4. *Set S_A can be formulated with $\mathbb{O}(2^{\mathrm{poly}(n)})$ number of constraints.*

Proof. Note that the dimension of S_A is not necessarily $n(A)$. Let d be the dimension of S_A and $\mathscr{H}(S_A)$ be the affine hull of S_A. Because $\mathscr{H}(S_A)$ is a d-dimensional subspace of $\mathbb{R}^{n(A)}$, it can be represented as the intersection of $n(A) - d$ orthogonal hyperplanes. Let L be a set of such hyperplanes. Let C be the set of all hyperplanes perpendicular to hyperplanes in L that contain a facet of S_A. We need Lemmas 5, 6, 7, and 8 for proving Lemma 4.

Lemma 5. *There exists exactly one hyperplane perpendicular to all hyperplanes in L that contains all points of a $(d-1)$-dimensional subspace of $\mathscr{H}(S_A)$.*

Proof. Let O be the orthogonal basis of the $(d-1)$-dimensional subspace and N be the set of normal vectors of hyperplanes in L. Because every vector in O is in $\mathscr{H}(S_A)$, all vectors in O are orthogonal to all vectors in N. Therefore, the desired hyperplane should be orthogonal to all vectors $N \cup O$. Because $|N \cup O| = n(A) - 1$ and all vectors in $N \cup O$ are pairwise orthogonal, there exists exactly one hyperplane containing the subspace and perpendicular to all hyperplanes in L. $\square$

Lemma 6. *There exists exactly one hyperplane perpendicular to all hyperplanes of L that contain all points of some facet f of S_A.*

Proof. Because f is a facet of S_A, its dimension is $d-1$. Therefore, its affine hull is a $(d-1)$-dimensional subspace of $\mathscr{H}(S_A)$. Note that a hyperplane contains f if and only if it contains the affine hull of f. Lemma 5 states that there exists exactly one hyperplane perpendicular to all hyperplanes of L that contains the affine hull of f. Hence, there exists a unique hyperplane containing f that is perpendicular to all hyperplanes of L. $\square$

Lemma 7. *For every point $\hat{\mathbf{x}} \in \mathscr{H}(S_A)$ that is not in S_A, there exists a hyperplane perpendicular to all hyperplanes in L that contains a facet of S_A and separates $\hat{\mathbf{x}}$ from S_A.*

Proof. Consider a segment between $\hat{\mathbf{x}}$ and one point of S_A. Because one endpoint of the segment is in S_A and the other one is not, it has intersection with at least one facet of S_A, namely f. By Lemma 6, there exists one hyperplane that contains f and is perpendicular to hyperplanes in L. This hyperplane has intersection with the segment and separates $\hat{\mathbf{x}}$ from S_A. $\square$

Lemma 8. *Point $\hat{\mathbf{x}} \in \mathbb{R}^{n(A)}$ is in S_A if and only if all hyperplanes in L contain $\hat{\mathbf{x}}$ and no hyperplane in C separates $\hat{\mathbf{x}}$ from S_A.*

Proof. Because S_A is in the intersection of all hyperplanes in L, if $\hat{\mathbf{x}} \in S_A$ it is also in all hyperplanes in L, and no hyperplane in C separates $\hat{\mathbf{x}}$ from S_A. Now suppose $\hat{\mathbf{x}} \notin S_A$. If $\hat{\mathbf{x}} \notin \mathscr{H}(S_A)$, then $\hat{\mathbf{x}}$ is not in all hyperplanes of L; otherwise by Lemma 7 there exists a hyperplane in C that separates $\hat{\mathbf{x}}$ from S_A. $\square$

Now we are ready to prove Lemma 4. By Lemma 8 we conclude that the set of hyperplanes in L and C is sufficient to formulate S_A. We know $|L|$ is at most $n(A)$, and by Lemma 6 we can find out that $|C|$ is equal to the number of facets of S_A. Moreover, Lemma 1 states that $|C|$ is $\mathbb{O}(2^{\mathrm{poly}(n)})$, which means $|C| + |L| \in \mathbb{O}(2^{\mathrm{poly}(n)})$. $\square$

3.2. Verifying the Membership and the Payoff Constraints

As we briefly described in Section 2, the final goal of this section is to determine an NE of the Colonel Blotto game. To do this, we use linear program 2 and show that this LP can be solved in polynomial time. Because we use the ellipsoid method to solve the LP, we have to implement an oracle function that reports a violating constraint for any infeasible solution.

We will first focus on the membership constraints of LP 2. We show there exists a polynomial-time algorithm that finds a separating hyperplane for any point $\hat{\mathbf{x}}$ that violates a membership constraint.

Lemma 9. *There exists a polynomial-time algorithm that takes as input a point $\hat{\mathbf{x}}$ and either finds a hyperplane that separates $\hat{\mathbf{x}}$ from S_A or reports that no such hyperplane exists.*

Proof. Let $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_{n(A)})$. Consider the following LP, which we will refer to as LP 5:

$$\alpha_0 + \sum_{j=1}^{n(A)} \alpha_j \hat{\mathbf{x}}_j < 0 \tag{5}$$

$$\alpha_0 + \sum_{j=1}^{n(A)} \alpha_j \hat{v}_j \geq 0 \qquad \forall \hat{v} \in I_A. \tag{6}$$

The variables of this LP are $\alpha_0, \alpha_1, \dots, \alpha_{n(A)}$, which describe the following hyperplane:

$$\alpha_0 + \sum_{j=1}^{n(A)} \alpha_j \hat{x}'_j = 0.$$

Constraints (5) and (6) force LP 5 to find a hyperplane that separates $\hat{\mathbf{x}}$ from S_A. Hence, LP 5 finds a separating hyperplane if and only if $\hat{\mathbf{x}}$ is not in S_A.

A *hyperplane separating oracle* is an oracle that gets variables $\alpha_0, \alpha_1, \dots, \alpha_{n(A)}$ as input and finds whether constraints (6) are satisfied. Moreover, if some constraints are violated it returns at least one of the violated constraints. In Section 7 we describe a polynomial-time algorithm for the hyperplane separating oracle. Note that (5) is a single constraint and can be verified in polynomial time. LP 5 has $n(A) + 1$ variables and $|I_A| + 1$ constraints. By Lemma 1, $|I_A| = \mathbb{O}(2^{\text{poly}(n)})$. Thus we can use the ellipsoid method [23] to solve this LP in polynomial time. □

Having handled the membership constraints, we now turn to the payoff constraints of LP 2. We present an algorithm to determine the outcome of the game (i.e., the payoffs) when both players play optimally. We say $\mathbf{x}$ is an optimal strategy of player A if it maximizes the guaranteed payoff of player A. By the MinMax theorem, an NE of a zero-sum game is precisely a strategy profile in which each player plays an optimal strategy [45]. Therefore, it is enough to find an optimal strategy of both players. Before we discuss the algorithm, we show the payoff $h^A_{\mathcal{B}}(\mathbf{x}, \mathbf{y})$ can be determined by $\mathcal{G}_A(\mathbf{x})$ and $\mathcal{G}_B(\mathbf{y})$. Recall the definition of $h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, which is

$$h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = \sum_{i=1}^{k} \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \hat{\mathbf{x}}_{i,\alpha} \hat{\mathbf{y}}_{i,\beta} u^A_i(\alpha, \beta).$$

Lemma 10. *Let $\mathbf{x} \in \mathcal{M}(\mathcal{X})$ and $\mathbf{y} \in \mathcal{M}(\mathcal{Y})$ be two mixed strategies for players A and B, respectively. Let $\hat{\mathbf{x}} = \mathcal{G}_A(\mathbf{x})$ and $\hat{\mathbf{y}} = \mathcal{G}_B(\mathbf{y})$. The outcome of the game is determined by $h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.*

Proof. Let $\mathbf{x}$ and $\mathbf{y}$ be two mixed strategies of players A and B, respectively, and let $\mathbb{E}[u^A_i(\mathbf{x}, \mathbf{y})]$ be the expected value of the outcome in battlefield i. We can write $\mathbb{E}[u^A_i(\mathbf{x}, \mathbf{y})]$ as follows:

$$\mathbb{E}[u^A_i(\mathbf{x}, \mathbf{y})] = \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \hat{\mathbf{x}}_{i,\alpha} \hat{\mathbf{y}}_{i,\beta} u^A_i(\alpha, \beta).$$

We know that the total outcome of the game is the sum of the outcome in all battlefields, which is

$$\mathbb{E}\left[\sum_{i=1}^{k} u^A_i(\mathbf{x}, \mathbf{y})\right] = \sum_{i=1}^{k} \mathbb{E}[u^A_i(\mathbf{x}, \mathbf{y})] = \sum_{i=1}^{k} \sum_{\alpha=0}^{a} \sum_{\beta=0}^{b} \hat{\mathbf{x}}_{i,\alpha} \hat{\mathbf{y}}_{i,\beta} u^A_i(\alpha, \beta) = h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{\mathbf{y}}). \quad \square$$

We now have all the pieces we require to solve LP 2, which solves the equilibrium problem in the new space.

Lemma 11. *There exists a polynomial-time algorithm that finds an NE of the Colonel Blotto game in the new space.*

Proof. The Colonel Blotto is a zero-sum game, and the MinMax theorem states that a pair of strategies $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a Nash equilibrium if $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ maximize the guaranteed payoff of players A and B, respectively [45].

Recall that LP 2 finds a point $\hat{\mathbf{x}} \in S_A$ that describes an optimal strategy of player A.[5] This LP has $n(A)+1$ variables, which are $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{n(A)}$ and U. The membership constraints guarantee $\hat{\mathbf{x}}$ is in S_A. It is known that in any normal-form game there always exists a best-response strategy that is a pure strategy [39]. Hence, variable U represents the maximum payoff of player A with strategy $\hat{\mathbf{x}}$ when player B plays his best-response strategy against $\hat{\mathbf{x}}$. Note that Lemma 10 shows $h^A_{\mathcal{B}}(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a linear function of $\hat{\mathbf{x}}$, when $\hat{\mathbf{y}}$ is a fixed strategy of player B. This means the payoff constraints are linear constraints. Putting all these together implies that LP 2 finds a point $\hat{\mathbf{x}}$ subject to the following conditions:

1. There exists a strategy $\mathbf{x}$ such that $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$.
2. The minimum value of $h^A_{\mathcal{B}}(\mathbf{x}, \mathbf{y})$ is maximized for every $\mathbf{y} \in \mathcal{M}(\mathcal{Y})$.

Now we prove that this LP can be solved in polynomial time using the ellipsoid method. The *best-response separating oracle* is an oracle that gets a point $\hat{\mathbf{x}}$ and a variable U as input and either reports that point $\hat{\mathbf{x}}$ meets all payoff constraints or reports a violated payoff constraint. In Section 7, we will show that the running time of this oracle is $\mathbb{O}(\text{poly}(n))$. Note that Lemma 9 provides a polynomial-time algorithm for verifying the membership constraints, and that the best-response separating oracle is a polynomial-time algorithm for verifying the payoff constraints.

Now by Lemmas 1 and 4, LP 2 has $\mathbb{O}(2^{\text{poly}(n)})$ many constraints. Thus we can use ellipsoid method to solve LP 2 in polynomial time. □

3.3. Finding a Nash Equilibrium in the Original Space

In the previous subsection, we presented an algorithm that finds a Nash equilibrium $(\mathcal{G}_A(\mathbf{x}), \mathcal{G}_B(\mathbf{y}))$ of the game in the new space. The remaining problem is to translate $\mathcal{G}_A(\mathbf{x})$ back to the original space and compute $\mathbf{x}$.

Lemma 12. *Given a point $\hat{\mathbf{x}} \in S_A$, there exists a polynomial-time algorithm that finds a strategy $\mathbf{x} \in \mathcal{M}(\mathcal{X})$ such that $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$.*

Proof. Because every strategy of player A is a convex combination of elements of $\mathcal{X}$, we can find such an $\mathbf{x}$ by finding a feasible solution to the following LP. Because every feasible solution is acceptable, the objective function does not matter. In this LP, α_x's are decision variables: for each pure strategy $x \in \mathcal{X}$, α_x denotes the corresponding probability of x in the mixed strategy.

$$
\begin{aligned}
\min \quad & 0 && && (7)\\
\text{s.t.} \quad & \sum_{x\in\mathcal{X}} \alpha_x = 1 && && (8)\\
& \sum_{x\in\mathcal{X}} \alpha_x \mathcal{G}_A(x)_j = \hat{x}_j && \forall 1 \le j \le n(A) && (9)\\
& \alpha_x \ge 0 && \forall x \in \mathcal{X} && (10)
\end{aligned}
$$

To find a solution of the above LP, we write its dual LP as follows.

$$
\begin{aligned}
\max \quad & \beta_0 + \sum_{j=1}^{n(A)} \hat{x}_j \beta_j && && (11)\\
\text{s.t.} \quad & \beta_0 + \sum_{j=1}^{n(A)} \mathcal{G}_A(x)_j \beta_j \le 0 && \forall x \in \mathcal{X} && (12)
\end{aligned}
$$

In this LP, the decision variables are $\beta_0, \beta_1, \ldots, \beta_{n(A)}$. The variable β_0 corresponds to constraint (8), and variables $\beta_1, \beta_2, \ldots, \beta_{n(A)}$ correspond to constraint (9). An oracle similar to the hyperplane separating oracle (described in Section 3.2) can find a violating constraint for any infeasible solution of the dual LP. Because the number of constraints in the dual LP is $|I_A| = \mathbb{O}(2^{\text{poly}(n)})$ owing to Lemma 1, we use the ellipsoid method to find an optimal solution of the dual LP in polynomial time.

The next challenge is to find an optimal solution of the primal LP from an optimal solution of the dual LP. This can be done via classic LP solving techniques [27]. However, for the sake of completeness, we include a formal statement and proof as Lemma 13, below.

Lemma 13. *Assume we have a separation oracle for primal LP* $\max\{c^T x : Ax \le b\}$ *with exponentially many constraints and polynomially many variables. If primal LP is feasible, then there is a polynomial-time algorithm that returns an optimum solution of dual LP* $\min\{b^T y : A^T y \ge c\}$.

Proof. Because the primal LP is feasible, we can assume $\mathrm{OPT} = \max\{c^T x : Ax \le b\}$. The ellipsoid method returns an optimum solution of primal LP by doing binary search and finding the largest K that guarantees feasibility of $\{c^T x \le K : Ax \le b\}$. Let $(\hat{A}, \hat{b})$ be the set of polynomially many constraints returned by the separation oracle during all iterations. We first prove $\max\{c^T x : \hat{A}x \le \hat{b}\} = \mathrm{OPT}$. Note that $(\hat{A}, \hat{b})$ is a set of constraints returned by the ellipsoid method. Note also that $(\hat{A}, \hat{b})$ is a subset of all constraints (A, b). This means every vector x that satisfies $Ax \le b$ will satisfy $\hat{A}x \le \hat{b}$ as well. Therefore, $\max\{c^T x : \hat{A}x \le \hat{b}\} \ge \max\{c^T x : Ax \le b\} = \mathrm{OPT}$. On the other hand, we know $(\hat{A}, \hat{b})$ contains constraints that guarantee that the set $\{c^T x \ge \mathrm{OPT} + \epsilon : Ax \le b\}$ is empty. In particular, $\{c^T x \ge \mathrm{OPT} + \epsilon : \hat{A}x \le \hat{b}\}$ is empty, which means that $\max\{c^T x : \hat{A}x \le \hat{b}\} \le \mathrm{OPT}$. Putting all these together we can conclude that $\max\{c^T x : \hat{A}x \le \hat{b}\} = \mathrm{OPT}$.

The linear program $\max\{c^T x : \hat{A}x \le \hat{b}\}$ has polynomially many constraints and polynomially many variables, and we can find an optimum solution to its dual, $\min\{\hat{b}^T \hat{y} : \hat{A}^T \hat{y} \ge c\}$, in polynomial time. Let $\hat{y}^*$ be an optimum solution of dual LP $\min\{\hat{b}^T \hat{y} : \hat{A}^T \hat{y} \ge c\}$, and let $S = \{i | (A_i, b_i) \text{ is in } (\hat{A}, \hat{b})\}$, where A_i is the ith row of matrix A, be the set of indices corresponding to constraints in $(\hat{A}, \hat{b})$. For every vector y and every set of indices R we define y_R to be the projection of vector y on set R. Now let vector y^* be a solution of dual LP $\min\{b^T y : A^T y \ge c\}$ such that $y^*_S = \hat{y}^*$ and $y^*_i = 0$ for all $i \notin S$. We prove y^* is an optimum solution of dual LP $\min\{b^T y : A^T y \ge c\}$ as follows:

- We first show y^* is feasible. Note that $y^*_i = 0$ for all $i \notin S$, which means $A^T y^* = \hat{A}^T \hat{y}^* \ge c$, where the last inequality comes from the feasibility of $\hat{y}^*$ in dual LP $\min\{\hat{b}^T y : \hat{A}^T y \ge c\}$.
- Note that $b^T y^* = \sum_i b_i^T y_i^* = \sum_{i \in S} b_i^T y_i^* + \sum_{i \notin S} b_i^T y_i^* = \hat{b}^T \hat{y}^*$. The last equality comes from the facts that $y_i^* = 0$ for all $i \notin S$ and $\sum_{i \in S} b_i^T y_i^* = \hat{b}^T \hat{y}^*$. Because $\hat{y}^*$ is an optimum solution of the dual LP $\min\{\hat{b}^T y : \hat{A}^T y \ge c\}$, by the weak duality, it is equal to $\max\{c^T x : \hat{A}x \le \hat{b}\} = \mathrm{OPT}$. Therefore, $b^T y^* = \mathrm{OPT}$.

We have proved y^* is a feasible solution to dual LP $\min\{b^T y : A^T y \ge c\}$ and $b^T y^* = \mathrm{OPT}$. We also know $\mathrm{OPT} = \max\{c^T x : Ax \le b\}$ by definition. Therefore, the weak duality insures y^* is an optimum solution of dual LP $\min\{b^T y : A^T y \ge c\}$. □

We know that $\hat{\mathbf{x}}$ is in S_A. This means there is a strategy $\mathbf{x} \in \mathcal{M}(\mathcal{X})$ such that $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}}$. Therefore, LP 7 and its dual are both feasible, and thus we can apply Lemma 13 to find an optimal solution in polynomial time, completing the proof of Lemma 12. □

4. Application to Dueling Games

Immorlica et al. [25] introduced the class of dueling games. In these games, an optimization problem with an element of uncertainty is used to define a competition between two players. They also provide a technique for finding Nash equilibria for a set of games in this class. In this section, we formally define the dueling games and bilinear duels. Then, in Section 4.3, we describe our method and show that our technique solves a more general class of dueling games. Furthermore, we provide examples of bilinear duel games, considered previously [25], to demonstrate the simplicity of our method. Finally, in Section 4.4, we examine the matching duel game to provide an example in which the method of Immorlica et al. [25] does not apply directly but our presented method can still be applied.

4.1. Dueling Games

Formally, dueling games are two player zero-sum games with a set of strategies X, a set of possible situations Ω, a probability distribution p over Ω, and a cost function $c : X \times \Omega \to \mathbb{R}$ that defines the cost measure for each player on the basis of her strategy and the element of uncertainty. The payoff of each player is defined as the probability that she beats her opponent minus the probability that she is beaten. More precisely, the utility function is defined as

$$h^A(x, y) = -h^B(x, y) = \Pr_{\omega \sim p}[c(x, \omega) < c(y, \omega)] - \Pr_{\omega \sim p}[c(x, \omega) > c(y, \omega)],$$

where x and y are strategies for player A and B, respectively. We will consider the following two games, introduced in [25].

4.1.1. Binary Search Tree Duel. In the binary search tree duel, there is a set of elements Ω and a probability distribution p over Ω. Each player is going to construct a binary search tree containing the elements of Ω. Strategy x beats strategy y for element $\omega \in \Omega$ if and only if the path from ω to the root in x is shorter than the path from ω to the root in y. Thus, the set of strategies X is the set of all binary search trees with elements of Ω, and $c(x, \omega)$ is defined to be the depth of element ω in strategy x.

4.1.2. Ranking Duel. In the ranking duel, there is a set of m pages Ω, and a probability distribution p over Ω, notifying the probability that each page is going to be searched. In the ranking duel, two search engines compete against each other. Each search engine has to provide a permutation of these pages, and a player beats the other if page ω comes earlier in her permutation. In other words, the set of strategies X contains all $m!$ permutations of the pages, and for any permutation $x = (x_1, x_2, \ldots, x_m)$ and page ω, $c(x, \omega) = i$ if and only if $\omega = x_i$.

4.2. Dueling Games Are Polynomially Separable

Consider a dueling game in which each strategy $\hat{x}$ of player A is an $n(A)$ dimensional point in Euclidean space. Let S_A be the convex hull of these strategy points. Thus each point $\hat{\mathbf{x}}$ in S_A is a mixed strategy of player A. Similarly define strategy $\hat{\mathbf{y}}$, $n(B)$, and S_B for player B. A dueling game is *bilinear* if utility function $h^A(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ has the form $\hat{\mathbf{x}}^t M \hat{\mathbf{y}}$, where M is an $n(A) \times n(B)$ matrix. Again for player B, we have $h^B(\hat{\mathbf{x}}, \hat{\mathbf{y}}) = -h^A(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. Immorlica et al. [25] provide a method for finding an equilibrium of a class of bilinear games, which is defined as follows.

Definition 1. Polynomially Representable Bilinear Dueling Games). A bilinear dueling game is *polynomially representable* if one can represent the convex hull of strategies S_A and S_B with a polynomial number m of linear constraints; that is, there are m vectors $\{v_1, v_2, \ldots, v_m\}$ and m real numbers $\{b_1, b_2, \ldots, b_m\}$ such that $S_A = \{\hat{\mathbf{x}} \in R^{n(A)} | \forall i \in \{1, 2, \ldots, m\}, v_i \cdot \hat{\mathbf{x}} \geq b_i\}$. Similarly $S_B = \{\hat{\mathbf{y}} \in R^{n(B)} | \forall i \in \{1, 2, \ldots, m'\}, v'_i \cdot \hat{\mathbf{y}} \geq b'_i\}$.

In the following theorem, we show that every polynomially representable bilinear duel is also polynomially separable, as defined in Section 2.2. This implies that the general reduction described in Section 2.2 can be used to solve polynomially representable bilinear duels as well.

Theorem 4. *Every polynomially representable bilinear duel is polynomially separable.*

Proof. Let S_A and S_B be the set of strategy points for player A and player B, respectively. We show that if S_A can be specified with a polynomial number of linear constraints, then one could design an algorithm that finds out whether there exists a point $\hat{\mathbf{x}} \in S_A$ such that $\alpha_0 + \sum \alpha_i \hat{\mathbf{x}}_i \geq 0$. Let $\{(v_1, b_1), (v_2, b_2), \ldots, (v_m, b_m)\}$ be the set of constraints that specify S_A, where v_i is a vector of size $n(A)$ and b_i is a real number. We need to check whether there exists a point satisfying both constraints in $\{(v_1, b_1), (v_2, b_2), \ldots, (v_m, b_m)\}$ and $\alpha_0 + \sum \alpha_i \hat{\mathbf{x}}_i \geq 0$. Recall that m is assumed to be polynomial in the natural game representation. Because all these constraints are linear, we can solve this feasibility problem by an LP in polynomial time. The same argument holds for S_B; therefore, every polynomially representable bilinear duel is polynomially separable as well. □

4.3. A Simplified Argument for Ranking and Binary Search Duels

In this section, we revisit some examples of dueling games and demonstrate how Theorem 2 can be used to solve them in polynomial time. To apply Theorem 2, one should take two main steps. First, it is necessary to express the duel as a bilinear game: that is, one must transfer every strategy of the players to a point in $n(A)$ and $n(B)$ dimensional space, such that the outcome of the game can be determined for two given strategy points with an $n(A) \times n(B)$ matrix M. Second, one must implement an oracle that determines whether there exists a strategy point satisfying a given linear constraint.

To illustrate our method more precisely, we propose a polynomial-time algorithm for finding an NE for ranking and binary search tree dueling games in what follows.

Theorem 5. *There exists an algorithm that finds an NE of the ranking duel in polynomial time.*

Proof. We transfer each strategy $\mathbf{x}$ of player A to point $\hat{\mathbf{x}}$ in $\mathbb{R}^{m^2}$ where $\hat{\mathbf{x}}_{i,j}$ denotes the probability that ω_i stands at position j in $\mathbf{x}$. The outcome of the game is determined by the following equation:

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=j+1}^{m} \hat{\mathbf{x}}_{i,j} \hat{\mathbf{y}}_{i,k} p(\omega_i) - \sum_{i=1}^{m} \sum_{j=1}^{m} \sum_{k=1}^{j-1} \hat{\mathbf{x}}_{i,j} \hat{\mathbf{y}}_{i,k} p(\omega_i),$$

where $p(\omega_i)$ denotes the probability that ω_i is searched.

Here, we need to provide an oracle that determines whether there exists a strategy point for a player that satisfies a given linear constraint $\alpha_0 + \sum \alpha_{i,j}\hat{\mathbf{x}}_{i,j} \geq 0$. Because each pure strategy is a matching between pages and indices, we can find the pure strategy that maximizes $\sum \alpha_{i,j}\hat{\mathbf{x}}_{i,j}$ with the maximum weighted matching algorithm. Therefore, this query can be answered in polynomial time. Because we have reduced this game to a polynomially separable bilinear duel, we can find a Nash equilibrium in polynomial time. □

Theorem 6. *There exists an algorithm that finds an NE of the binary search tree duel in polynomial time.*

Proof. Here we map each strategy $\mathbf{x}$ to the point $\hat{\mathbf{x}} = \langle \hat{\mathbf{x}}_{1,1}, \hat{\mathbf{x}}_{1,2}, \ldots, \hat{\mathbf{x}}_{1,m}, \hat{\mathbf{x}}_{2,1}, \hat{\mathbf{x}}_{2,2}, \ldots, \hat{\mathbf{x}}_{m,m}\rangle \in \mathbb{R}^{m^2}$, where $\hat{\mathbf{x}}_{i,j}$ denotes the probability that depth of the ith element is equal to j. Therefore, the payoff of the game for strategies $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ is equal to

$$\sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=j+1}^{m} \hat{\mathbf{x}}_{i,j}\hat{\mathbf{y}}_{i,k}p(\omega_i) - \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{j-1} \hat{\mathbf{x}}_{i,j}\hat{\mathbf{y}}_{i,k}p(\omega_i),$$

where $p(\omega_i)$ denotes the probability that ith element is searched.

Next, we need to provide an oracle that determines whether there exists a strategy point for a player that satisfies a given linear constraint $\alpha_0 + \sum \alpha_{i,j}\hat{\mathbf{x}}_{i,j} \geq 0$. To do this, we find the binary search tree that maximizes $\sum \alpha_{i,j}\hat{\mathbf{x}}_{i,j}$. This is done with a dynamic program. Let $D(a,b,k)$ denote the maximum value of $\sum_{i=a}^{b} \alpha_{i,j}\hat{\mathbf{x}}_{i,j}$ for a subtree that its root is at depth k. $D(a,b,k)$ is formulated as

$$D(a,b,k) = \begin{cases} \min_{a\leq c\leq b}\{D(a,c-1,k+1) + D(c+1,b,k+1) + \alpha_{c,k}\} & \text{if } a < b, \\ \alpha_{a,k} & \text{if } a = b. \end{cases}$$

Therefore, we find the binary search tree that maximizes $\sum \alpha_{i,j}\hat{\mathbf{x}}_{i,j}$ in polynomial time and determine whether it meets the constraint. Because we have reduced this game to a polynomially separable bilinear duel, we can find an NE in polynomial time. □

4.4. Matching Duel

In a matching duel, two players play a game on a bidirectional graph $G = (V, E, W)$ whose every edge has an arbitrary weight. Each of the players selects a perfect matching of G as his strategy, and the payoffs are determined according to the matchings of the players. More precisely, once both players make their decisions, a random node ω is drawn from a probability distribution Ω. Next, each player gets a score equal to the weight of the edge incident to ω in his matching, and the winner is the one with the higher score. In case of a tie, the game is a draw and both players receive a payoff of 0; otherwise the winner gets a payoff of 1 and the loser gets a payoff of −1. In other words, a strategy x beats a strategy y for element $\omega \in \Omega$ if ω is matched to a higher weighted edge in strategy x than strategy y.

The matching duel may find its application in a competition between websites that try to match people according to their desire. In this competition, the goal of each website is to maximize its number of users, and a website attracts a user if it suggests a match that is better than other websites' suggestions. We mention that the ranking duel is a special case of the matching duel when G is a complete bipartite graph with n nodes on each side, in which the first part denotes the web pages and the second part denotes the ranking positions. Thus, the weight of the edge between page i and rank j is equal to j.

Note that Rothvoss [42] showed that the feasible strategy polytope (the perfect matching polytope) has exponentially many facets. Therefore, any approach that requires a polynomial number of constraints, such as the method described by Immorlica et al. [25], cannot directly apply. However, it turns out that the matching duel is polynomial separable, and thus our technique can be used to compute an equilibrium in polynomial time.

Theorem 7. *There exists an algorithm that computes a Nash equilibrium of the matching duel in polynomial time.*

Proof. We transfer every strategy $\mathbf{x}$ to a point in $|E|$-dimensional Euclidean space $\hat{\mathbf{x}}$, where $\hat{\mathbf{x}}_e$ denotes the probability that $\mathbf{x}$ chooses e in the matching. Thus, the payoff function is bilinear and is as follows:

$$\sum_{\omega\in\Omega}\sum_{e_1\in N(\omega)}\sum_{e_2\in N(\omega)} [p(\omega)\hat{\mathbf{x}}_{e_1}\hat{\mathbf{y}}_{e_2} \times \operatorname{sign}(w(e_1) - w(e_2))],$$

where $N(\omega)$ is the set of edges adjacent to ω.[6] Next, we need to prove that the game is polynomially separable. Given a vector α and a real number α_0, we are to find out whether there is a strategy $\hat{\mathbf{x}}$ such that $\alpha_0 + \alpha \cdot \hat{\mathbf{x}} \geq 0$.

This problem can be solved by a maximum weighted perfect matching, where the graph is $G = (V, E, W)$ and $w(e) = \alpha_e$. Thus using our framework, we find an NE of the matching duel in polynomial time. □

5. Approximating the Payoff of the Game

We observe that in some dueling games the separation problem cannot be solved in polynomial time. However, having a fixed error threshold, it is possible to approximately solve the separation problem in polynomial time. In this section, we present a technique for approximating the payoff of the dueling games via providing an approximate solution for the separation problem.

We say a (not necessarily feasible) point $\hat{\mathbf{x}}$ is an ϵ-solution of the LP in a bilinear dueling game where there exists a feasible strategy point $\hat{\mathbf{x}}'$ such that $|\hat{\mathbf{x}} - \hat{\mathbf{x}}'| \le \epsilon$ and the minimum payoff of strategy $\hat{\mathbf{x}}$ differs by at most ϵ from that of the MinMax strategies. Note that in the separation problem we are given a linear constraint $\alpha_0, \alpha_1, \ldots$ and are to find out whether there exists a strategy point for a player which has the property $\alpha_0 + \sum_i \alpha_i \hat{\mathbf{x}}_i \ge 0$. We say a separation oracle solves this problem with approximation factor ϵ if it always finds such a point when there exists a violating point having a distance of at least ϵ from the hyperplane. Note that the approximate oracle may not report a correct answer when the distance of all desired points to the hyperplane is less than ϵ. We say a dueling game is *polynomially ϵ-separable* if we can approximately solve the separation problem in polynomial time for every $\epsilon > 0$. The following theorem provides a strong tool for approximating the payoff of the game in an NE for a broad set of dueling games.

Theorem 8. *Let B be a bilinear dueling game ranging over $[0,1]^{n(A)} \times [0,1]^{n(B)}$ (i.e., $S_A \subset [0,1]^{n(A)}$ and $S_B \subset [0,1]^{n(B)}$) with payoff matrix M and $n = \max\{n(A), n(B)\}$. If we can solve the ϵ-separation problem in time $f(\epsilon, n)$, then we can find an ϵ-solution of the LP in time $\mathbb{O}(\mathrm{poly}(n) \cdot f(\epsilon/\beta, n))$, where $\beta = \max\{1, \sum_{i,j} |M_{i,j}|\}$.*

Proof. Suppose we solve LP 2 with the ellipsoid method using an ϵ/β-separating oracle instead of the exact oracle, and let $\hat{\mathbf{x}}$ be the solution we find. First we show that there exists at least one feasible strategy $\hat{\mathbf{x}}'$ such that $|\hat{\mathbf{x}}' - \hat{\mathbf{x}}| \le \epsilon$. Next, we will compare the minimum payoff of $\hat{\mathbf{x}}'$ with that of the MinMax strategies. Note that to determine whether $\hat{\mathbf{x}}$ is within the set of feasible strategy points, we solve the following linear program (which is the same as LP 5 from the proof of Lemma 9) and determine whether there is any violating constraint.

$$\begin{aligned} &\alpha_0 + \sum_j \alpha_j \hat{\mathbf{x}}_j < 0 \\ &\alpha_0 + \sum_j \alpha_j \hat{v}_j \ge 0 \quad \forall \hat{v} \in I_A \end{aligned} \tag{13}$$

Because the ϵ/β-separating oracle does not report any violating constrains for $\hat{\mathbf{x}}$, either there is no violating constrain or the distance of $\hat{\mathbf{x}}$ from all violating hyperplanes is less than ϵ/β. Therefore, there exists a point $\hat{\mathbf{x}}'$ in the set of feasible solution such that $|\hat{\mathbf{x}}' - \hat{\mathbf{x}}| \le \epsilon/\beta \le \epsilon$.

Next, we compare the minimum payoff of strategy $\hat{\mathbf{x}}'$ with the minimum payoff of the MinMax strategies. From the previous argument we have $|\hat{\mathbf{x}}' - \hat{\mathbf{x}}| < \epsilon/\beta$. Because $\beta = \sum_{i,j} |M_{i,j}|$, we have $|(\hat{\mathbf{x}} M \hat{\mathbf{y}}) - (\hat{\mathbf{x}}' M \hat{\mathbf{y}})| < |\hat{\mathbf{x}}' - \hat{\mathbf{x}}| \cdot \beta < \epsilon$ for each strategy $\hat{\mathbf{y}}$ of the second player. Moreover, because every MinMax strategy is a potential solution of the linear program and $\hat{\mathbf{x}}$ is the solution that maximizes the objective function, the minimum possible payoff off $\hat{\mathbf{x}}$ is not less than the minimum possible payoff of the MinMax strategies. Thus, the difference between the minimum possible payoff of $\hat{\mathbf{x}}'$ and that of the MinMax strategies is at most ϵ. Because we are using the ellipsoid method, the number of times we call the ϵ-separation oracle is $\mathrm{poly}(n)$, hence the running time of the algorithm is $\mathbb{O}(\mathrm{poly}(n) \cdot f(\epsilon/\beta, n))$. □

Note that, by Theorem 8, if we can find an approximate solution of the separation problem in polynomial time, then we can find an approximate solution of the LP in polynomial time as well. Because an approximate solution may not necessarily be a feasible strategy point, it cannot be used to characterize the properties of the MinMax strategies. However, we can approximate the payoff of the players in an NE by computing the payoffs for the ϵ-solution of the LP.

6. General Lotto

In this section, we study the General Lotto game. An instance of the General Lotto game is defined by $\Gamma(a, b, u)$, where players A and B simultaneously define probability distributions of nonnegative integers X and Y, respectively, such that $\mathbb{E}[X] = a$ and $\mathbb{E}[Y] = b$. In this game, player A's aim is to maximize $h_\Gamma^A(X, Y)$ and player B's aim is to maximize $h_\Gamma^B(X, Y) = -h_\Gamma^A(X, Y)$, where $h_\Gamma^A(X, Y)$ is defined as $\mathbb{E}_{i \sim X, j \sim Y} u(i, j)$. The previous studies of the General Lotto game considered a special case of the problem in which $u(i,j) = sign(i-j)$.[7] Here, we generalize the payoff function to a bounded distance function and present an algorithm for finding a Nash equilibrium of the General Lotto game in this case. Function u is a bounded distance function, if one can write it as $u(i,j) = f_u(i-j)$ such that f_u is a monotone function and reaches its maximum value at $f_u(t_u)$, where $t_u \in O(\text{poly}(a,b))$.

We first define a new version of the General Lotto game, which is called the *finite General Lotto* game. We prove a Nash equilibrium of the finite General Lotto game can be found in polynomial time. Then we reduce the problem of finding a Nash equilibrium for the General Lotto game with bounded distance function to the problem of finding a Nash equilibrium for the finite General Lotto game. This helps us to propose a polynomial-time algorithm for finding a Nash equilibrium of the General Lotto game with bounded distance function.

6.1. Finite General Lotto

We define the finite General Lotto game $\Gamma(a,b,u,S)$ to be an instance of the General Lotto game in which every strategy of players is a distribution over a finite set of numbers S. Here, we leverage our general technique to show the finite General Lotto game is a polynomially separable bilinear game and, as a consequence, it leads to a polynomial-time algorithm to find a Nash equilibrium for this game.

Theorem 9. *There exists an algorithm that finds a Nash equilibrium of the finite General Lotto game $\Gamma(a,b,u,S)$ in time $O(poly(|S|))$.*

Proof. First we map each strategy X to a point $\hat{x} = \langle \hat{x}_1, \hat{x}_2, \dots, \hat{x}_{|S|}\rangle$, where $\hat{x}_i$ denotes $\Pr(X = S_i)$. Without loss of generality we assume the elements of S are sorted in strictly ascending order (i.e., for each $1 \le i < j \le |S|$, $S_i < S_j$). Now the utility of player A when A plays a strategy corresponding to $\hat{x}$ and B plays a strategy corresponding to $\hat{y}$ is obtained by the following linear function:

$$h^A_\Gamma(\hat{x},\hat{y}) = \sum_{i=1}^{|S|}\sum_{j=1}^{i-1} \hat{x}\hat{y}u(i,j) - \sum_{i=1}^{|S|}\sum_{j=i+1}^{|S|} \hat{x}\hat{y}u(i,j).$$

Therefore the game is bilinear. Now we prove the game is polynomially separable.

Given a real number r and a vector v, we provide a polynomial-time algorithm that determines whether there exists a strategy point $\hat{x}$ such that $r + v\cdot\hat{x} \ge 0$. We design the following feasibility program for this problem with $|S|$ variables $\hat{x}_1$ to $\hat{x}_{|S|}$ and three constraints.

$$\sum_{i=1}^{|S|} \hat{x}_i = 1 \tag{14}$$

$$\sum_{i=1}^{|S|} \hat{x}_i S_i = a \tag{15}$$

$$r + v\cdot\hat{x} \ge 0 \tag{16}$$

Constraints 14 and 15 force the variables to represent a valid strategy point (i.e., the probabilities sum to 1 and the expectation equals a). Thus every point $\hat{x}$ is a valid strategy point if and only if it satisfies constraints (14) and (15). On the other hand, constraint (16) enforces the program to satisfy the given linear constraint of the separation problem. Thus there exists a strategy point $\hat{x}$ such that $b + v\cdot\hat{x} \ge 0$ if and only if there is a solution for the feasibility LP. The feasibility of the program can be determined in polynomial time, hence the separation problem is polynomially tractable.

Therefore the finite General Lotto game is a polynomially separable bilinear game, and by Theorem 2, there exists a polynomial-time algorithm that finds a Nash equilibrium of the finite General Lotto game. □

6.2. General Lotto with Bounded Distance Functions

Below is the formal definition of bounded distance functions.

Definition 2. Function u is a bounded distance function, if one can write it as $u(i,j) = f_u(i-j)$ such that f_u is a monotone function and reaches its maximum value at $u^M = f_u(u^T)$, where $u^T \in O(\text{poly}(a,b))$. We call u^T the threshold of function u and u^M the maximum of function u.[8]

The following theorem shows the main result for the General Lotto game with bounded distance funciton.

Theorem 10. *There is a polynomial-time algorithm that finds a Nash equilibrium of the General Lotto game $\Gamma(a, b, u)$, where u is a bounded distance function.*

To prove this theorem, we require to first state several lemmas. The main proof idea is to show there is a bound on the optimal strategies in the Nash equilibria of General Lotto games with bounded distance functions. We show that for bounded distance function this bound actually exists and is polynomial in the size of input. Then we use our results for the finite General Lotto problem to solve the General Lotto problem with bounded distance function.

Next we state the necessary steps to prove Theorem 10. First, we define a specific class of strategies called *paired strategies* and claim that for every player's strategy there is a best-response strategy that is a paired strategy. Using this observation, we can bound the set of optimal strategies.

Consider a probability distribution that only allows two possible outcomes; that is, there are only two elements in S with nonzero probabilities. We call such distribution a *paired strategy*. Let $T_{i,j}$ be a paired strategy that only has nonzero probabilities at elements i and j. Also let $T^a_{i,j}$ be such strategy with $\mathbb{E}[T^a_{i,j}] = a$. Given a paired strategy $T^a_{i,j}$, we have $\Pr(T^a_{i,j} = i) \cdot i + \Pr(T^a_{i,j} = j) \cdot j = a$; and because we know $\Pr(T^a_{i,j} = i) + \Pr(T^a_{i,j} = j) = 1$, these probabilities can be computed as functions of i, j, and a. Some algebraic manipulation yields $\Pr(T^a_{i,j} = i) = (a - j)/(i - j)$ and $\Pr(T^a_{i,j} = j) = (a - i)/(j - i)$. To shorten the notation let $\alpha^a_{i,j} = \Pr(T^a_{i,j} = i)$ and $\alpha^a_{j,i} = \Pr(T^a_{i,j} = j)$. In the following structural lemma, we show that every distribution T over a finite set S can be constructed by a set of paired strategies.

Lemma 14. *For every distribution T over S with $\mathbb{E}[T] = a$ and t elements with nonzero probabilities, there exist an integer $m \le t$ m and paired strategies $\sigma_1, \sigma_2, \dots, \sigma_m$ with corresponding probabilities $\beta_1, \beta_2, \dots, \beta_m$ such that $T = \sum_{r=1}^m \beta_r \sigma_r$,[9] and for all $1 \le i \le m$ we have $\beta_i \in [0, 1]$ and $\mathbb{E}[\sigma_i] = a$.*

Proof. We prove this claim by an induction on the number of elements with nonzero probabilities in T. If there is only one element with nonzero probability in T (i.e., $t = 1$), then we have $\Pr(T = a) = 1$. Thus $T = T^a_{0,a}$ is a paired strategy, and the claim holds by setting $\sigma_1 = T^a_{0,a}$ and $\beta_1 = 1$.

Now assuming the claim holds for all $1 \le t' < t$, we prove the claim also holds for t. Let T be a probability distribution with t nonzero probability elements. Because $t \ge 2$, there should be some $i < a$ with $\Pr(T = i) > 0$ and some $j > a$ with $\Pr(T = j) > 0$. We choose the largest possible number $0 < \beta \le 1$ such that $\beta\alpha^a_{i,j} \le \Pr(T = i)$ and $\beta\alpha^a_{j,i} \le \Pr(T = j)$. If $\beta = 1$, then $\Pr(T = i) + \Pr(T = j) \ge \alpha^a_{i,j} + \alpha^a_{j,i} = 1$. This means $T = T^a_{i,j}$ is a paired strategy, and the claim holds by setting $\sigma_1 = T^a_{i,j}$ and $\beta_1 = 1$. Otherwise, we can write $T = (1 - \beta)T' + \beta T^a_{i,j}$, where $T' = (T - \beta T^a_{i,j})/(1 - \beta)$. Furthermore we have

$$\mathbb{E}[T'] = \frac{\mathbb{E}[T - \beta T^a_{i,j}]}{1 - \beta} = \frac{\mathbb{E}[T] - \beta E[T^a_{i,j}]}{1 - \beta} = a.$$

We select β such that at least one of the probabilities $\Pr(T' = i)$ or $\Pr(T' = j)$ becomes zero. Thus, compared with T, the number of elements with nonzero probability in T' is at least decreased by one, and by the induction hypothesis we can write $T' = \beta'_1\sigma'_1 + \beta'_2\sigma'_2 + \dots + \beta'_{m'}\sigma'_{m'}$, where $m' \le t - 1$. Let $\beta_i = (1 - \beta)\beta'_i$ and $\sigma_i = \sigma'_i$ for $1 \le i \le m'$ and $\beta_{m'+1} = \beta$ and $\sigma_{m'+1} = T^a_{i,j}$. Now we can write $T = \beta_1\sigma_1 + \beta_2\sigma_2 + \dots + \beta_{m'+1}\sigma_{m'+1}$, where each σ_i is a paired strategy and $E(\sigma_i) = a$. Furthermore, $m = m' + 1 \le t$, and the proof is complete. Because $t \le |S|$, m is polynomial in the size of input. Therefore paired strategies $\sigma_1, \sigma_2, \dots, \sigma_m$ and their corresponding coefficients $\beta_1, \beta_2, \dots, \beta_m$ can be computed in polynomial time. □

Lemma 15. *For every strategy of player A in a finite General Lotto game there is a best-response strategy of player B that is a paired strategy.*

Proof. Consider finite General Lotto game $\Gamma(a, b, u, S)$, strategy X of player A, and a best-response strategy Z of player B. Because Z is a distribution on S, by using Lemma 14 we can write $Z = \sum_{r=1}^m \beta_r\sigma_r$. Thus, we have $h^B_\Gamma(X, Z) = h^B_\Gamma(X, \sum_{r=1}^m \beta_r\sigma_r)$, and because of the linearity of expectation we can write $h^B_\Gamma(X, Z) = \sum_{r=1}^m \beta_r h^B_\Gamma(X, \sigma_r)$. Because Z is a best-response strategy, we have

$$\forall 1 \le r \le m, \quad h^B_\Gamma(X, \sigma_r) = h^B_\Gamma(X, Z). \tag{17}$$

This means paired strategy σ_r, for each $1 \le r \le m$, is a best-response strategy of player B. □

In the following lemmas, using the structural property of the best-response strategies, we show some bounds for each player's optimal strategies.

Lemma 16. *For any strategy X with $\mathbb{E}[X] = c$ and any integer j, we have $\sum_{i=0}^{j} \Pr(X = i) \geq 1 - \frac{c}{j+1}$.*

Proof. Because $\sum_{i=0}^{+\infty} i\Pr(X = i) = c$, we have $(j+1)\sum_{i=j+1}^{+\infty} \Pr(X = i) \leq c$. This implies

$$\sum_{i=0}^{j} \Pr(X = i) = 1 - \sum_{i=j+1}^{+\infty} \Pr(X = i) \geq 1 - \frac{c}{j+1}. \quad \square$$

Lemma 17. *Consider Nash equilibrium (X, Y) of General Lotto game $\Gamma(a, b, u)$, where u is a bounded distance function with threshold u^T. We have $\sum_{i=0}^{a-1} \Pr[Y = i] \leq \frac{u^T}{u^T+1}$.*

Proof. Let X' be a pair distribution of player A that chooses $a-1$ with probability p and chooses $a + u^T - 1$ with probability $1-p$. Thus $p = (u^T - 1)/u^T$. The payoff of playing strategy X' against Y is

$$h_\Gamma^A(X', Y) = \sum_{i=0}^{+\infty} \Pr(Y = i)[pu(a-1, i) + (1-p)u(a + u^T - 1, i)]. \tag{18}$$

Note that by the definition of u, $u(i, j) \geq 0$ if and only if $i - j \geq 0$ and $u(i, j) \leq 0$ if and only if $i - j \leq 0$. Furthermore, if $i - j \geq u^T$, then $u(i, j) = u^M$, and if $i - j \leq -u^T$, then $u(i, j) = -u^M$. Therefore,

$$\begin{aligned} \sum_{i=0}^{a-1} \Pr(Y = i)pu(a-1, i) &\geq 0 \\ \sum_{i=0}^{a-1} \Pr(Y = i)(1-p)u(a + u^T - 1, i) &\geq (1-p)u^M \sum_{i=0}^{a-1} \Pr(Y = i) \\ \sum_{i=a}^{+\infty} \Pr(Y = i)[pu(a-1, i) + (1-p)u(a + u^T - 1, i)] &\geq -u^M \sum_{i=a}^{+\infty} \Pr(Y = i). \end{aligned} \tag{19}$$

Note that $a \leq b$ and (X, Y) is a Nash equilibrium, which means $h_\Gamma^A(X, Y) \leq 0$. This implies $h_\Gamma^A(X', Y) \leq 0$. Thus, by applying equality (18) and inequality (19) we have

$$0 \geq h_\Gamma^A(X', Y) \geq (1-p)u^M \sum_{i=0}^{a-1} \Pr(Y = i) - u^M \sum_{i=a}^{+\infty} \Pr(Y = i),$$

which implies $\sum_{i=0}^{+\infty} \Pr(Y = i) \geq (2-p)\sum_{i=0}^{a-1} \Pr(Y = i)$. Thus, $\sum_{i=0}^{a-1} \Pr(Y = i) \leq 1/(2-p)$. By substituting $(u^T - 1)/u^T$ instead of p we can conclude $\sum_{i=0}^{a-1} \Pr(Y = i) \leq u^T/(u^T + 1)$. $\square$

In the following lemma, we provide an upper bound for the maximum variable with nonzero probability of a player's strategy in the equilibrium.

Lemma 18. *Consider a Nash equilibrium (X, Y) of General Lotto game $\Gamma(a, b, u)$, where u is a bounded distance function with threshold u^T. If $\hat{u} = (4bu^T + 4b + u^T)(2u^T + 2)$, then we have $\Pr(Y > \hat{u} + u^T) = 0$ and $\Pr(X > \hat{u}) = 0$.*

Proof. First, we prove for any integer $z > \hat{u}$, $\Pr(X = z) = 0$. The proof is by contradiction. Let $z > \hat{u}$ be an integer with nonzero probability in X. Thus there is an integer $x < a$ with nonzero probability in X. Consider the pair distribution $T_{x,z}^a$. We define another pair distribution $T_{x,y}^a$, where $y = 4bu^T + 4b + u^T$.

Consider strategy $X^\epsilon = X - \epsilon T_{x,z}^a + \epsilon T_{x,y}^a$. Note that (X, Y) is a Nash equilibrium of the game. This means strategy X is a best response of player A to strategy Y of player B, which implies $h_\Gamma^A(X, Y) \geq h_\Gamma^A(X^\epsilon, Y)$. On the other hand, because of the linearity of expectation we can write

$$h_\Gamma^A(X^\epsilon, Y) = h_\Gamma^A(X, Y) - \epsilon h_\Gamma^A(T_{x,z}^a, Y) + \epsilon h_\Gamma^A(T_{x,y}^a, Y).$$

Therefore, we conclude $w = h^A_\Gamma(T^a_{x,z}, Y) - h^A_\Gamma(T^a_{x,y}, Y) \geq 0$. Let $p = \alpha^a_{z,x}$ and $q = \alpha^a_{y,x}$. We have

$$w = \sum_{i=0}^{+\infty} \Pr(Y = i)[(1-p)u(x,i) + pu(z,i)] - \sum_{i=0}^{+\infty} \Pr(Y = i)[(1-q)u(x,i) + qu(y,i)] \geq 0.$$

We write w as $w = w_1 + w_2 - w_3 - w_4 + w_5$, where

$$\begin{aligned}
w_1 &= \sum_{i=0}^{x} \Pr(Y = i)[[(1-p)u(x,i) + pu(z,i)] - [(1-q)u(x,i) + qu(y,i)]],\\
w_2 &= \sum_{i=x+1}^{+\infty} \Pr(Y = i)[(1-p)u(x,i) - (1-q)u(x,i)],\\
w_3 &= \sum_{i=x+1}^{y-u^T-1} \Pr(Y = i)qu(y,i),\\
w_4 &= \sum_{i=y-u^T}^{+\infty} \Pr(Y = i)qu(y,i),\\
w_5 &= \sum_{i=x+1}^{+\infty} \Pr(Y = i)pu(z,i).
\end{aligned}$$

Because $1 - p \geq 1 - q$ and $u(z,i) = u(y,i) = u^M$ for all $i \leq x$, we can conclude $w_1 \leq 0$. For all $i > x$, we have $u(x,i) \leq 0$, and we also know $1 - p \geq 1 - q$. These mean $w_2 \leq 0$. Because for all $i \leq y - u^T$ we have $u(y,i) = u^M$, we conclude

$$w_3 = qu^M \sum_{i=x+1}^{y-u^T-1} \Pr(Y = i). \tag{20}$$

Moreover, for any arbitrary integers i and j, we have $-u^M \leq u(i,j) \leq u^M$. Thus,

$$-w_4 \leq qu^M \sum_{i=y-u^T}^{+\infty} \Pr(Y = i) \tag{21}$$

$$w_5 \leq pu^M \sum_{i=x+1}^{+\infty} \Pr(Y = i). \tag{22}$$

Therefore, by knowing $w \geq 0$, $w_1 \leq 0$, $w_2 \leq 0$, and considering inequalities (20), (21), and (22), we conclude

$$u^M\left(-q \sum_{i=x+1}^{y-u^T-1} \Pr(Y = i) + q \sum_{i=y-u^T}^{+\infty} \Pr(Y = i) + p \sum_{i=x+1}^{+\infty} \Pr(Y = i)\right) \geq w \geq 0. \tag{23}$$

From the definition of y we have $y - u^T - 1 = 4bu^T + 4b - 1$ and by Lemma 16, $\sum_{i=0}^{y-u^T-1} \Pr(Y = i) \geq 1 - b/(4bu^T + 4b) = (4u^T + 3)/(4u^T + 4)$. Note that $x < a$, which means $\sum_{i=0}^{x} \Pr(Y = i) \leq \sum_{i=0}^{a-1} \Pr(Y = i)$. On the other hand, Lemma 17 states that $\sum_{i=0}^{a-1} \Pr(Y = i) \leq u^T/(u^T + 1)$ holds. Hence we can conclude $\sum_{i=0}^{x} \Pr(Y = i) \leq u^T/(u^T + 1)$. Therefore

$$\sum_{i=x+1}^{y-u^T-1} \Pr(Y = i) = \sum_{i=0}^{y-u^T-1} \Pr(Y = i) - \sum_{i=0}^{x} \Pr(Y = i) \geq \frac{3}{4u^T + 4} \tag{24}$$

and

$$\sum_{i=y-u^T}^{+\infty} \Pr(Y = i) = 1 - \sum_{i=0}^{y-u^T-1} \Pr(Y = i) \leq \frac{1}{4u^T + 4}. \tag{25}$$

By inequalities (23), (24), (25) and $\sum_{i=x+1}^{+\infty} \Pr(Y = i) \leq 1$, we have

$$-q\frac{3}{4u^T + 4} + q\frac{1}{4u^T + 4} + p \geq 0, \tag{26}$$

which implies $q/p \le 2u^T + 2$. Recalling $p = \alpha^a_{z,x} = (a-x)/(z-x)$, $q = \alpha^a_{y,x} = (a-x)/(y-x)$, and $z > y$, we can bound z/y as follows: $z/y \le (z-x)/(y-x) = q/p \le 2u^T + 2$. Therefore $z \le y(2u^T + 2) = \hat{u}$, which is a contradiction. Knowing that player A put zero probability on every number $z > \hat{u}$ and considering the definition of bounded distance function u, player B will put zero probability of every number greater than $\hat{u} + u^T$ in any Nash equilibrium. □

Theorem 10 follows after Theorem 9 and Lemma 18. Below is the formal proof of Theorem 10.

Proof of Theorem 10. Let $\bar{u} = (4bu^T + 4b + u^T)(2u^T + 2) + u^T$. Lemma 18 shows there is a bound on the optimal strategies in a Nash equilibrium. More precisely, $\Pr(Y > \bar{u}) = 0$, where Y is a strategy of player A or B. Thus, General Lotto game $\Gamma(a, b, u)$ is equivalent to finite General Lotto game $\Gamma(a, b, f, S)$, where $S = \{1, 2, \dots, \bar{u}\}$. By Theorem 9, a polynomial-time algorithm finds a Nash equilibrium of the game. □

7. Separation Oracles

In this section, we describe, in precise detail, the separating oracles used by the ellipsoid method to solve our represented linear programs. Consider we are given a sequence $c_0, c_1, \dots, c_{k(m+1)}$, where k is the number of battlefields and m is the number of troops for a player. We first present an algorithm that finds a pure strategy $x = (x_1, x_2, \dots, x_k) \in \mathcal{X}$ such that $\sum_{i=1}^{k} x_i = m$, and $\hat{x} = \mathcal{G}(x)$ minimizes the expression

$$c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i. \tag{27}$$

We will then leverage this algorithm and design polynomial-time algorithms for the hyperplane separating oracle and best-response separating oracle. The following lemma shows that Algorithm 1 (FindBestPure) finds the minimizer of expression (27).

Algorithm 1 (FindBestPure)

```
input: $m$, $k$, $c_0, c_1, c_2, \dots, c_{k(m+1)}$
 1: for $j \leftarrow 1$ to $m$ do
 2:     $d[0, j] \leftarrow c_0$
 3: end for
 4: for $i \leftarrow 1$ to $k$ do
 5:     for $t \leftarrow 0$ to $m$ do
 6:         for $j \leftarrow 0$ to $t$ do
 7:             if $d[i-1, t-j] + c_{(i-1)(m+1)+j+1} < d[i, t]$ then
 8:                 $d[i, t] \leftarrow d[i-1, t-j] + c_{(i-1)(m+1)+j+1}$
 9:                 $r[i, t] \leftarrow j$
10:             end if
11:         end for
12:     end for
13: end for
14: $rem \leftarrow m$
15: for $i \leftarrow k$ downto 1 do
16:     $x_i \leftarrow r[i, rem]$
17:     $rem \leftarrow rem - r[i, rem]$
18: end for
19: return $x = (x_1, x_2, \dots, x_k)$
```

Lemma 19. *Given two integers m and k and a sequence $c_0, c_1, \dots, c_{k(m+1)}$, algorithm* FindBestPure *correctly finds an optimal pure strategy $x = (x_1, x_2, \dots, x_k)$, where $\sum_{i=1}^{k} x_i = m$, $\hat{x} = \mathcal{G}(x)$ and $\hat{x}$ minimizes $c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i$.*

Proof. In algorithm FindBestPure, using a dynamic programming approach, we define $d[i, t]$ to be the minimum possible value of $c_0 + \sum_{i'=1}^{i(t+1)} c_{i'} \hat{x}_{i'}$, where $\sum_{i'=1}^{i} x_{i'} = t$. Hence, $d[k, m]$ denotes the minimum possible value of $c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i$. Now, we show that algorithm FindBestPure correctly computes $d[i, t]$ for all $0 \le i \le k$ and $0 \le t \le m$. Obviously $d[0, j]$ is equal to c_0. For an arbitrary $i > 0$ and t, the optimal strategy x puts $t' \in \{0, 1, \dots, t\}$ units in the ith battlefield, and the applied cost in Equation (27) is equal to $c_{(i-1)(m+1)+t'+1}$. Thus,

$$d[i, t] = \min_{0 \le t' \le t} \{d[i-1, t-t'] + c_{(i-1)(m+1)+t'+1}\}.$$

To compute the optimal pure strategy $x = (x_1, x_2, \ldots, x_k)$ we also keep a value

$$r[i,t] = \operatorname{argmin}_{0 \le t' \le t}\{d[i-1, t-t'] + c_{(i-1)(m+1)+t'+1}\},$$

which determines the number of units the optimal strategy should put in the ith battlefield to minimize $c_0 + \sum_{i'=1}^{i(t+1)} c_{i'}\hat{x}_{i'}$. Assuming we have correctly computed $x_{i+1}, \ldots, x_k$, in line 16, algorithm FindBestPure correctly computes x_i, which is equal to $r[i, m - \sum_{j=i+1}^{k} x_j]$. Because $x_k = r[k, m]$ we can conclude algorithm FindBestPure correctly computes the optimal strategy $x = (x_1, x_2, \ldots, x_k)$. □

7.1. Hyperplane Separating Oracle

Algorithm 2 (HyperplaneOracle) gets a hyperplane as input and either finds a point in I_A that violates constraints in LP 5 or reports that all points in I_A are satisfying all constraints in LP 5. We suppose that the input hyperplane is described by the equation

$$\alpha_0 + \alpha_1\hat{x}_1 + \ldots + \alpha_{k(a+1)}\hat{x}_{k(a+1)} = 0, \tag{28}$$

and we want to find a point $\hat{x} \in I_A$ that violates the constraint

$$\alpha_0 + \sum_{i=1}^{n} \alpha_i\hat{x}_i \ge 0. \tag{29}$$

This problem is equivalent to finding a point $\hat{x}^{\min} \in I_A$ that minimizes equation $\alpha_0 + \sum_{i=1}^{n} \alpha_i\hat{x}_i$. If $\alpha_0 + \sum_{i=1}^{n} \alpha_i\hat{x}_i^{\min} \ge 0$, it means all points in I_A are satisfying the constraints of LP 5, and otherwise $\hat{x}^{\min}$ is a point that violates constraint (6). Because points in I_A are equivalent to pure strategies of player A, we can use algorithm FindBestPure to find $\hat{x}^{\min}$. Thus we can conclude algorithm HyperplaneOracle correctly finds a violated constraint or reports (by returning "pass") that the hyperplane satisfies constraints 6 of LP 5.

Algorithm 2 (HyperplaneOracle)

```
input: $a$, $k$, $\alpha_0, \alpha_1, \ldots, \alpha_{k(a+1)}$
1: $x^{\min} \leftarrow$ FindBestPure$(a, k, \alpha_0, \alpha_1, \ldots, \alpha_{k(a+1)})$
2: $\hat{x}^{\min} \leftarrow \mathcal{G}_A(x^{\min})$
3: if $\alpha_0 + \sum_{i=1}^{k(a+1)} \alpha_i\hat{x}_i^{\min} \ge 0$ then
4:     return pass
5: else
6:     return $\alpha_0 + \sum_{i=1}^{k(a+1)} \alpha_i\hat{x}_i^{\min} < 0$
7: end if
```

7.2. Best-Response Separating Oracle

Algorithm 3 (BestRespOracle) gets a pair $(\hat{\mathbf{x}}, U)$ as input and decides whether there is a pure strategy $y = (y_1, y_2, \ldots, y_k) \in \mathcal{Y}$ such that for $\hat{y} = \mathcal{G}_B(y)$, we have

$$\sum_{i=1}^{k}\sum_{t_a=0}^{a}\sum_{t_b=0}^{b} \hat{\mathbf{x}}_{i,t_a}\hat{y}_{i,t_b}u_i^A(t_a, t_b) < U. \tag{30}$$

We can rewrite inequality (30) as follows:

$$\sum_{i=1}^{k}\sum_{t_b=0}^{b} \hat{y}_{i,t_b}\sum_{t_a=0}^{a} \hat{\mathbf{x}}_{i,t_a}u_i^A(t_a, t_b) < U.$$

Therefore, by letting $c_{i,t_b} = \sum_{t_a=0}^{a} \hat{\mathbf{x}}_{i,t_a}u_i^A(t_a, t_b)$, this problem is equivalent to finding a point $\hat{y}^{\min} \in I_B$ that minimizes $\sum_{i'=1}^{k(b+1)} c_{i'}\hat{y}_{i'}$. If $\sum_{i'=1}^{k(b+1)} c_{i'}\hat{y}_{i'}^{\min} < U$, we have found a violating payoff constraint of LP 2, and if $\sum_{i'=1}^{k(b+1)} \cdot c_{i'}\hat{y}_{i'}^{\min} \ge U$, pair $(\hat{\mathbf{x}}, U)$ satisfies all the payoff constraints of LP 2. Thus, by Lemma 19 we conclude that algorithm BestRespOracle correctly finds a violating payoff constraint of LP 2 or reports that $(\hat{\mathbf{x}}, U)$ satisfies all the payoff constrains.

Algorithm 3. BestRespOracle

```
input: $a$, $b$, $k$, $U$, $\hat{x}_1, \ldots, \hat{x}_{k(a+1)}$
 1: for $i \leftarrow 1$ to $k$ do
 2:     for $t_b \leftarrow 0$ to $b$ do
 3:         $c_{(i-1)(b+1)+t_b+1} = c_{i,t_b} = \sum_{t_a=0}^{a} \hat{x}_{i,t_a}\ u_i^A(t_a, t_b)$
 4:     end for
 5: end for
 6: $y^{\min} \leftarrow$ FindBestPure$(b, k, c_0, c_1, c_2, \ldots, c_{k(b+1)})$
 7: $\hat{y}^{\min} \leftarrow \mathcal{G}_B(y^{\min})$
 8: if $\sum_{i=1}^{k(b+1)} c_i \hat{y}_i^{\min} \geq U$ then
 9:     return pass
10: else
11:     return $\sum_{i=1}^{k(b+1)} c_i \hat{y}_i^{\min} < U$
12: end if
```

Acknowledgments

An earlier version of this paper was presented at an Association for the Advancement of Artificial Intelligence conference (http://dl.acm.org/citation.cfm?id=3015869).

Endnotes

[1] Or, equivalently, constant-sum games.

[2] A more appropriate name for equilibrium in zero-sum games may be *saddle-point equilibrium*, but for the sake of consistency with the majority of Colonel Blotto literature we may use the term *Nash equilibrium*.

[3] Independently and in parallel with an earlier version of this work, Xu et al. [49] implicitly used a similar idea to solve a class of Stackelberg security games.

[4] Note that if we were to relax this assumption and allow $u_i^A(x_i, y_i)$ to not necessarily equal $-u_i^B(x_i, y_i)$, then a special case of this game with two battlefields can model an arbitrary two-person normal-form game, and thus finding a Nash Equilibrium would be PPAD-complete.

[5] The same procedure finds an optimal strategy of player B.

[6] Note that sign(w) is 1, −1, and 0 if w is positive, negative, and zero respectively.

[7] $sign(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$

[8] Note that the sign function is a special case of bounded distance functions.

[9] This lemma claims that the strategy of a player in the finite General Lotto game can be written as a probability distribution over paired strategies. Thus $T = \beta_1\sigma_1 + \beta_2\sigma_2 + \ldots + \beta_m\sigma_m$ describes a strategy in which the paired strategy σ_i is played with probability β_i.

References

[1] Ashlagi I, Krysta P, Tennenholtz M (2008) Social context games. Saberi A, ed. *Internet and Network Economics: 6th Internat. Workshop, WINE 2010, Stanford CA, Decmeber 13–17* (Springer, New York), 675–683.
[2] Azar Y, Cohen E, Fiat A, Kaplan H, Racke H (2003) Optimal oblivious routing in polynomial time. *Proc. 35th Annual ACM Symp. Theory Comput.* (ACM, New York), 383–388.
[3] Bell RM, Cover TM (1980) Competitive optimality of logarithmic investment. *Math. Oper. Res.* 5(2):161–166.
[4] Bellman R (1969) On Colonel Blotto and analogous games. *SIAM Rev.* 11(1):66–68.
[5] Blackett DW (1954) Some Blotto games. *Naval Res. Logist. Quart.* 1(1):55–60.
[6] Blackett DW (1958) Pure strategy solutions to blotto games. *Naval Res. Logist. Quart.* 5(2):107–109.
[7] Borel É (1921) La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes Rendus de l'Académie* 173(13041308):97–100.
[8] Borel É (1953) The theory of play and integral equations with skew symmetric kernels. *Econometrica* 21(1):97–100.
[9] Brandt F, Fischer F, Harrenstein P, Shoham Y (2009) Ranking games. *Artificial Intelligence* 173(2):221–239.
[10] Charnes A (1953) Constrained games and linear programming. *Proc. Natl. Acad. Sci. USA* 39(7):639–641.
[11] Chen X, Deng X (2006) Settling the complexity of two-player Nash equilibrium. *47th Annual IEEE Symp. Foundations Comput. Sci. FOCS'06* (IEEE, New York), 261–272.
[12] Chen X, Deng X, Teng SH (2006) Computing nash equilibria: Approximation and smoothed complexity. *47th Annual IEEE Symp. Foundations Comput. Sci. FOCS'06* (IEEE, New York), 603–612.
[13] Chowdhury SM, Kovenock D, Sheremeta RM (2013) An experimental investigation of Colonel Blotto games. *Econom. Theory* 52(3):1–29.
[14] Dantzig GB (1963) *Linear Programming and Extensions* (Princeton University Press, Princeton, NJ).
[15] Daskalakis C, Goldberg PW, Papadimitriou CH (2009) The complexity of computing a Nash equilibrium. *SIAM J. Comput.* 39(1):195–259.
[16] Dziubiński M (2011) Non-symmetric discrete general lotto games. *Internat. J. Game Theory* 42(4):801–833.
[17] Fréchet M (1953) Commentary on the three notes of Emile Borel. *Econometrica* 21(1):118–124.
[18] Fréchet M (1953) Emile Borel, initiator of the theory of psychological games and its application. *Econometrica* 21(1):95–96.

[19] Garg J, Jiang AX, Mehta R (2011) Bilinear games: Polynomial time algorithms for rank based subclasses. Saberi A, ed. *Internet and Network Economics: 6th Internat. Workshop, WINE 2010, Stanford CA, Decmeber 13–17* (Springer, New York), 399–407.
[20] Goldberg PW, Papadimitriou CH (2006) Reducibility among equilibrium problems. *Proc. 38th Annual ACM Sympos. Theory Comput.* (ACM, New York), 61–70.
[21] Golman R, Page SE (2009) General Blotto: Games of allocative strategic mismatch. *Public Choice* 138(3-4):279–299.
[22] Grötschel M, Lovász L, Schrijver A (1981) The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1(2): 169–197.
[23] Grötschel M, Lovász L, Schrijver A (2012) *Geometric Algorithms and Combinatorial Optimization*, vol. 2 (Springer Science & Business Media, New York).
[24] Hart S (2008) Discrete Colonel Blotto and General Lotto games. *Internat. J. Game Theory* 36(3–4):441–460.
[25] Immorlica N, Kalai AT, Lucier B, Moitra A, Postlewaite A, Tennenholtz M (2011) Dueling algorithms. *Proc. 43rd Annual ACM Symp. Theory Comput.* (ACM, New York), 215–224.
[26] Jiang AX, Leyton-Brown K (2015) Polynomial-time computation of exact correlated equilibrium in compact games. *Games Econom. Behav.* 91(May):347–359.
[27] Khachiyan LG (1980) Polynomial algorithms in linear programming. *USSR Comput. Math. Math. Phys.* 20(1):53–72.
[28] Koller D, Megiddo N, Von Stengel B (1994) Fast algorithms for finding randomized strategies in game trees. *Proc. 26th Annual ACM Symp. Theory Comput.* (ACM, New York), 750–759.
[29] Kontogiannis S, Spirakis P (2010) Exploiting concavity in bimatrix games: New polynomially tractable subclasses. Serna M, Shaltiel R, Jansen K, Rolim J, eds. *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques* (Springer, New York), 312–325.
[30] Kovenock D, Roberson B (2010) Conflicts with multiple battlefields. CESifo Working Paper Series 3165, CESifo Group Munich, Munich, Germany.
[31] Kovenock D, Roberson B (2012) Coalitional Colonel Blotto games with application to the economics of alliances. *J. Public Econom. Theory* 14(4):653–676.
[32] Kvasov D (2007) Contests with limited resources. *J. Econom. Theory* 136(1):738–748.
[33] Laslier JF, Picard N (2002) Distributive politics and electoral competition. *J. Econom. Theory* 103(1):106–130.
[34] Letchford J, Conitzer V (2013) Solving security games on graphs via marginal probabilities. *Proc. 27th AAAI Conf. Artificial Intelligence* (AAAI Press, Palo Alto, CA), 591–597.
[35] Lipton RJ, Markakis E, Mehta A (2003) Playing large games using simple strategies. *Proc. 4th ACM Conf. Electronic Commerce* (ACM, New York), 36–41.
[36] Merolla J, Munger M, Tofias M (2005) In play: A commentary on strategies in the 2004 us presidential election. *Public Choice* 123(1–2):19–37.
[37] Myerson RB (1993) Incentives to cultivate favored minorities under alternative electoral systems. *Amer. Political Sci. Rev.* 87(4):856–869.
[38] Nash J (1951) Non-cooperative games. *Ann. Math.* 54(2):286–295.
[39] Osborne MJ, Rubinstein A (1994) *A Course in Game Theory* (MIT Press, Cambridge, MA).
[40] Papadimitriou CH, Steiglitz K (1998) *Combinatorial Optimization: Algorithms and Complexity* (Courier Dover Publications, Mineola, NY).
[41] Roberson B (2006) The Colonel Blotto game. *Econ. Theory* 29(1):1–24.
[42] Rothvoss T (2014) The matching polytope has exponential extension complexity. *Proc. 46th Annual ACM Symp. Theory Comput.* (ACM, New York), 263–272.
[43] Sahuguet N, Persico N (2006) Campaign spending regulation in a model of redistributive politics. *J. Econom. Theory* 28(1):95–124.
[44] Shubik M, Weber RJ (1981) Systems defense games: Colonel Blotto, command and control. *Naval Res. Logist. Quart.* 28(2):281–287.
[45] Sion M (1958) On general minimax theorems. *Pacific J. Math.* 8(1):171–176.
[46] Tukey JW (1949) A problem of strategy. *Econometrica* 17:73.
[47] Von Neumann J, Fréchet M (1953) Communication on the Borel notes. *Econometrica* 21(1):124–127.
[48] Weinstein J (2012) Two notes on the Blotto game. *BE J. Theoret. Econom.* 12(1):1–13.
[49] Xu H, Fang F, Jiang AX, Conitzer V, Dughmi S, Tambe M (2014) Solving zero-sum security games in discretized spatio-temporal domains. *Proc. 28th Conf. Artificial Intelligence* (AAAI, Palo Alto, CA), 1500–1506.