Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

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

AmirMahdi Ahmadinejad,[*] Sina Dehghani,[†] MohammadTaghi Hajiaghayi,[†]
Brendan Lucier,[‡] Hamid Mahini,[†], Saeed Seddighin[†]
[*]Stanford University [†]Univeristy of Maryland, [‡]Microsoft Research
amirmahdi.ahmadi@gmail.com, brlucier@microsoft.com, {dehghani, hajiagha, hmahini, sseddigh}@umd.edu

Abstract

We study the problem of computing Nash equilibria of zero-sum games. Many natural zero-sum games have exponentially many strategies, but highly structured payoffs. For example, in the well-studied Colonel Blotto game (introduced by Borel in 1921), players must divide a pool of troops among a set of battlefields with the goal of winning (i.e., having more troops in) a majority. The Colonel Blotto game is commonly used for analyzing a wide range of applications from the U.S presidential election, to innovative technology competitions, to advertisement, to sports. However, because of the size of the strategy space, standard methods for computing equilibria of zero-sum games fail to be computationally feasible. Indeed, despite its importance, only few solutions for special variants of the problem are known.

In this paper we show how to compute equilibria of Colonel Blotto games. Moreover, our approach takes the form of a general reduction: to find a Nash equilibrium of a zero-sum game, it suffices to design a separation oracle for the strategy polytope of any bilinear game that is payoff-equivalent. We then apply this technique to obtain the first polytime algorithms for a variety of games. In addition to Colonel Blotto, we also show how to compute equilibria in an infinite-strategy variant called the General Lotto game; this involves showing how to prune the strategy space to a finite subset before applying our reduction. We also consider the class of dueling games, first introduced by Immorlica et al. (2011). We show that our approach provably extends the class of dueling games for which equilibria can be computed: we introduce a new dueling game, the matching duel, on which prior methods fail to be computationally feasible but upon which our reduction can be applied.

Introduction

Computing a Nash equilibrium 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) (Nash 1951). But it is not necessarily obvious how to find an equilibrium. Indeed, the conclusions to date have been largely negative: computing a Nash equilibrium of a normal-form game is known to be PPAD-complete (Daskalakis, Goldberg, and Papadimitriou 2009; Goldberg and Papadimitriou 2006), even for two-player games (Chen and Deng 2006). In fact, it is PPAD-complete to find an $\frac{1}{n^{O(1)}}$ approximation to a Nash equilibrium (Chen, Deng, and Teng 2006). These results call into question the predictiveness of 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 structure that can be exploited to admit computational results (Dantzig 1963; Garg, Jiang, and Mehta 2011; Kontogiannis and Spirakis 2010; Lipton, Markakis, and Mehta 2003; Alon et al. 2013; Demaine et al. 2007; 2009; Ahmadinejad et al. 2015). Perhaps the most well-known example is the class of zero-sum two-player games[1], where player 2's payoff is the negation of player 1's payoff. The normal-form representation of a zero-sum game is a matrix A, which specifies the game payoffs for player 1. This is a very natural class of games, as it models perfect competition between two parties. Given the payoff matrix for a zero-sum game as input, a Nash equilibrium can be computed in polynomial time, and hence time polynomial in the number of pure strategies available to each player (Dantzig 1963). 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.

The Colonel Blotto Game A classical and important example illustrating these issues is the Colonel Blotto game, first introduced by Borel in 1921 (Borel 1921; 1953; Fréchet 1953a; 1953b; von Neumann 1953). 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

[*]Supported in part by NSF CAREER award 1053605, NSF grant CCF-1161626, ONR YIP award N000141110662, DARPA/AFOSR grant FA9550-12-1-0423, and a Google faculty research award.
Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

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

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 different forms of competition: from sports, to advertisement, to politics (Myerson 1993; Kovenock and Roberson 2010; 2012; Roberson and Kvasov 2012; Bachrach, Syrgkanis, and Vojnovic 2011; Polevoy, Trajanovski, and de Weerdt 2014), 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. There are $\binom{n+k-1}{k-1}$ ways to partition n troops among k battlefields. The classical methods for computing the equilibra 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 (Tukey 1949; Kvasov 2007; Hart 2007; Golman and Page 2009; Kovenock and Roberson 2012). Despite this effort, progress remains sparse. Much of the existing work considers a continuous relaxation of the problem where troops are divisible, and for this relaxation a significant breakthrough came only quite recently in the seminal work of Roberson (Roberson 2006), 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 non-relaxed 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.

An Approach: Bilinear Games How should one approach equilibrium computation in such a game? 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, Megiddo and von Stengel (Koller, Megiddo, and Von Stengel 1994) show how to compute equilibria for zero-sum extensive-form games with perfect recall. Immorlica et al. (Immorlica et al. 2011) give an approach for solving algorithmically-motivated "dueling games" with uncertainty. Letchford and Conitzer (Letchford and Conitzer 2013) compute equilibria for a variety of graphical security games. Each of these cases involve games with exponentially many strategies. In each case, a similar approach is employed: 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 (Charnes 1953; Koller, Megiddo, and Von Stengel 1994). 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. If one can also map a strategy in the bilinear game back to the original game, then one has a polytime 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.

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.

The reduction described above follows from a repeated application of the classic equivalence of separation and optimization (Grötschel, Lovász, and Schrijver 1981). In more detail, we formulate the equilibrium conditions as an 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 via 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 (Grötschel, Lovász, and Schrijver 1981). 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 (Grötschel, Lovász, and Schrijver 1981).

The reduction described above is not specific to the Colonel Blotto game: it applies to any zero-sum game, and any payoff-equivalent bilinear form thereof. 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[2] and similar ideas have been used to compute correlated equilibria in compact games (Jiang and Leyton-Brown 2015). In particular, it is notable that one requires only a single linear optimization oracle, over the set of pure strategies, to both find an equilibrium of the bilinear game and convert this to a mixed equilibrium in the original game. 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 polytime 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.

Dueling Games In a *dueling game*, introduced by Immorlica et al. (Immorlica et al. 2011), 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. This framework falls within a natural class of ranking or social context games (Ashlagi, Krysta, and Tennenholtz 2008; Brandt et al. 2009), 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 who must 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. (Immorlica et al. 2011) 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, since the matching polytope does not have a compact representation (Rothvoss 2014), the original method of (Immorlica et al. 2011) 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.

General Lotto Game Hart (Hart 2007) considers a variant of the Colonel Blotto game, namely the *General Lotto* game. In this game, each player chooses a distribution over non-negative 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; the players' payoffs are then 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 well studied in the literature (see, for example, (Bell and Cover 1980; Sahuguet and Persico 2006; Hart 2007; Dziubiński 2011)).

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 by considering the Colonel Blotto game as a specific example. In the following section we describe our approach in detail for the Colonel Blotto game, explaining the process by which equilibria can be computed. Then we will present the general reduction. Further applications of this technique are provided in the full version of the paper (for the General Lotto game).

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 two players have different number 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 i-th battlefield, respectively. Note that the payoff functions of the i-th battlefield, u_i^A and u_i^B, have $(a+1) \times (b+1)$ entries. This means the size of input is $\Theta(kab)$. Since Colonel Blotto is a zero-sum game, we have $u_i^A(x_i, y_i) = -u_i^B(x_i, y_i)$[3]. Note that we do not need to put any constraint on the payoff functions, and our result works for all payoff functions. 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 $h_{\mathcal{B}}^B(x, y) = \sum_i u_i^B(x_i, y_i)$,

[2]Independently and in parallel with an earlier version of this work, (Xu et al. 2014) implicitly used a similar idea to solve a class of Stackleberg security games.

[3]Note that in the Colonel Blotto game if $u_i^A(x_i, y_i)$ is not necessarily equal to $-u_i^B(x_i, y_i)$ then a special case of this game with two battlefields can model an arbitrary 2-person normal-form game and thus finding a Nash Equilibrium would be PPAD-complete.

respectively. A mixed strategy of each player would be a probability distribution over his pure 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, i.e., 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 represent a mixed strategy of player A with function $p : \mathcal{X} \to [0,1]$ such that $\sum_{x \in \mathcal{X}} p(x) = 1$. Similarly, let function $q : \mathcal{Y} \to [0,1]$ be a mixed strategy of player B. We may also use $\mathbf{x}$ and $\mathbf{y}$, instead of p and q, for referring to a mixed strategy of player A and B respectively. Since 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 (Sion 1957). 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^*. For each mixed strategy p, at least one of the best-response strategies of p is a pure strategy. Therefore, a solution to the following program characterizes strategy p^*.

$$\begin{array}{lll} \max & U & (1) \\ s.t. & \sum_{x \in \mathcal{X}} p_x = 1, & \\ & \sum_{x \in \mathcal{X}} p_x h^A_{\mathcal{B}}(x, y) \geq U, \quad \forall y \in \mathcal{Y}, & \end{array}$$

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.

Step 1: Transferring to a new space. We address this issue by transforming the solution space to a new space in which an LP equivalent to LP 1 becomes tractable (See, e.g., (Azar et al. 2003), for similar technique). 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 $\hat{\mathbf{x}} \in \{0,1\}^{n(A)}$ 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 $\mathcal{G}_A(x) = \hat{\mathbf{x}} \in \{0,1\}^{n(A)}$ 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 i-th battlefield then $\hat{x}_{i,j} = 1$. Let $I_A = \{\hat{\mathbf{x}} \in \{0,1\}^{n(A)} | \exists x \in \mathcal{X}, \mathcal{G}_A(x) = \hat{\mathbf{x}}\}$ be the set of points in $\{0,1\}^{n(A)}$ which represent pure strategies of player A. Let $\mathcal{M}(\mathcal{X})$ and $\mathcal{M}(\mathcal{Y})$ be the set of mixed strategies of players A and B, respectively. Similarly, we map mixed strategy $\mathbf{x}$ to point $\mathcal{G}_A(\mathbf{x}) = \hat{\mathbf{x}} \in [0,1]^{n(A)}$ such that $\hat{x}_{i,j}$ represents the probability that mixed strategy $\mathbf{x}$ puts j troops in the i-th battlefield. Note that mapping $\mathcal{G}_A$ is not necessarily one-to-one nor onto, i.e., each point in $[0,1]^{n(A)}$ may be mapped to zero, one, or more than one strategies. 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 $[0,1]^{n(A)}$ which 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 $I_B = \{\hat{\mathbf{y}} \in \{0,1\}^{n(B)} | \exists y \in \mathcal{Y}, \mathcal{G}_B(y) = \hat{\mathbf{y}}\}$ and $S_B = \{\hat{\mathbf{y}} \in [0,1]^{n(B)} | \exists \mathbf{y} \in \mathcal{M}(\mathcal{Y}), \mathcal{G}_B(\mathbf{y}) = \hat{\mathbf{y}}\}$.

Lemma 0.1 *The set S_A forms a convex polyhedron with an exponential number of vertices and facets.*

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

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

where

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

is the expected payoff of player A.

Step 2: Solving LP 2. The modified LP above, 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 (Grötschel, Lovász, and Schrijver 1988; Papadimitriou and Steiglitz 1998). By the equivalence of separation and optimization (Grötschel, Lovász, and Schrijver 1981), 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, 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 $\hat{\mathbf{x}} = \mathcal{G}(x)$ minimizes the following equation:

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

and similarly for polytope $\mathcal{Y}$. The following lemma shows that one can indeed find a minimizer of Equation (3) in polynomial time.

Lemma 0.2 *Given two integers m and k and a sequence $c_0, c_1, \ldots, c_{k(m+1)}$, one can find (in polynomial time) an optimal pure strategy $x = (x_1, x_2, \ldots, x_k)$ where $\sum_{i=1}^{k} x_i = m$, $\hat{\mathbf{x}} = \mathcal{G}(x)$ and $\hat{\mathbf{x}}$ minimizes $c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i$.*

Proof: We employ a dynamic programming approach. Define $d[i,t]$ as 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 $c_0 + \sum_{i=1}^{k(m+1)} c_i \hat{x}_i$. We have that $d[0,j]$ is equal to c_0 for all j. For an arbitrary $i > 0$ and t, the optimal strategy x puts $0 \leq t' \leq t$ units in the i-th battlefield and the applied cost in the equation 3 is equal to $c_{(i-1)(m+1)+t'+1}$. Thus, we can express $d[i,t]$ as

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

Solving this dynamic program, we can find the allocation that minimizes $\sum \alpha_i \hat{\mathbf{x}}_i$ in polynomial time, as required. ■

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 (Grötschel, Lovász, and Schrijver 1981) 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 $\hat{\mathbf{x}} = \sum_i \alpha_i \hat{\mathbf{x}}_i$. Since each $\hat{\mathbf{x}}_i$ corresponds to a pure strategy in $\mathcal{X}$, it is trivial to find point $\mathbf{x}_i$ with $\mathcal{G}_A(\mathbf{x}_i) = \hat{\mathbf{x}}_i$, since the marginals of each $\hat{\mathbf{x}}_i$ lie in $\{0, 1\}$. We then have that $\mathbf{x} = \sum_i \alpha_i \mathbf{x}_i$ is the required mixed strategy profile.

Combining these three steps, we find a Nash Equilibrium of the Colonel Blotto game in polynomial time. ■

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, where 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 x of player A to $\mathcal{G}_A(x) = \hat{\mathbf{x}} \in R^{n(A)}$, and each strategy y of player B to $\mathcal{G}_B(y) = \hat{\mathbf{y}} \in R^{n(B)}$ such that the payoff of strategies $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ can be represented in a bilinear form based on $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$.
2. For any given vector α and real number α_0 we can find, in polynomial time, whether there is a pure strategy $\hat{\mathbf{x}}$ 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 polytime algorithm which finds a Nash Equilibrium of a given polynomially separable game.*

This general methodology can be used for finding a 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, known as General Lotto games, and for a class of dueling games introduced by Immorlica et al. (Immorlica et al. 2011).

We also show one can use similar techniques to compute the approximate equilibrium payoffs of a dueling game when we are not able to answer the separation problem in polynomial time but instead we can polynomially solve the ϵ-separation problem for any $\epsilon > 0$.

Theorem 3 *Given an oracle function for the ϵ-separation problem, one can find an ϵ-approximation to the equilibrium payoffs of a polynomially separable game in polynomial time.*

General Lotto

The General Lotto game is a relaxation of the Colonel Lotto game (See (Hart 2007) 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

$$h_\Gamma^A(X, Y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \Pr(X = i) \Pr(Y = j) u(i, j), \qquad (4)$$

and again the payoff of player B is the negative of the payoff of player A, i.e., $h_\Gamma^B(X, Y) = -h_\Gamma^A(X, Y)$. Hart (Hart 2007) presents a solution for the General Lotto game when $u(i, j) = \text{sign}(i - j)$. 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 $u^T \in O(\text{poly}(a, b))$. Note that $u(i, j) = \text{sign}(i - j)$ is a bounded distance function where it reaches its maximum value at $i - j = 1$. Now, we are ready to present our main result regarding the General Lotto game.

Theorem 4 *There is a polynomial-time algorithm which finds an equilibrium of a General Lotto game where the payoff function is a bounded distance function.*

Main challenge Note that in the General Lotto game, each player has an infinite number of pure strategies, and thus one cannot use either our proposed algorithm for the Colonel Blotto game or the technique of (Immorlica et al. 2011) for solving the problem. We should prune strategies such that the problem becomes tractable. Therefore, we 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 algorithm of this kind which computes an NE of a game with infinite number of pure strategies.

Application to Dueling Games

Immorlica et al. (Immorlica et al. 2011) introduced the class of dueling games. In these games, an optimization problem with an element of uncertainty is considered as 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, we describe our method and show that our technique solves a more general class of dueling games. Furthermore, we provide examples to show how our method can be a simpler tool for solving bilinear duel games compared to (Immorlica et al. 2011) method. Finally, in Section

, we examine the matching duel game to provide an example where the method of (Immorlica et al. 2011) does not work, but our presented method can yet be applied.

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 based on 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. In the following there are two dueling games mentioned in (Immorlica et al. 2011).

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.

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. Hence, set of strategies X is all $m!$ permutations of the pages and for permutation $x = (x_1, x_2, \dots, x_m)$ and page ω, $c(x, \omega) = i$ iff $\omega = x_i$.

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 in S_A is a mixed strategy of player A. Similarly define strategy $\hat{y}$, $n(B)$, and S_B for player B. A dueling game is *bilinear* if utility function $h^A(\hat{x}, \hat{y})$ has the form $\hat{x}^t M \hat{y}$ where M is an $n(A) \times n(B)$ matrix. Again for player B, we have $h^B(\hat{x}, \hat{y}) = -h^A(\hat{x}, \hat{y})$. Immorlica et al. (Immorlica et al. 2011) 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 present the convex hull of strategies S_A and S_B with m polynomial linear constraints, i.e. there are m vectors $\{v_1, v_2, \dots, v_m\}$ and m real numbers $\{b_1, b_2, \dots, b_m\}$ such that $S_A = \{\hat{x} \in R^{n(A)} | \forall i \in \{1, 2, \dots, m\}, v_i.\hat{x} \geq b_i\}$. Similarly $S_B = \{\hat{y} \in R^{n(B)} | \forall i \in \{1, 2, \dots, m'\}, v'_i.\hat{y} \geq b'_i\}$.*

In the following theorem, we show that every polynomially representable bilinear duel is also polynomially separable. This implies we can use the general reduction to solve polynomially representable bilinear dueling games as well.

Theorem 5 *Every polynomially-representable bilinear dueling game is polynomially separable.*

Matching duel

In a matching duel we are given a weighted graph $G = (V, E, W)$ which is not necessarily bipartite. In a matching duel each pure strategy of players is a perfect matching, set of possible situations Ω is the same as the set of nodes in G, and probability distribution p over Ω determines the probability of selection of each node. In this game, strategy x beats 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 website that suggest a better match for each user will get that user, and the goal of each website is to maximize the number of its users. Note that ranking duel is a special case of the matching duel when G is a $K_{n,n}$ graph, in which one part denotes the web pages and the other part denotes the ranks. Thus, the weight of the edge between page i and rank j is equal to j.

First, we describe how our method can solve this game and then we show the method of Immorlica et al. (Immorlica et al. 2011) cannot be applied to find an NE of the matching duel.

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

Note that Rothvoss (Rothvoss 2014) showed that the feasible strategy polytope (the perfect matching polytope) has exponentially many facets. Therefore, the prior approach represented in the work of Immorlica et al. (Immorlica et al. 2011) is not applicable to the matching duel. This example shows that our framework nontrivially generalizes the method of Immorlica et al. (Immorlica et al. 2011) and completes the presentation of our simpler and more powerful tool for solving bilinear duels.

References

Ahmadinejad, A.; Dehghani, S.; Hajiaghayi, M.; Mahini, H.; Seddighin, S.; and Yazdanbod, S. 2015. Forming external behaviors by leveraging internal opinions. In *INFOCOM*, 1849–1857. IEEE.

Alon, N.; Demaine, E. D.; Hajiaghayi, M. T.; and Leighton, T. 2013. Basic network creation games. *SIAM Journal on Discrete Mathematics* 27(2):656–668.

Ashlagi, I.; Krysta, P.; and Tennenholtz, M. 2008. Social context games. In *WINE*. Springer. 675–683.

Azar, Y.; Cohen, E.; Fiat, A.; Kaplan, H.; and Racke, H. 2003. Optimal oblivious routing in polynomial time. In *STOC*, 383–388.

Bachrach, Y.; Syrgkanis, V.; and Vojnovic, M. 2011. Efficiency and the redistribution of welfare. Technical report, Technical report, Microsoft Research.

Bell, R. M., and Cover, T. M. 1980. Competitive optimality of logarithmic investment. *Math. Oper. Res.* 5(2):161–166.

Borel, É. 1921. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes Rendus de l'Académie* 173(13041308):97–100.

Borel, É. 1953. The theory of play and integral equations with skew symmetric kernels. *Econometrica* 21:97–100.

Brandt, F.; Fischer, F.; Harrenstein, P.; and Shoham, Y. 2009. Ranking games. *Artificial Intelligence* 173(2):221–239.

Charnes, A. 1953. Constrained games and linear programming. *Proceedings of the National Academy of Sciences of the United States of America* 39(7):639.

Chen, X., and Deng, X. 2006. Settling the complexity of two-player nash equilibrium. In *FOCS*, volume 6, 47th.

Chen, X.; Deng, X.; and Teng, S.-H. 2006. Computing nash equilibria: Approximation and smoothed complexity. In *FOCS*, 603–612. IEEE.

Dantzig, G. B. 1963. *Linear programming and extensions*. Princeton university press.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The complexity of computing a nash equilibrium. *SIAM Journal on Computing* 39(1):195–259.

Demaine, E. D.; Hajiaghayi, M.; Mahini, H.; and Zadimoghaddam, M. 2007. The price of anarchy in network creation games. In *PODC*, 292–298. ACM.

Demaine, E. D.; Hajiaghayi, M.; Mahini, H.; and Zadimoghaddam, M. 2009. The price of anarchy in cooperative network creation games. *ACM SIGecom Exchanges* 8(2):2.

Dziubiński, M. 2011. Non-symmetric discrete general lotto games. *Int. J. Game Theory* 1–33.

Fréchet, M. 1953a. Commentary on the three notes of emile borel. *Econometrica* 21:118–124.

Fréchet, M. 1953b. Emile borel, initiator of the theory of psychological games and its application. *Econometrica* 21:95–96.

Garg, J.; Jiang, A. X.; and Mehta, R. 2011. Bilinear games: Polynomial time algorithms for rank based subclasses. In *WINE*. Springer. 399–407.

Goldberg, P. W., and Papadimitriou, C. H. 2006. Reducibility among equilibrium problems. In *STOC*, 61–70. ACM.

Golman, R., and Page, S. E. 2009. General blotto: games of allocative strategic mismatch. *Public Choice* 138(3-4):279–299.

Grötschel, M.; Lovász, L.; and Schrijver, A. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1(2):169–197.

Grötschel, M.; Lovász, L.; and Schrijver, A. 1988. *Geometric algorithms and combinatorial optimization*. Springer4060 XII, 362 S.

Hart, S. 2007. Discrete colonel blotto and general lotto games.

Immorlica, N.; Kalai, A. T.; Lucier, B.; Moitra, A.; Postlewaite, A.; and Tennenholtz, M. 2011. Dueling algorithms. In *STOC*, 215–224.

Jiang, A. X., and Leyton-Brown, K. 2015. Polynomial-time computation of exact correlated equilibrium in compact games. *Games and Economic Behavior* 91(0):347 – 359.

Koller, D.; Megiddo, N.; and Von Stengel, B. 1994. Fast algorithms for finding randomized strategies in game trees. In *STOC*, 750–759. ACM.

Kontogiannis, S., and Spirakis, P. 2010. Exploiting concavity in bimatrix games: New polynomially tractable subclasses. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer. 312–325.

Kovenock, D., and Roberson, B. 2010. Conflicts with multiple battlefields. CESifo Working Paper Series 3165, CESifo Group Munich.

Kovenock, D., and Roberson, B. 2012. Coalitional colonel blotto games with application to the economics of alliances. *J. Pub. Econ. Theory* 14(4):653–676.

Kvasov, D. 2007. Contests with limited resources. *J. Econ. Theory* 136(1):738–748.

Letchford, J., and Conitzer, V. 2013. Solving security games on graphs via marginal probabilities. In *AAAI*.

Lipton, R. J.; Markakis, E.; and Mehta, A. 2003. Playing large games using simple strategies. In *EC*, 36–41. ACM.

Myerson, R. B. 1993. Incentives to cultivate favored minorities under alternative electoral systems. *Am. Polit. Sci. Rev.* 856–869.

Nash, J. 1951. Non-cooperative games. *Annals of mathematics* 286–295.

Papadimitriou, C. H., and Steiglitz, K. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Dover Publications.

Polevoy, G.; Trajanovski, S.; and de Weerdt, M. M. 2014. Nash equilibria in shared effort games. In *AAMAS*, 861–868. International Foundation for Autonomous Agents and Multiagent Systems.

Roberson, B., and Kvasov, D. 2012. The non-constant-sum colonel blotto game. *Economic Theory* 51(2):397–433.

Roberson, B. 2006. The colonel blotto game. *Econ. Theor.* 29(1):1–24.

Rothvoss, T. 2014. The matching polytope has exponential extension complexity. In *STOC*, 263–272.

Sahuguet, N., and Persico, N. 2006. Campaign spending regulation in a model of redistributive politics. *J. Econ. Theory* 28(1):95–124.

Sion, M. 1957. *General Minimax Theorems*. United States Air Force, Office of Scientific Research.

Tukey, J. W. 1949. A problem of strategy. *Econometrica* 17:73.

von Neumann, J. 1953. Communication on the borel notes. *Econometrica* 21:124–127.

Xu, H.; Fang, F.; Jiang, A. X.; Conitzer, V.; Dughmi, S.; and Tambe, M. 2014. Solving zero-sum security games in discretized spatio-temporal domains. In *AAAI*.