Nash equilibrium Computation in Resource Allocation Games

Extended Abstract

Shivam Gupta
University of Illinois at Urbana-Champaign
Champaign, IL, USA
sgupta72@illinois.edu

Ruta Mehta
University of Illinois at Urbana-Champaign
Champaign, IL, USA
rutameht@illinois.edu

ACM Reference Format:
Shivam Gupta and Ruta Mehta. 2018. Nash equilibrium Computation in Resource Allocation Games. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 3 pages.

1 INTRODUCTION

Nash equilibrium is one of the most fundamental solution concepts within game theory. It is defined as a strategy profile in which no individual can gain by changing its strategy unilaterally. Extensive work within algorithmic game theory in the last two decades has led to a plethora of results on computation of a Nash equilibrium (NE) in various finite normal-form games [1, 5–7, 9]. The problem is PPAD-complete even for two-player games [6, 9]. Despite this, the classical result of von Neumann (1928) gave a linear programming formulation for two-player zero-sum games, where one player's gain is the other's loss [8, 12].

In this paper, we study the computation of a Nash equilibrium in the polymatrix Blotto game that is zero-sum in total. To describe how general this setting is, we first need to understand the classical two-player Blotto setting, which has been previously studied [2, 3]. In this game, both players have a certain number of soldiers that they allocate to a number of battlefields. A function $h(b, k_1, k_2)$ represents the payoff of the first player from battlefield b when the first player allocates k_1 soldiers, and the second player allocates k_2 soldiers to battlefield b. The second player's payoff is $-h(b, k_1, k_2)$. The total payoff of a player is the payoffs summed over all battlefields.

The polymatrix zero-sum Blotto game is a multi-player version of the two-player Blotto game. In the polymatrix zero-sum Blotto game, each player has a number of soldiers to distribute among battlefields. For each edge (u, v) in the network we are given two functions, h_u^v and h_v^u, for players u and v respectively, and they need not add up to zero. The only guarantee is that the total payoff of all the players is zero in any play. We show a LP formulation for the zero-sum polymatrix game [4], and combine it with the algorithm of [2] to obtain a polynomial time algorithm to compute a NE. The Blotto game is typically used to model competition between two players on multiple fronts using limited resources. Since our game need not be pair-wise zero-sum, it is capable of capturing competition between multiple teams, where a team is made of a set of players who are coordinating with each other to beat other teams.

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

2 PRELIMINARIES

In this section, we briefly discuss the normal-form polymatrix game and show a linear program for the computation of the NE when the game is zero-sum. Additionally, we give a brief description of the Blotto game.

Notations. We will use $[n]$ to denote index set $\{1, \ldots, n\}$, bold-face letter $\mathbf{x}$ to denote vectors, and $\mathbf{x}_i$ or $\mathbf{x}(i)$ to denote the i^{th} coordinate of vector $\mathbf{x}$. For matrix A, we use $A(i, j)$ to denote its entry in i^{th} row and j^{th} column. For a finite set S, $\Delta(S)$ represents all probability distributions over elements of S.

Polymatrix Games. A game played on a network, where each node is a player, and plays a two-player game with each of its neighbors, is called a polymatrix game. Let the underlying undirected graph be $G = (V, E)$. Let S_u be the set of strategies of player $u \in V$ and $\Delta_u = \Delta(S_u)$ be its set of mixed strategies. On edge $(u, v) \in E$, let the payoff matrix of node u be $\boldsymbol{A_{u,v}}$ and that of v be $\boldsymbol{A_{v,u}}$. If each player $v \in [m]$ plays strategy $\mathbf{x}_v \in \Delta_v$, we will denote the strategy profile using $\mathbf{x} = (\mathbf{x}_v)_{v \in V}$. The payoff of player $u \in V$ at $\mathbf{x}$ is $\text{Payoff}_u(\mathbf{x}) = \sum_{v:(u,v) \in E} \mathbf{x_u}^T \boldsymbol{A_{u,v}} \mathbf{x_v}$.

Again, $\mathbf{x}$ is said to be at Nash equilibrium (NE) if no player gains by unilateral deviation. Existence of a NE follows from Nash's theorem [11]. Let us define the payoff of player $u \in V$ from her pure strategy $i \in S_u$ to be $\text{Payoff}_u(i, \mathbf{x}_{-u})$, where $\mathbf{x}_{-u}$ represents the strategies of all players except u. So, $\text{Payoff}_u(i, \mathbf{x}_{-u}) = \sum_{v:(u,v) \in E} (\boldsymbol{A_{u,v}} \mathbf{x_v})_i$.

Let $\mathcal{S} = \times_{u \in V} S_u$. The polymatrix Blotto game is said to be **zero-sum** if, for every pure profile $s \in \mathcal{S}$, the sum of payoffs of all the players is zero, i.e., $\sum_{u \in V} \sum_{v \neq u} \boldsymbol{A_{u,v}}(s_u, s_v) = 0$.

Theorem 2.1. *If the polymatrix game is zero-sum, then the following linear-program gives a NE.*

$$\begin{aligned} \textit{minimize} \quad & \sum_{u \in V} w_u \\ \textit{subject to} \quad & w_u \geq \textit{Payoff}_u(i, \mathbf{x_{-u}}) \quad \forall u \in V, \forall i \in S_u \\ & \mathbf{x_u} \in \Delta_u \quad \forall u \in V \end{aligned} \tag{1}$$

Blotto game. A Blotto game consists of B battle fields, and players with a number of soldiers. In the two player case, let S_1 and S_2 be the number of soldiers of the two players. If the first and second player places j and k soldiers on battlefield $b \in [B]$, then they receive payoff $h_1(b, j, k)$ and $h_2(b, j, k)$ respectively from this battlefield. Each player $i = 1, 2$ distributes her S_i soldiers among B battlefields, and her payoff is the sum of the payoffs from each battlefield. Clearly, the number of pure strategies of player i is $\binom{S_i + B - 1}{B - 1}$, but the payoff representation takes at most $O(B * S_1 * S_2)$ space. Thus, each player has exponentially many strategies in the game representation.

3 ZERO-SUM POLYMATRIX BLOTTO

We will derive polynomial time algorithms to solve zero-sum polymatrix Blotto games using ideas from [2].

Let the underlying graph of the polymatrix game be $G = (V, E)$, and let $m = |V|$. There are B battlefields, and each player $u \in V$ has S_u soldiers to allocate to these battlefields. A pure strategy of player u is an integer vector $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_B) \geq 0$ that defines a partition of S_u soldiers over B battlefields, *i.e.,* $\sum_{k=1}^{B} \mathbf{x}_k = S_u$. Let X_u denote the set of pure strategies of player u. Then, clearly, $|X_u| = \binom{S_u+B-1}{B-1}$. Let $\Delta_u = \Delta(X_u)$ be the set of mixed strategies available to player u. Let $h_u^v(b, j, k)$ be the (arbitrary) payoff that u receives from playing against v on battlefield b, when u uses j soldiers and v uses k soldiers.

Since the number of pure strategies of each player is exponential, we cannot directly find a NE by using LP 1. Instead, we will map each strategy to a different strategy space. Let $n(u) = B(S_u + 1)$ be the number of marginal strategy entries available to player $u \in V$. For a binary matrix $\hat{\mathbf{x}} \in \{0, 1\}^{n(u)}$, let $\hat{\mathbf{x}}(b, j) = 1$ if and only if player u places j soldiers on the b^{th} battlefield. For $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_b) \in X_u$, this defines a mapping $G_u(\mathbf{x}) = \hat{\mathbf{x}} \in \{0, 1\}^{n(u)}$. Now, define the set $I_u = \{\hat{\mathbf{x}} \in \{0, 1\}^{n(u)} | \exists x \in X, G_u(\mathbf{x}) = \hat{\mathbf{x}}\}$. For $\mathbf{x} \in \Delta_u$, similarly define $G_u(\mathbf{x}) = \hat{\mathbf{x}} \in [0, 1]^{n(u)}$ such that $\hat{\mathbf{x}}(b, j)$ is the probability that mixed strategy $\mathbf{x}$ puts j soldiers in the b^{th} battlefield. Define the set $J_u = \{\hat{\mathbf{x}} \in [0, 1]^{n(u)} | \exists \mathbf{x} \in \Delta(X), G_u(\mathbf{x}) = \hat{\mathbf{x}}\}$. Note that I_u does not have a succinct representation and may have exponentially many strategies.

If player u plays pure strategy $\hat{\mathbf{y}}$ and player v plays mixed strategy $\hat{\mathbf{x}}_v$, then the payoff of u against v on battlefield b is $g_u^v(b, \hat{\mathbf{y}}, \hat{\mathbf{x}}_v) = \sum_{j=0}^{S_u} \sum_{k=0}^{S_v} \hat{\mathbf{y}}(b, j) h_u^v(b, j, k) \hat{\mathbf{x}}_v(b, k)$. Then, if player $u \in V$ plays pure strategy $\hat{\mathbf{y}} \in I_u$ and every other player $v \in V$ such that $v \neq u$ plays strategy $\hat{\mathbf{x}}_v$, then u receives payoff $\text{Payoff}_u(\hat{\mathbf{y}}, \hat{\mathbf{x}}_{-u}) = \sum_{v \neq u} \sum_{b=1}^{B} g_u^v(b, \hat{\mathbf{y}}, \hat{\mathbf{x}}_v)$.

3.1 Linear Program with Exponentially Many Constraints

$$\text{minimize} \quad \sum_{u \in V} w_u \tag{2}$$

$$\text{subject to} \quad \hat{\mathbf{x}}_u \in J_u \quad \forall u \in V \qquad \text{(Membership constraints)}$$

$$w_u \geq \text{Payoff}_u(\hat{\mathbf{y}}, \hat{\mathbf{x}}_{-u}) \quad \forall u \in V, \forall \hat{\mathbf{y}} \in I_u \qquad \text{(Payoff constraints)}$$

Lemma 3.1. *LP (2) computes a NE of the polymatrix Blotto game if it is zero-sum in the new space.*

Since LP (2) can find a Nash equilibrium of the game as long as the sum of payoffs of all the players is always zero, and we do not require that $h_u^v(b, j, k) = -h_v^u(b, j, k)$ as in [2]. Note that LP (2) has polynomially many variables, but the representation of J_u may require exponentially many inequalities. So, LP (2) may have exponentially many constraints. Thus, the only way to solve it is by using the ellipsoid method [10]. For this, we need to construct a polynomial-time separation oracle for the polyhedron of LP (2).

A separation oracle (SO) is a polynomial time algorithm that takes a point and either finds a hyperplane that separates the point from the feasible region, or returns that the point is in the feasible region. We construct efficient separation-oracles for both Membership and Payoff constraints individually.

SO for the Membership Constraints. For every $u \in V$, we can construct a separation oracle for the membership constraint that takes a point $\hat{\mathbf{x}}_\mathbf{u}$ as input and either finds a hyperplane that separates $\hat{\mathbf{x}}_\mathbf{u}$ from J_u, or reports that no such hyperplane exists. We need to find a hyperplane $\alpha_0 + \sum_{j=1}^{n(u)} \alpha_j x_j = 0$ such that $\hat{\mathbf{x}}_\mathbf{u}$ is on one side of the hyperplane, and all $\hat{\mathbf{y}} \in I_u$ are on the other side. This is because J_u is the convex hull of points in I_u, so if a hyperplane separates $\hat{\mathbf{x}}_u$ from all the points in I_u, then it separates $\hat{\mathbf{x}}_u$ from J_u. The following linear feasibility problem finds such a hyperplane if it exists.

$$\alpha_0 + \sum_{j=1}^{n(u)} \alpha_j \hat{\mathbf{x}}_\mathbf{u}(j) \geq 0, \quad \alpha_0 + \sum_{j=1}^{n(u)} \alpha_j \hat{\mathbf{y}}(j) < 0 \ \forall \hat{\mathbf{y}} \in I_u \tag{3}$$

Since there are exponentially many constraints in this linear feasibility problem (3) also, we must construct a separation oracle to solve it. We construct a separation oracle that takes $\alpha_0, \ldots, \alpha_{n(u)}$ as input, and tries to find a $\hat{\mathbf{y}} \in I_u$ that maximizes $\alpha_0 + \sum_{j=1}^{n(u)} \alpha_j \hat{\mathbf{y}}(j)$. To do this, we will first write the following dynamic program (DP) that takes $c_0, \ldots c_{n(u)}$ for some $u \in V$ and returns the maximum possible value of $(c_0 + \sum_{j=1}^{n(u)} c_j \hat{\mathbf{y}}(j))$. We will use $c_{i,j}$ to denote $c_{B(i-1)+j}$.

$$d[b, j] = \max_{0 \leq j' \leq j} \{d[b-1, j-j'] + c_{i,j'}\} \quad \forall b \in [B], \forall t \in [0, S_u]$$

$$\text{Base case:} \qquad d[0, 0] = c_0 \tag{4}$$

Lemma 3.2. *Dynamic program* (4) *computes the maximum possible value of* $c_0 + \sum_{j=1}^{n(u)} c_j \hat{\mathbf{y}}(j)$ *for* $\hat{\mathbf{y}} \in I_u$.

We can use back pointers to find a $\hat{\mathbf{y}}^{max} \in I_u$ that maximizes the value. Then, our separation oracle will use DP (4) with $c_i = \alpha_i$ for every $i \in [0, \ldots, n(u)]$. If $d[B, S_u] < 0$, it will report that the hyperplane of the LP 3 satisfies the constraint, and if $d[B, S_u] \geq 0$, it will return the violated constraint $\alpha_0 + \sum_{j=1}^{n(u)} \alpha_j \hat{\mathbf{y}}^{max}(j) \geq 0$.

SO for Payoff Constraints. We construct a separation oracle for the payoff constraints in LP 2 that takes $\hat{\mathbf{x}}_\mathbf{u}$ and w_u $\forall u \in V$. The payoff, $\sum_{v \neq u} \sum_{b=1}^{B} \sum_{j=0}^{S_u} \sum_{k=0}^{S_v} \hat{\mathbf{y}}(b, j) h_u^v(b, j, k) \hat{\mathbf{x}}_\mathbf{v}(b, k)$
$= \sum_{b=1}^{B} \sum_{j=0}^{S_u} \hat{\mathbf{y}}(b, j) [\sum_{v \neq u} \sum_{k=0}^{S_v} h_u^v(b, j, k) \hat{\mathbf{x}}_\mathbf{v}(b, k)]$
We set $c_{b,j} = \sum_{v \neq u} \sum_{k=0}^{S_v} h_u^v(b, j, k) \hat{\mathbf{x}}_\mathbf{v}(b, k)$.

Then, we can run algorithm (4) to find the $\hat{\mathbf{y}}^{max} \in I_u$ that maximizes the above. Then, if $d[B, S_u] \leq w_u$, it will report that the payoff constraint is satisfied, and if $d[B, S_u] > w_u$, then it will return this violated constraint.

Algorithm (4) runs in time $O(B * (\max_{u \in [n]} S_u)^2)$ time using DP. Since algorithm (4) runs in polynomial time and both the separation oracles run in polynomial time, we can use ellipsoid method to solve the LP 2 in polynomial time. Next, since the vector $\{\hat{\mathbf{x}}_\mathbf{u}\}_{u \in V}$ obtained from the LP is in the marginal strategy space J_u, we must retrieve the corresponding strategy profile $\{\mathbf{x}_\mathbf{u}\}_{u \in V}$ in the original strategy space X_u. For this, we can use the efficient procedure developed in [2]. From the above analysis, together with Lemma 3.1, the next theorem follows.

Theorem 3.3. *There is a polynomial time algorithm to compute a NE of the zero-sum polymatrix Blotto game.*

REFERENCES

[1] Bharat Adsul, Jugal Garg, Ruta Mehta, and Milind Sohoni. 2011. Rank-1 Bimatrix Games: A Homeomorphism and a Polynomial Time Algorithm *(STOC '11)*. ACM, New York, NY, USA, 195–204. https://doi.org/10.1145/1993636.1993664

[2] Amir Mahdi Ahmadinejad, Sina Dehghani, Mohammad Taghi Hajiaghayi, Brendan Lucier, Hamid Mahini, and Saeed Seddighin. 2016. From Duels to Battlefields: Computing Equilibria of Blotto and Other Games *(AAAI'16)*. AAAI Press, Phoenix, Arizona, 369–375. http://dl.acm.org/citation.cfm?id=3015812.3015869

[3] Soheil Behnezhad, Sina Dehghani, Mahsa Derakhshan, Mohammad Taghi Hajiaghayi, and Saeed Seddighin. 2017. Faster and Simpler Algorithm for Optimal Strategies of Blotto Game. In *AAAI*.

[4] Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. 2016. Zero-Sum Polymatrix Games: A Generalization of Minmax. *Mathematics of Operations Research* 41 (Jan. 2016). https://doi.org/10.1287/moor.2015.0745

[5] Y. Cai and C. Daskalakis. 2011. On Minmax Theorems for Multiplayer Games. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 217–234. http://epubs.siam.org/doi/abs/10.1137/1.9781611973082.20 DOI: 10.1137/1.9781611973082.20.

[6] Xi Chen and Xiaotie Deng. 2006. Settling the Complexity of Two-Player Nash Equilibrium *(FOCS '06)*. IEEE Computer Society, Washington, DC, USA, 261–272. https://doi.org/10.1109/FOCS.2006.69

[7] Vincent Conitzer and Tuomas Sandholm. 2006. A Technique for Reducing Normal-form Games to Compute a Nash Equilibrium *(AAMAS '06)*. ACM, New York, NY, USA, 537–544. https://doi.org/10.1145/1160633.1160731

[8] George Bernard Dantzig. 1951. A Proof of the equivalence of the programming problem and the game problem. *Activity Analysis of Production and Allocationn, T C. Koopmans (eds). John Wiley & Sons, New York: 330-335* (1951).

[9] Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. 2006. The Complexity of Computing a Nash Equilibrium *(STOC '06)*. ACM, New York, NY, USA, 71–78. https://doi.org/10.1145/1132516.1132527

[10] M. Grötschel, L. Lovasz, and A. Schrijver. 1981. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1, 2 (June 1981), 169–197. https://doi.org/10.1007/BF02579273

[11] J. F. Nash. 1951. Non-cooperatie games. *Annals of Mathematics* 54(2) (1951), 286–295.

[12] John Von Neumann. 1944. *Theory Of Games And Economic Behavior*. Princeton University Press. http://archive.org/details/theoryofgamesand030098mbp