

Faster and Simpler Algorithm for Optimal Strategies of Blotto Game

Soheil Behnezhad*, Sina Dehghani*, Mahsa Derakhshan*,
MohammadTaghi HajiAghayi*, Saeed Seddighin*

Department of Computer Science, University of Maryland
{soheil, dehghani, mahsaa, hajiagha, sseddigh}@cs.umd.edu

Abstract

In the Colonel Blotto game, which was initially introduced by Borel in 1921, two colonels simultaneously distribute their troops across different battlefields. The winner of each battlefield is determined independently by a winner-take-all rule. The ultimate payoff of each colonel is the number of battlefields he wins. 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, and to politics. There has been persistent efforts for finding the optimal strategies for the Colonel Blotto game. After almost a century Ahmadinejad, Dehghani, Hajiaghai, Lucier, Mahini, and Seddighin [2] provided an algorithm for finding the optimal strategies in polynomial time.

Ahmadinejad *et al.* [2] first model the problem by a Linear Program (LP) with both an exponential number of variables and an exponential number of constraints which makes the problem intractable. Then they project their solution to another space to obtain another exponential-size LP, for which they can use Ellipsoid method. However, despite the theoretical importance of their algorithm, it is highly impractical. In general, even Simplex method (despite its exponential running time in practice) performs better than Ellipsoid method in practice.

In this paper, we provide the first polynomial-size LP formulation of the optimal strategies for the Colonel Blotto game. We use linear extension techniques. Roughly speaking, we project the strategy space polytope to a higher dimensional space, which results in lower number of facets for the polytope. In other words, we add a few variables to the LP, such that surprisingly the number of constraints drops down to a polynomial. We use this polynomial-size LP to provide a novel simpler and significantly faster algorithm for finding optimal strategies for the Colonel Blotto game.

We further show this representation is asymptotically tight, which means there exists no other linear representation of the problem with less number of constraints. We also extend our approach to multi-dimensional Colonel Blotto games, where each player may have different sorts of budgets, such as money, time, human resources, etc.

By implementing this algorithm we were able to run tests which were previously impossible to solve in a reasonable time. These informations, allow us to observe some interesting properties of Colonel Blotto; for example we find out the behaviour of players in the discrete model is very similar to the continuous model Roberson [34] solved.

*Supported in part by NSF CAREER award CCF-1053605, NSF BIGDATA grant IIS-1546108, NSF AF:Medium grant CCF-1161365, DARPA GRAPHs/AFOSR grant FA9550-12-1-0423, and another DARPA SIMPLEX grant.

1 Introduction

In the U.S. presidential election, the President is elected by the Electoral College system. In the Electoral College system, each state has a number of electoral votes, and the candidate who receives the majority of electoral votes is elected as the President of the United States. In most of the states¹, a winner-take-all rule determines the electoral votes, and the candidate who gets the majority of votes in a state will benefit from all the electoral votes of the corresponding state. Since the President is not elected by national popular vote directly, any investment in the states which are highly biased toward a party would be wasted. For example, a Democratic candidate can count on the electoral votes of states like California, Massachusetts, and New York, and a Republican candidate can count on the electoral votes of states like Texas, Mississippi, and South Carolina. This highlights the importance of those states that are likely to choose either parties, and would determine the outcome of the election. These states, known as *swing states* or *battleground states*, are the main targets of a campaign during the election, e.g., the main battleground states of the 2012 U.S. presidential election were Colorado, Florida, Iowa, New Hampshire, North Carolina, Ohio, Virginia, and Wisconsin. Now answers to the following questions seem to be essential: how can a national campaign distribute its resources like time, people, and money across different battleground states? What is the outcome of the game between two parties?

One might see the same type of competition between two companies which are developing new technologies. These companies need to distribute their efforts across different markets. The winner of each market would become the market-leader and takes almost all the benefits of the corresponding market [24, 25]. For instance, consider the competition between Samsung and Apple, where they both invest on developing products like cell-phones, tablets, and laptops, and all can have different specifications. Each product has its own specific market and the most plausible brand will lead that market. Again, a strategic planner with limited resources would face a similar question: what would be the best strategy for allocating the resources across different markets?

Colonel Blotto Game. The *Colonel Blotto* game, which was first introduced by Borel [10], provides a model to study the aforementioned problems. This paper was later discussed in an issue of *Econometrica* [11, 17, 18, 42]. Although the Colonel Blotto model was initially proposed to study a war situation, it has been applied for analyzing the competition in different contexts from sports, to advertisement, and to politics [30, 27, 29, 12, 24, 25]. In the original Colonel Blotto game two colonels fight against each other over different battlefields. They should simultaneously divide their troops among different battlefields without knowing the actions of their opponents. A colonel wins a battlefield if and only if the number of his troops dominates the number of troops of his opponent. The final payoff of each colonel, in its classical form, is the number of the battlefields he wins. The *MaxMin* strategy of a player maximizes the minimum gain that can be achieved. In two player zero-sum games a MaxMin strategy is also the optimal strategy, since any other strategy may result in a lower payoff across a rational player. It is also worth mentioning that in zero-sum games a pair of strategies is a Nash equilibrium if and only if both players are playing MaxMin strategies. Therefore finding MaxMin strategies results in finding the optimal strategies for players and also the Nash equilibria of the game. It is easy to show that solving Colonel Blotto is computationally harder than finding the optimal strategies of any two player game. Consider a two player game in which the players have x and y pure strategies, respectively. Now, construct a Blotto game with two battlefield in which player A has $x - 1$ troops and player B has $y - 1$ troops. Note that in this Blotto game, the number of strategies of the players are x and y respectively, and one can easily encode the payoff function of the original game in the partial payoffs of the two battlefields. Thus,

¹All states except Maine and Nebraska

any solution for Colonel Blotto yields a solution for an arbitrary two player game.

Colonel Blotto is a zero-sum game, but the fact that the number of pure strategies of the agents are exponential in the number of troops and the number of battlefields, makes the problem of finding optimal strategies quite hard. There were several attempts for solving variants of the problem since 1921 [41, 7, 8, 5, 40, 43, 34, 26, 22, 20, 25]. Most of the works consider special cases of the problem. For example many results in the literature relax the integer constraint of the problem, and study a *continuous* version of the problem where troops are divisible. For example, Borel and Ville [9] proposed the first solution for three battlefields. Gross and Wagner [21] generalized this result for any number of battlefields. However, they assumed colonels have the same number of troops. Roberson [34] computes the optimal strategies of the Blotto games in the continuous version of the problem where all the battlefields have the same weight, i.e. the game is symmetric across the battlefields. Hart [22] considered the discrete version, again when the game is symmetric across the battlefields, and solved it for some special cases. Very recently Ahmadijad, Dehghani, Hajiaghayi, Lucier, Mahini, and Seddighin [2] made a breakthrough in the study of this problem by finding optimal strategies for the Blotto games after nearly a century, which brought a lot of attention [33, 23, 39, 16, 1, 38, 37, 15, 31]. They obtain exponential sized LPs, and then provide a clever use of Ellipsoid method for finding the optimal strategies in polynomial time.

Although theoretically Ellipsoid method is a very powerful tool with deep consequences in complexity and optimization, it is “too inefficient to be used in practice” [6]. Interior point methods, and Simplex method (even though it has exponential running-time in the worst case) are “far more efficient” [6]. Thus a practical algorithm for finding optimal strategies for the Blotto games remains an open problem. In fact there has been huge studies in existence of efficient LP reformulations for different exponential-size LPs. For example Rothvoss [36] proved that the answer of the long-standing open problem, asking whether a graph’s perfect matching polytope can be represented by an LP with polynomial number of constraints, is negative. The seminal work of Applegate and Cohen [3] also provides polynomial-size LPs for finding an optimal oblivious routing. We are the first to provide a polynomial-size LP for finding the optimal strategies of the Colonel Blotto games. Although Ahmadijad *et al.* [2] use an LP with exponential number of constraints, our LP formulation has only $O(N^2K)$ constraints, where N denotes the number of troops and K denotes the number of battlefields. Consequently we provide a novel simpler and significantly faster algorithm using the polynomial-size LP.

Furthermore we show that our LP representation is asymptotically tight. The rough idea behind obtaining a polynomial-size LP is the following. Given a polytope P with exponentially many facets, we project P to another polytope Q in a higher dimensional space which has polynomial number of facets. Thus basically we are adding a few variables to the LP in order to reduce the number of constraints down to a polynomial. Q is called the *linear extension* of P . The minimum number of facets of any linear extension is called the *extension complexity*. We show that the extension complexity of the polytope of the optimal strategies of the Colonel Blotto game is $\Theta(N^2K)$. In other words, there exists no LP-formulation for the polytope of MaxMin strategies of the Colonel Blotto game with fewer than $\Theta(N^2K)$ constraints.

We also extend our approach to the *Multi-Resource Colonel Blotto* (MRCB) game. In MRCB, each player has different types of resources. Again the players distribute their budgets in the battlefields. Thus each player allocates a vector of resources to each battlefield. The outcome in each battlefield is a function of both players’ resource vectors that they have allocated to that battlefield. MRCB models a very natural and realistic generalization of the Colonel Blotto game. For example in U.S. presidential election, the campaigns distribute different resources like people, time, and money among different states. We provide an LP formulation for finding optimal strategies in MRCB with $\Theta(N^{2c}K)$ constraints and $\Theta(N^{2c}K)$ variables, where c is the number of resources. We

prove this result is also tight up to constant factors, since the extension complexity of MRCB is $\Theta(N^{2c}K)$.

By implementing our LP, we observe the payoff of players in the continuous version considered by Roberson [34] very well predicts the outcome of the game in the auctionary and symmetric version of our model.

2 Preliminaries

Throughout this paper we assume the number of battlefields is denoted by K and the number of troops of players A and B are denoted by A and B respectively. Also in some cases we use N to denote the number of troops of an unknown player.

Generally mixed strategies are shown by a probability vector over pure strategies. However at some points in this paper we project this representation to another space that specifies probabilities to each battlefield and troop count pair. More precisely, we map a mixed strategy x of player A to $\mathcal{G}^A(x) = \hat{x} \in [0, 1]^{d(A)}$ where $d(A) = K \times (A + 1)$. We may abuse this notation for convenience and use $\hat{x}_{i,j}$ to show the probability the mixed strategy x puts j troops in the i -th battlefield. Note that this mapping is not one-to-one. Similarly, we define $\mathcal{G}^B(x)$ to map a mixed strategy x of player B to a point in $[0, 1]^{d(B)}$ where $d(B) = K \times (B + 1)$. Let \mathcal{R}^A and \mathcal{R}^B denote the set of all possible mixed strategies of A and B in a Nash equilibrium. We define $\mathcal{P}_A = \{\hat{x} \mid \exists x \in \mathcal{R}^A, \mathcal{G}^A(x) = \hat{x}\}$ and $\mathcal{P}_B = \{\hat{x} \mid \exists x \in \mathcal{R}^B, \mathcal{G}^B(x) = \hat{x}\}$ to be the set of all Nash equilibrium strategies in the new space for A and B respectively.

Multi-Resource Colonel Blotto is a generalization of Colonel Blotto where each player may have different types of resources. In MRCB, there are K battlefields and c resource types. Players simultaneously distribute all their resources of all types over the battlefields. Let A_i and B_i denote the number of resources of type i player A and B respectively have. A pure strategy of a player would be a partition of his resources over battlefields. In other words, let $x_{i,j}$ and $y_{i,j}$ denote the amount of resources of type j , players A and B put in battlefield i respectively. A vector $x = \langle x_{1,1}, \dots, x_{K,c} \rangle$ is a pure strategy of player A if for any $1 \leq j \leq c$, $\sum_{i=1}^K x_{i,j} = A_j$. Similarly a vector $y = \langle y_{1,1}, \dots, y_{K,c} \rangle$ is a pure strategy of player B if for any $1 \leq j \leq c$, $\sum_{i=1}^K y_{i,j} = B_j$. Let $U^A(x, y)$ and $U^B(x, y)$ denote the payoff of A and B and let $U_i^A(x, y)$ and $U_i^B(x, y)$ show their payoff over the i -th battlefield respectively. Note that

$$U^A(x, y) = \sum_{i=1}^K U_i^A(x, y)$$

and

$$U^B(x, y) = \sum_{i=1}^K U_i^B(x, y).$$

On the other hand since MRCB is a zero-sum game $U_i^A(x, y) = -U_i^B(x, y)$. Similar to Colonel Blotto we define $\mathcal{R}_{\mathcal{M}}^A$ and $\mathcal{R}_{\mathcal{M}}^B$ to denote the set of all possible mixed strategies of A and B in a Nash equilibrium of MRCB and for any mixed strategy x for player A we define the mapping $\mathcal{G}_{\mathcal{M}}^A(x) = \hat{x} \in [0, 1]^{d^{\mathcal{M}}(A)}$ where $d^{\mathcal{M}}(A) = K \times (A_1 + 1) \dots \times (A_c + 1)$ and by $\hat{x}_{i,j_1, \dots, j_c}$ we mean the probability that in mixed strategy x , A puts j_t amount of resource type t in the i -th battlefield for any t where $1 \leq t \leq c$. We also define the same mapping for player B, $\mathcal{G}_{\mathcal{M}}^B(x) = \hat{x} \in [0, 1]^{d^{\mathcal{M}}(B)}$ where $d^{\mathcal{M}}(B) = K \times (B_1 + 1) \dots \times (B_c + 1)$. Lastly we define $\mathcal{P}_{\mathcal{M}}^A = \{\hat{x} \mid \exists x \in \mathcal{R}_{\mathcal{M}}^A, \mathcal{G}_{\mathcal{M}}^A(x) = \hat{x}\}$ and $\mathcal{P}_{\mathcal{M}}^B = \{\hat{x} \mid \exists x \in \mathcal{R}_{\mathcal{M}}^B, \mathcal{G}_{\mathcal{M}}^B(x) = \hat{x}\}$ to be the set of all Nash equilibrium strategies after the mapping.

3 LP Formulation

In this section we explain the LP formulation of Colonel Blotto proposed by Ahmadinejad *et al.* [2] and show how it can be reformulated in a more efficient way. Recall that in the Colonel Blotto game, we have two players A and B, each in charge of a number of troops, namely A and B respectively. Moreover, the game is played on K battlefields and every player's pure strategy is an allocation of his troops to the battlefields. Therefore, the number of pure strategies of the players is $\binom{A+K-1}{K-1}$ for player A and $\binom{B+K-1}{K-1}$ for player B.

The conventional approach to formulate the mixed strategies of a game is to represent every strategy by a vector of probabilities over the pure strategies. More precisely, a mixed strategy of a player is denoted by a vector of size equal to the number of his pure strategies, whose every element indicates the likelihood of taking a specific action in the game. The only constraint that this vector adheres to, is that the probabilities are non-negative and add up to 1. Such a formulation for Colonel Blotto requires a huge amount of space and computation, since the number of pure strategies of each player in this game is exponentially large.

To overcome this hardness, Ahmadinejad *et al.* [2] propose a more concise representation that doesn't suffer from the above problem. This is of course made possible by taking a significant hit on the simplicity of the description. They suggest, instead of indicating the probability of taking every action in the representation, we only keep track of the probabilities that a mixed strategy allocates a certain amount of troops to every battlefield. In other words, in the new representation, for every number of troops and any battlefield we have a real number, that denotes the probability of allocating that amount of troops to the battlefield. As a result, the length of the representation reduces from the number of pure strategies to $(A+1)K$ for player A and $(B+1)K$ for player B. This is indeed followed by a key observation: given the corresponding representations of the strategies of both players, one can determine the outcome of the game regardless of the actual strategies. In other words, the information stored in the representations of the strategies suffices to determine the outcome of the game.

In contrast to the conventional formulation, Ahmadinejad *et al.*'s representation is much more complicated and not well-understood. For example, in order to see if a representation corresponds to an actual strategy in the conventional formulation, we only need to verify that all of the probabilities are non-negative and their total sum is equal to 1. Ahmadinejad *et al.*'s representation, however, is not trivial to verify. Apart from the trivial constraints such as the probabilities add up to 1 or the number of allocated troops matches the number of the player's troops, there are many other constraints to be met. Moreover, it is not even proven whether such a representation can be verified with a polynomial number of linear constraints.

Ahmadinejad *et al.* [2] leverage the new representation to determine the equilibria of Colonel Blotto in polynomial time. Recall that in zero-sum games such as Colonel Blotto, the minmax strategies are the same as the maxmin strategies, and the game is in Nash Equilibrium if and only if both players play a maxmin strategy [32]. Roughly speaking, the high-level idea of Ahmadinejad *et al.* is to find a mixed strategy which performs the best against every strategy of the opponent. By the equivalence of the minmax and maxmin strategies then, one can show such a strategy is optimal for that player. Therefore, the naive formulation of the equilibria of Blotto is as follows:

$$\begin{aligned}
 & \max && u && (1) \\
 \text{s.t.} &&& \hat{x} \text{ is a valid strategy for player A} \\
 &&& U^A(\hat{x}, \hat{y}) \geq u && \forall \hat{y}
 \end{aligned}$$

Note that, \hat{x} is a vector of size $(A+1)K$ that represents a strategy of player A. Similarly, for every mixed strategy of player B, represented by \hat{y} , we have a constraint to ensure \hat{x} achieves a payoff of at least u against \hat{y} . Notice that the only variables of the program are the probabilities encoded in vector \hat{x} . All other parameters are given as input, and hence appear as constant coefficients in the program. As declared, there are two types of constraints in Program 1. The first set of constraints ensures the validity of \hat{x} , and the second set of constraints makes sure \hat{x} performs well against every strategy of player B. Ahmadinejad *et al.* [2] call the first set *the membership constraints* and the second set *the payoff constraints*. Throughout the paper Since for every mixed strategy, there exists a best response of the opponent which is pure, one can narrow down the payoff constraints to the pure strategies of player B.

The last observation of Ahmadinejad *et al.* [2] is to show both types of the constraints are convex in the sense that if two strategy profiles \hat{x}_1 and \hat{x}_2 meet either set of constraints, then $\frac{\hat{x}_1 + \hat{x}_2}{2}$ is also a feasible solution for that set. This implies that Program 1 is indeed a linear program that can be solved efficiently via the Ellipsoid method. However, Ahmadinejad *et al.*'s algorithm is practically impossible to run, as its computational complexity is $O(N^{12}K^4)$.

The reason Ahmadinejad *et al.*'s algorithm is so slow is that their LP has exponentially many constraints. Therefore, they need to run the Ellipsoid algorithm run solve the program. In addition to this, their separation oracle is itself a linear program with exponentially many constraints which is again very time consuming to run. However, a careful analysis shows that these exponentially many constraints are all necessary and none of them are redundant. This implies that the space of the LP as described by Ahmadinejad *et al.* requires exponentially many constraints to formulate and hence we cannot hope for a better algorithm. A natural question that emerges, however, is whether we can change the space of the LP to solve it with a more efficient algorithm?

In this paper we answer the above question in the affirmative. There has been persistent effort to find efficient formulations for many classic polytopes. As an example, *spanning trees* of a graph can be formulated via a linear program that has an exponential number of linear constraints. It is also not hard to show none of those constraints are redundant [14]. However, Martin [28] showed that the same polytope can be formulated with $O(n^3)$ linear constraints where n is the number of nodes of the graph. Other examples are *the permutahedron* [19], *the parity polytope* [35], and *the matching polytope* [36]. In these examples, a substantial decrease in the number of constraints of the linear formulation of a problem is made possible by adding auxiliary variables to the program. Our work follows the same guideline to formulate the equilibria of Blotto with a small number of constraints.

In Section 4, we explain how to formulate the membership and payoff limitations with a small number of linear constraints. Finally in Section 5, we show that our formulation is near optimal. In other words, we show that any linear program that formulates the equilibria of Blotto, has to have as many linear constraints as the number of constraints in our formulation within a constant factor. We show this via *rectangle covering lower bound* proposed by Yannakakis [44]

4 Main Results

In this section we give a linear program to find a maxmin strategy for a player in an instance of Colonel Blotto with polynomially many constraints and variables. To do this, we describe the same representation described by Ahmadinejad *et al.*'s [2] LP in another dimension, to reduce the number of constraints. This gives us a much better running time, since they had to use ellipsoid method to find a solution for their LP in polynomial time, which makes their algorithm very slow and impractical. We define a *layered graph* for each player and show any mixed strategy of a player

can be mapped to a particular flow in his layered graph. Our LP includes two set of constraints, *membership constraints* and *payoff constraints*. Membership constraints guarantee we find a valid strategy and payoff constraints guarantee this strategy minimizes the maximum benefit of the other player.

Definition 4.1 (Layered Graph) For an instance of a Blotto game with K battlefields, we define a layered graph for a player with N troops as follows: The layered graph has $K + 1$ layers and $N + 1$ vertices in each layer. Let $v_{i,j}$ denote the j 'th vertex in the i 'th layer ($0 \leq i \leq K$ and $0 \leq j \leq N$). For any $1 \leq i \leq K$ there exists a directed edge from $v_{i-1,j}$ to $v_{i,l}$ iff $0 \leq j \leq l \leq N$. We denote the layered graph of player A and B by \mathcal{L}^A and \mathcal{L}^B respectively.

Based on the definition of layered graph we define *canonical paths* as follows:

Definition 4.2 (Canonical Path) A canonical path is a directed path in a layered graph that starts from $v_{0,0}$ and ends at $v_{K,N}$.

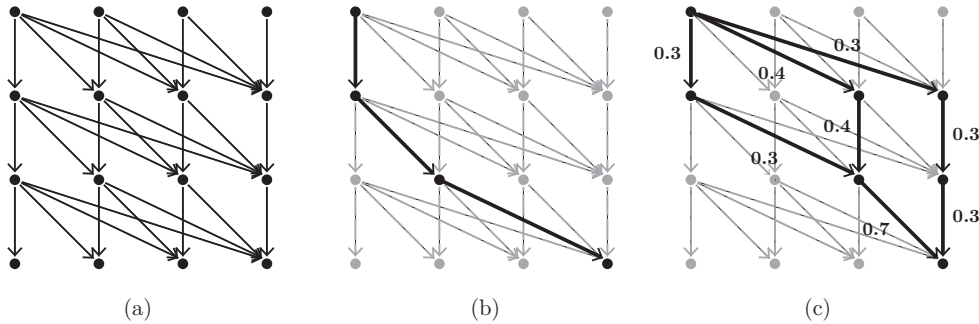


Figure 1: Figure (a) shows a layered graph for a player with 3 troops playing over 3 battlefields. In Figure (b) a canonical path corresponding to a pure strategy where the player puts no troops on the first battlefield, 1 troop on the second one and two troops on the 3rd one is shown. Figure (c) shows a flow of size 1, that is a representation of a mixed strategy consisting of three pure strategies with probabilities 0.3, 0.4 and 0.3.

Figure 1 shows a layered graph and a canonical path. Now, we give a one-to-one mapping between canonical paths and pure strategies.

Lemma 4.3 Each pure strategy for a player is equivalent to exactly one canonical path in the layered graph of him and vice versa.

Proof. Since the edges in the layered graph exist only between two consecutive layers, each canonical path contains exactly K edges. Let p be an arbitrary canonical path in the layered graph of a player with N troops. In the equivalent pure strategy put l_i troops in the battlefield i if p contains the edge between $v_{i-1,j}$ and $v_{i,j+l_i}$ for some j . By definition of the layered graph, we have $l_i \geq 0$. Also since p starts from $v_{0,0}$ and ends in $v_{K,N}$ we have $\sum_{i=0}^K l_i = N$. Therefore this strategy is a valid pure strategy.

On the other hand, let s be a valid pure strategy and let s_i denote the total number of troops in battlefields 1 to i in strategy s . We claim the set of edges between $v_{i-1,s_{i-1}}$ and v_{i,s_i} for $1 \leq i \leq K$ is a canonical path. Note that $s_0 = 0$ and $s_K = N$ also the endpoint of any of such edges is the

starting point of the edge chosen from the next layer, so we have constructed a valid canonical path. \square

So far it is clear how layered graphs are related to pure strategies using canonical paths. Now we explain the relation between mixed strategies and flows of size 1 where $v_{0,0}$ is the source and $v_{K,N}$ is the sink. One approach to formulate the mixed strategies of a game is to represent every strategy by a vector of probabilities over the pure strategies. Since based on Lemma 4.3 each pure strategy is equivalent to a canonical path in the layered graph; for any pure strategy s with probability $P(s)$ in a mixed strategy we assign a flow of size $P(s)$ to the corresponding canonical paths of s in the layered graph. All these paths begin and end in $v_{0,0}$ and $v_{K,N}$ respectively. Therefore since $\sum P(s) = 1$ for all pure strategies of a mixed strategy, the size of the corresponding flow would be exactly 1.

Corollary 4.4 *For any mixed strategy of a player with N troops there is exactly one corresponding flow from vertex $v_{0,0}$ to $v_{K,N}$ in the layered graph of that player.*

Note that although we map any given mixed strategy to a flow of size 1 in the layered graph, this is not a one-to-one mapping because several mixed strategies could be mapped to the same flow. However in the following lemma we show that this mapping is surjective.

Lemma 4.5 *For any flow of size 1 from $v_{0,0}$ to $v_{K,N}$ in the layered graph of a player with N troops, there is at least one mixed strategy of that player with a polynomial size support that is mapped to this flow.*

Proof. First, note that we can decompose any given flow to polynomially many flow paths from source to sink [13]. A flow path is a flow over only one path from source to sink. One algorithm to find such decomposition finds a path p from source to sink in each step and subtracts the minimum passing flow through its edges from every edge in p . The steps are repeated until there is no flow from source to sink. Since the flow passing through at least one edge becomes 0 at each step, the total number of these paths will not exceed the total number of edges in the graph. This means the number of flow paths in the decomposition will be polynomial.

Now, given a flow of size 1 from $v_{0,0}$ to $v_{K,N}$, we can basically decompose it to polynomially many flow paths using the aforementioned algorithm. The paths over which these flow paths are defined correspond to pure strategies and the amount of flow passing through each, corresponds to its probability in the mixed strategy. \square

Using the flow representation for mixed strategies and the shown properties for it, we give the first LP with polynomially many constraints and variables to find a maxmin strategy for any player in an instance of Colonel Blotto. Our LP consists of two set of constraints, the first set (membership constraints) ensures we have a valid flow of size 1. This means we will be able to map the solution to a valid mixed strategy. The second set of constraints are needed to ensure the minimum payoff of the player we are finding the maxmin strategy for, is at least u . Now, by maximizing u we will get a maxmin strategy. In the following theorem we prove \mathcal{P}_A could be formulated with polynomially many constraints and variables. Note that one can swap A and B and use the same LP to formulate \mathcal{P}_B .

Theorem 4.6 *In an instance of Colonel Blotto, with K battlefields and at most N troops for each player, \mathcal{P}_A could be formulated with $\Theta(N^2K)$ constraints and $\Theta(N^2K)$ variables.*

1	0	0	0
0	1	0	0
0	0	1	0

(a)

0.3	0	0.4	0.3
0.7	0	0.3	0
0.3	0.7	0	0

(b)

Figure 2: Figure (a) shows $P_{k,i}^A$ for the pure strategy specified in Figure 1-b and Figure (b) shows $P_{k,i}^A$ for the mixed strategy specified in Figure 1-c. The rows correspond to battlefields and the columns correspond to the number of troops.

Proof. The high-level representation of our LP is as follows:

$$\begin{aligned}
& \max && u && (2) \\
& \text{s.t.} && \hat{x} \text{ is a valid strategy for player A} \\
& && U^B(\hat{x}, \hat{y}) \leq -u \quad \forall \hat{y}.
\end{aligned}$$

The strategies \hat{x} and \hat{y} are represented using a flow of size 1 in the layered graph of player A and B respectively. In Lemma 4.5 we proved any valid flow representation could be mapped to a mixed strategy.

To ensure we have a valid flow of size 1 from $v_{0,0}$ to $v_{K,A}$ in \mathcal{L}^A (recall that \mathcal{L}^A denotes the layered graph of player A), we use the classic LP representation of flow [4]. That is, not having any negative flow and the total incoming flow of each vertex must be equal to its total outgoing flow except for the source and the sink. We denote the amount of flow passing through the edge from $v_{k,i}$ to $v_{k+1,j}$ by variable $F_{k,i,j}$. The exact membership constraints are shown in Linear Program 1-a.

On the other hand, we maximize the guaranteed payoff of player A, by bounding the maximum possible payoff of player B. To do this, first note that for any given strategy of player A, there exists a pure strategy for player B, that maximizes his payoff. Let $P_{k,j}^A$ denote the probability that player A puts j troops in the k -th battlefield. Figure 2 shows the value of $P_{k,j}^A$ for the illustrated examples in Figure 1. We can compute these probabilities using the variables defined in the previous constraints, as follows:

$$P_{k,i}^A = \sum_{i=0}^{A-j} F_{k,i,i+j} \quad (3)$$

By having these probabilities we can compute the expected payoff that player B gets over battlefield k , if he puts i troops in it. Moreover consider a given canonical path p in \mathcal{L}^B and let s_p be the pure strategy of player B, equivalent to p . We use $W_{k,i}^B$ to denote the expected payoff of player B over battlefield k by putting i troops in it. This means the expected payoff of playing strategy s_p would be $\sum W_{k,j-i}^B$ for any k, i and j such that there exists an edge from $v_{k,i}$ to $v_{k+1,j}$ in p . It is possible to compute $W_{k,i}^B$ using the following equation:

$$W_{k,i}^B = \sum_{l=0}^A P_{k,l}^A \times U_k^B(i, l) \quad \forall k : 1 \leq k \leq K \quad (4)$$

Note that both equations to compute $P_{k,i}^A$ and $W_{k,i}^B$ are linear and could be computed in our LP.

$$\begin{aligned}
& \max \quad u \\
& \text{(a)} \quad \begin{cases} \sum_{i=0}^l F_{k,i,l} = \sum_{j=l}^A F_{k+1,l,j} & \forall k, l : 1 \leq k \leq K-1, 0 \leq l \leq A \\ F_{k,i,j} \geq 0 & \forall k, i, j : 1 \leq k \leq K, 0 \leq i \leq j \leq A \\ \sum_{j=l}^A F_{1,l,j} = 0 & \forall l : 0 < l \leq A \\ \sum_{j=0}^A F_{1,0,j} = 1 \\ \sum_{j=0}^A F_{K,j,A} = 1 \end{cases} \\
& \text{(b)} \quad \begin{cases} P_{k,i}^A = \sum_{i=0}^{A-j} F_{k,i,i+j} & \forall k, j : 1 \leq k \leq K, 0 \leq j \leq A \\ W_{k,i}^B = \sum_{l=0}^A P_{k,l}^A \times U_k^B(i, l) & \forall k, i : 1 \leq k \leq K, 0 \leq i \leq B \\ D_{0,i}^B = 0 & \forall i : 0 \leq i \leq B \\ D_{k,i}^B \geq D_{k-1,j}^B + W_{k-1,i-j}^B & \forall i, j : 0 \leq j \leq i \leq B \\ D_{K,B}^B \leq -u \end{cases}
\end{aligned}$$

Linear Program 1: The detailed linear program to find a maxmin strategy for player A. The first set of constraints denoted by (a) ensure we get a valid flow of size 1 from $v_{0,0}$ to $v_{K,A}$ in the layered graph of player A (a mixed strategy of him) and the second set of constraints denoted by (b) ensure the guaranteed payoff of player A is at least u . The value of variable $F_{k,i,j}$ is the amount of flow passing through the edge from $v_{k,i}$ to $v_{k+1,j}$ for any valid k, i and j . Variable $D_{i,j}^B$ is the size of the maximum weighted path from $v_{0,0}$ to $v_{i,j}$ in the layered graph of player B, therefore $D_{K,B}^B$ denotes the maximum payoff of B and u is the guaranteed payoff of player A. For an informal explanation of the LP see the text.

Assume $W_{k,i}^B$ is the weight of the edge from $v_{k,j}$ to $v_{k+1,i+j}$ in \mathcal{L}^B . Given the probability distribution of player A (which we denoted by $P_{k,i}^A$), the problem of finding the pure strategy of B with the maximum possible expected payoff, would be equivalent to finding a path from $v_{0,0}$ to $v_{K,B}$ with the maximum weight.

To find the path with the maximum weight from $v_{0,0}$ to $v_{K,B}$, we define an LP variable $D_{k,i}^B$ where its value is equal to the weight of the maximum weighted path from $v_{0,0}$ to $v_{k,i}$ and we update it using a simple dynamic programming like constraint:

$$D_{k,i}^B \geq D_{k-1,j}^B + W_{k-1,i-j}^B \quad \forall i, j : 0 \leq j \leq i \leq B$$

The maximum weighted path from $v_{0,0}$ to $v_{K,B}$ would be equal to the value of $D_{K,B}^B$. The detailed constraints are shown in Linear Program 1-b.

Note that the number of variables we use in Linear Program 1 is as follows:

- Variables of type $F_{k,i,l}$: $\Theta(A^2K)$.
- Variables of type $P_{k,i}^A$: $\Theta(AK)$.
- Variables of type $W_{k,i}^B$: $\Theta(BK)$.
- Variables of type $D_{k,i}^B$: $\Theta(BK)$.

Therefore the total number of variables is $\Theta(N^2K)$. Also note that the number of non-negativity constraints ($F_{k,i,j} \geq 0$) is more than any other constraints and is $\Theta(N^2K)$, therefore the total number of constraints is also $\Theta(N^2K)$. \square

To obtain a mixed strategy for player A, it suffices to run Linear Program 1 and find a mixed strategy of A that is mapped to the flow it finds. Note that based on Lemma 4.5 such mixed strategy always exists. Afterwards we do the same for player B by simply substituting A and B in the LP.

5 Lower Bound

A classic approach to reduce the number of LP constraints needed to describe a polytope is to do it in a higher dimension. More precisely, adding extra variables might reduce the number of facets of a polytope. This means a complex polytope may be much simpler in a higher dimension. This is exactly what we did in Section 4 to improve Ahmadinejad *et al.*'s algorithm. In this section we prove that any LP formulation that describes solutions of a Blotto game requires at least $\Theta(N^2K)$ constraints, no matter what the dimension is. This proves the given LP in Section 4 is tight up to constant factors.

The minimum needed number of constraints in any formulation of a polytope P is called *extension complexity* of P , denoted by $\text{xc}(P)$. It is not usually easy to prove a lower bound directly on the extension complexity, because all possible formulations of the polytope must be considered. A very useful technique given by Yannakakis [44] is to prove a lower bound on the *positive rank* of the *slack matrix* of P which is proven to be equal to $\text{xc}(P)$. Note that you could define the slack matrix over any formulation of P and its positive rank would be equal to $\text{xc}(P)$, which means you do not have to worry about all possible formulations. To prove this lower bound we use a method called *rectangle covering lower bound*, already given in Yannakakis's paper. We will now formally define some of the concepts we used:

Definition 5.1 (Extension Complexity) *Extension complexity of a polytope P , denoted by $\text{xc}(P)$ is the smallest number of facets of any other higher dimensional polytope Q that has a linear projection function π with $\pi(Q) = P$.*

The next concept we need is slack matrix, which is a matrix of non-negative real values where its columns correspond to vertices of P and its rows correspond to its facets. The value of each element of slack matrix is basically the distance of the vertex corresponding to its column from the facet corresponding to its row. More formally:

Definition 5.2 (Slack Matrix) *Let $\{v_1, \dots, v_v\}$ be the set of vertices of P and let $\{x \in \mathbb{R}^n \mid Ax \leq b\}$ be the description of it. The slack matrix of P denoted by S^P , is defined by $S_{ij}^P = b_i - A_i v_j$.*

Also, the non-negative rank of a matrix S is the minimum number m such that S could be factored into two non-negative matrices F and V with dimensions $f \times m$ and $m \times v$.

Definition 5.3 (Non-negative Rank) *We define the non-negative rank of a matrix S with f rows and v columns, denoted by $\text{rk}_+(S)$ to be:*

$$\text{rk}_+(S) = \min\{m \mid \exists F \in \mathbb{R}_{\geq 0}^{f \times m}, V \in \mathbb{R}_{\geq 0}^{m \times v} : S = FV\} \quad (5)$$

Yannakakis [44] proved that $\text{xc}(P) = \text{rk}_+(S^P)$. Therefore instead of proving a lower bound on the extension complexity of P , it only suffices to prove a lower bound on the positive rank of the corresponding slack matrix. As mentioned before, to do so, we will use the rectangle covering lower bound. A rectangle covering for a given non-negative matrix S is the minimum number of rectangles needed, to cover all the positive elements of S and none of its zeros (Figure 3), formally defined as follows:

Definition 5.4 (Rectangle Covering) *Suppose $r = \text{rk}_+(S)$ and let $S = UV$ be a factorization of S by non-negative matrices U and V . Let $\text{supp}(S)$ denote the set of all the positive values of S . Then*

$$\text{supp}(S) = \bigcup_{l=1}^r (\{i | U_{il} > 0\} \times \{j | V_{lj} > 0\})$$

is a rectangle covering of S with r rectangles.

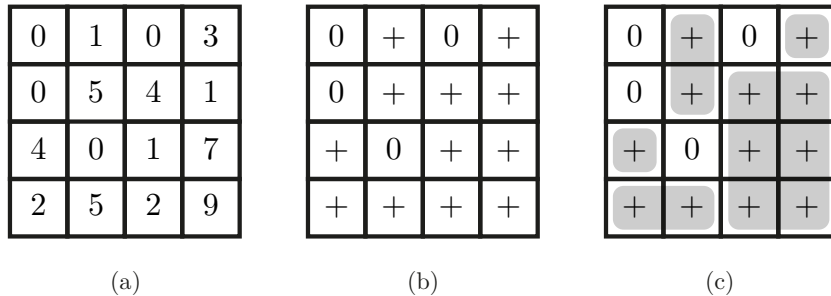


Figure 3: Figure (a) shows a sample matrix, in Figure (b) we change any non-negative value in the matrix of Figure (a) to “+” and in Figure (c) all these non-negative elements are covered by the minimum possible number of rectangles. Note that the non-negative rank of the matrix in Figure (a) could not be less than 5 (the number of rectangles).

Yannakakis showed that the number of rectangles in a minimum rectangle covering could never be greater than $\text{rk}_+(S)$, using a very simple proof. This means any lower bound of it, is also a lower bound of the actual $\text{rk}_+(S)$. This is the technique we use in the proof of the following lemma, which is used later to prove the main theorem:

Lemma 5.5 *The extension complexity of the membership polytope of a player in an instance of Blotto with K battlefields and N troops for each player is at least $\Theta(N^2K)$.*

Proof. Assume w.l.g. that we are trying to describe the polytope of all valid strategies of player A , denoted by P . One way of describing this polytope was explained in the LP described in Section 4. Now from its membership constraints, only consider the ones that ensure the non-negativity of the flow passing through the edges of the layered graph of player A :

$$F_{i,j,t} \geq 0 \quad \forall i, j, t : 0 \leq i \leq K - 1, 0 \leq j \leq j + t \leq A \tag{6}$$

From now on, only consider the part of the slack matrix corresponding to these constraints (we may occasionally call it the slack matrix), its columns as mentioned before, correspond to the vertices of the polytope, which in this case are all possible pure strategies of player A . Also its rows correspond to the mentioned constraints. Recall that any pure strategy is a canonical path in

the layered graph of player A . Note that the slack matrix element corresponding to any arbitrarily chosen non-negativity constraint $e \geq 0$ and any arbitrary vertex v_j corresponding to a pure strategy S is 0 iff the equivalent canonical path of S does not contain e and is 1 if it does; since the elements of the slack matrix are calculated using the formula $S_{ij}^P = b - A_i v_j$ and in this case b is always zero and $A_i v_j$ is -1 iff S contains the edge in the constraint and is zero otherwise. This implies it is only consisted of zero and one values.

We call any edge $F_{b,i,j}$ with $j - i > \frac{N}{2}$ a long edge. A canonical path may only contain at most one such edge. On the other hand, any rectangle in the rectangle covering is basically a set of vertices and a set of constraints. Note that all the equivalent pure strategies of those vertices must contain the edges over which the constraints are defined. Therefore no rectangle could contain more than one constraint over long edges. The number of long edges in the layered graph is exactly

$$\frac{K(N - \lceil \frac{N+1}{2} \rceil)(N - \lceil \frac{N+1}{2} \rceil + 1)}{2}. \quad (7)$$

Therefore the minimum number of rectangles to cover all non-negative elements of the slack matrix is at least of the same size and therefore $\Theta(N^2 K)$. \square

Theorem 5.6 *In an instance of Blotto with K battlefields and N troops for each player the extension complexity of \mathcal{P}_A is $\Theta(N^2 K)$.*

Proof. Assume the utility function is defined as follows:

$$U^A(\hat{x}, \hat{y}) = 0 \quad \forall \hat{x}, \hat{y}. \quad (8)$$

This means any possible strategy is a maxmin strategy for both players. In particular, the polytope of all possible maxmin strategies of any arbitrarily chosen player of this game, denoted by P contains all possible valid strategies. Now using Lemma 5.5 we know $\text{xc}(P)$ is at least $\Theta(N^2 K)$. On the other hand, in Section 4 we gave an LP with $\Theta(N^2 K)$ constraints to formulate the maxmin polytope, therefore the extension complexity of it is exactly $\Theta(N^2 K)$. \square

6 Multi-Resource Colonel Blotto

In this section we explain how our results could be generalized to solve Multi-Resource Colonel Blotto, or *MRCB*. We define MRCB to be exactly the same game as Colonel Blotto, except instead of having only one type of resource (troops), players may have any constant number of resource types. Examples of resource types would be time, money, energy, etc.

To solve MRCB we generalize some of the concepts we defined for Colonel Blotto. We first define generalized layered graphs and generalized canonical paths as follows:

Definition 6.1 (Generalized Layered Graph) *Let N_m denote the total number of available resources of m -th resource type for player X . The generalized layered graph of X has $K \times N_1 \times \dots \times N_c$ vertices denoted by $v(i, r_1, \dots, r_c)$, with a directed edge from $v(i, r_1, \dots, r_{m-1}, x, r_{m+1}, \dots, r_c)$ to $v(i+1, r_1, \dots, r_{m-1}, y, r_{m+1}, \dots, r_c)$ for any possible i, r and $0 \leq x \leq y \leq N_m$.*

Definition 6.2 (Generalized Canonical Path) *A generalized canonical path is defined over a generalized layered graph and is a directed path from $v_{0,0,\dots,0}$ to v_{K,N_1,\dots,N_c} .*

By these generalization we can simply prove that pure strategies of a player are equivalent to canonical paths in his generalized layered graph and there could be a surjective mapping from his mixed strategies to flows of size 1 from $v(0, \dots, 0)$ to $v(K, N_1, \dots, N_c)$ using similar techniques we used in Section 4.

Lemma 6.3 *Each pure strategy for a player in an instance of MRCB is equivalent to exactly one generalized canonical path in the generalized layered graph of him and vice versa.*

Lemma 6.4 *For any flow f of size 1 from $v(0, \dots, 0)$ to $v(K, N_1, \dots, N_c)$ in the generalized layered graph of a player with N_i troops of type i , there is at least one mixed strategy with a polynomial size support that is mapped to f .*

Using these properties, we can prove the following theorem:

Theorem 6.5 *In an instance of MRCB, \mathcal{P}_A^M could be formulated with $O(N^{2c}K)$ constraints and $\Theta(N^{2c}K)$ variables.*

Proof. The linear program would again look like this:

$$\begin{aligned} \max \quad & u & (9) \\ \text{s.t.} \quad & \hat{x} \text{ is a valid strategy for player A} \\ & U^B(\hat{x}, \hat{y}) \leq -u \quad \forall \hat{y} \end{aligned}$$

For the first set of constraints (membership constraints) we can use the flow constraints over the generalized layered graph of player A to make sure we have a valid flow of size 1 from $v(0, \dots, 0)$ to $v(K, N_1, \dots, N_c)$. And for the second constraint (payoff constraint) we can find the maximum payoff of player B using a very similar set of constraints to the described one in Section 4, but over the generalized layered graph of player B. \square

We can also prove the following lowerbound for MRCB.

Theorem 6.6 *In an instance of MRCB, the extension complexity of \mathcal{P}_A^M is $\Theta(N^{2c}K)$.*

Proof. The proof is very similar to the proof of Theorem 5.6. We only consider the rectangle covering lower bound over the part of the slack matrix corresponding to the non-negativity of flow through edges in the maxmin. We call an edge from $v(i, r_1, \dots, r_{m-1}, x, r_{m+1}, \dots, r_c)$ to $v(i+1, r_1, \dots, r_{m-1}, y, r_{m+1}, \dots, r_c)$ long if $y-x > \frac{r_m}{2}$. No generalized canonical path could contain more than c long edges therefore no rectangle could cover more than c constraints. On the other hand there are $\Theta(N^{2c}K)$ long edges in the layered graph. Since c is a constant number the extension complexity is $\Omega(N^{2c}K)$. Moreover since we already gave a possible formulation with $O(N^{2c}K)$ constraints in Theorem 6 the extension complexity is also $O(N^{2c}K)$ and therefore $\Theta(N^{2c}K)$. \square

7 Experimental Results

We implemented the algorithm described in Section 4 using Simplex method to solve the LP. We ran the code on a machine with a dual-core processor and an 8GB memory. The running time and the number of constraints of the LP for each input is shown in Table 1. Using this fast implementation we were able to run the code for different cases. In this section we will mention some of our observations that mostly confirm the theoretical predications.

K	A	B	Constraints	Running Time
10	20	20	3595	0m3.575s
10	20	25	4855	0m3.993s
10	20	30	6365	0m6.695s
10	25	25	5295	0m8.245s
10	25	30	6805	0m7.502s
10	30	30	7320	0m30.955s
15	20	20	5065	0m14.965s
15	20	25	6950	0m11.842s
15	20	30	9210	0m24.196s
15	25	25	7440	0m46.165s
15	25	30	9700	0m31.714s
15	30	30	10265	2m20.776s
20	20	20	6535	0m46.282s
20	20	25	9045	0m35.758s
20	20	30	12055	0m38.507s
20	25	25	9585	1m38.367s
20	25	30	12595	0m51.795s
20	30	30	13210	9m13.288s

Table 1: The number of constraints and the running time of the implemented Colonel Blotto based on different inputs. The first column shows the number of battlefields, the second and third columns show the number of troops of player A and B respectively. The number of constraints does not include the non-negativity constraints since by default every variable was assumed to be non-negative in the library we used.

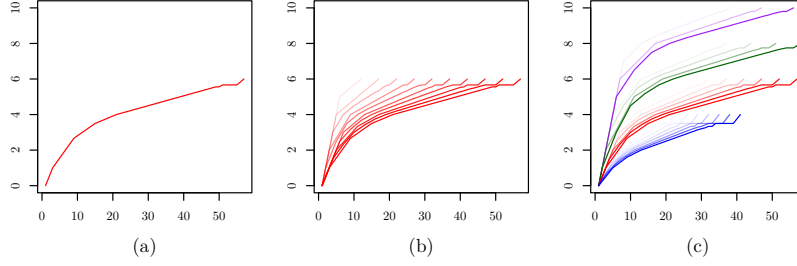


Figure 4: The y-axis is the payoff of A in the Nash equilibrium and the x-axis shows the value of $A - B$. In Figure (a), $K = 6$ and $B = 10$. In Figure (b), $K = 6$ and for different values of B in the range of 1 to 10 the same diagram as Figure (a-) is drawn. Figure (c) is the same plot as Figure (b) but for different values of K . For instance for the blue lines $K = 4$, for the red lines $K = 6$, for the green lines $K = 8$ and for the purple lines $K = 10$. In all examples payoff function of player A over a battlefield i , is $sgn(x_i - y_i)$ where x_i and y_i denote the number of troops A and B put in the i -th battlefield respectively.

An instance of Colonel Blotto is symmetric if the payoff function is the same for all battlefields, or in other words for any pure strategies x and y for player A and B and for any two battlefields i and j , $U_i^A(x, y) = U_j^A(x, y)$. Also, an instance of blotto is auctionary if the player allocating more troops in a battlefield wins it (gets more payoff over that battlefield). More formally in an auctionary instance of Colonel Blotto, if x and y are some pure strategies for player A and B respectively, then

$$U_i^A(x, y) = \begin{cases} +w(i), & \text{if } x_i > y_i \\ 0, & \text{if } x_i = y_i \\ -w(i), & \text{otherwise} \end{cases}$$

Recall that x_i and y_i denote the amount of troops A and B put in the i -th battlefield respectively.

Note that in an auctionary Colonel Blotto if $A \geq (B + 1)K$, then by putting $B + 1$ troops in each battlefield, player A wins all the battlefields and gets the maximum possible overall payoff. On the other hand if $A = B$, the payoff of player A in any Nash equilibrium is exactly 0 because there is no difference between player A and player B by definition of an auctionary Colonel Blotto if $A = B$, and any strategy for A could also be used for B and vice versa. W.l.g. we can ignore the case where $A < B$. However, it is not easy to guess the payoff of A in a Nash equilibrium if $B \leq A < (B + 1)K$. After running the code for different inputs, we noticed the growth of U^A with respect to A (when B is fixed) has a common shape for all inputs. Figure 4 shows the chart for different values of A , B and N .

There has been several attempts to mathematically find the optimum payoff of players under different conditions. For example Roberson [34] considered the continuous version of Colonel Blotto and solved it. Hart [22] solved the symmetric and auctionary model and solved it for some special cases. Little is known about whether it is possible to completely solve the discrete version when the game is symmetric and auctionary or not.

Surprisingly, we observed the payoff of players in the symmetric and auctionary discrete version, is very close to the continuous version Roberson considered. The payoffs are specially very close when the number of troops are large compared to the number of battlefields, making the strategies more flexible and more similar to the continuous version. Figure 5 compares the payoffs in the aforementioned models. In Roberson's model in case of a tie, the player with more resources wins while in the normal case there is no such assumption; however a tie rarely happens since by adding

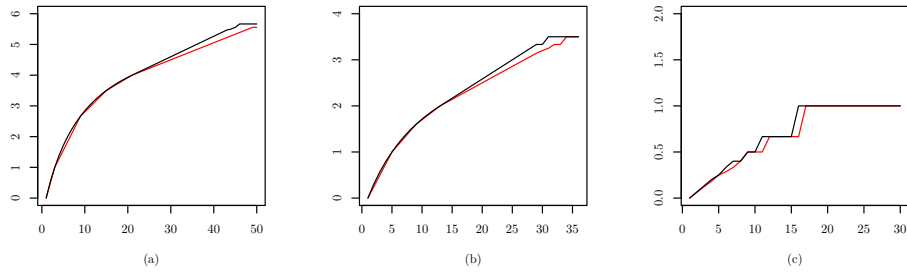


Figure 5: The y-axis is the payoff of A in the Nash equilibrium and the x-axis shows the value of $A - B$. The black and red line show the payoff in the continuous model and discrete model respectively. In figure (a), $K = 6$ and $B = 10$, in figure (b), $K = 4$ and $B = 12$ and in figure (c), $K = 2$ and $B = 30$.

any small amount of resources the player losing the battlefield would win it.

References

- [1] ACM TechNews. 2016. Umd-led team first to solve well-known game theory scenario. <http://technews.acm.org/archives.cfm?fo=2016-02-feb/feb-17-2016.html>.
- [2] Ahmadinejad, M.; Dehghani, S.; Hajiaghayi, M.; Lucier, B.; Mahini, H.; and Seddighin, S. 2016. From duels to battefields: Computing equilibria of blotto and other games. *AAAI*.
- [3] Applegate, D., and Cohen, E. 2003. Making intra-domain routing robust to changing and uncertain traffic demands: understanding fundamental tradeoffs. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, 313–324. ACM.
- [4] Bazaraa, M.; Jarvis, J.; and Sherali, H. 2011. *Linear Programming and Network Flows*. Wiley.
- [5] Bellman, R. 1969. On colonel blotto and analogous games. *Siam Review* 11(1):66–68.
- [6] Bernhard, H.; Korte, B.; and Vygen, J. 2008. *Combinatorial optimization: Theory and algorithms*. Heidelberg: Springer-Verlag.
- [7] Blackett, D. W. 1954. Some blotto games. *Nav. Res. Logist. Q.* 1:55–60.
- [8] Blackett, D. W. 1958. Pure strategy solutions to blotto games. *Nav. Res. Logist. Q.* 5:107–109.
- [9] Borel, É., and Ville, J. 1938. Applications de la théorie des probabilités aux jeux de hasard. *Gauthier-Vilars*.
- [10] Borel, É. 1921. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes Rendus de l'Académie* 173(13041308):97–100.
- [11] Borel, É. 1953. The theory of play and integral equations with skew symmetric kernels. *Econometrica* 21:97–100.
- [12] Chowdhury, S. M.; Kovenock, D.; and Sheremeta, R. M. 2009. An experimental investigation of colonel blotto games. *Econ. Theor.* 1–29.
- [13] Cormen, T. 2009. *Introduction to Algorithms*. MIT Press.
- [14] Edmonds, J. 1971. Matroids and the greedy algorithm. *Mathematical programming* 1(1):127–136.
- [15] engadget. 2016. Game algorithm could help win elections. <http://www.engadget.com/2016/02/12/game-theory-algorithm/>.
- [16] EurokAlert. 2016. Umd-led team first to solve well-known game theory scenario. http://www.eurekalert.org/pub_releases/2016-02/uom-utf021116.php.
- [17] Fréchet, M. 1953a. Commentary on the three notes of emile borel. *Econometrica* 21:118–124.
- [18] Fréchet, M. 1953b. Emile borel, initiator of the theory of psychological games and its application. *Econometrica* 21:95–96.
- [19] Goemans, M. X. 2015. Smallest compact formulation for the permutahedron. *Mathematical Programming* 153(1):5–11.

- [20] Golman, R., and Page, S. E. 2009. General blotto: games of allocative strategic mismatch. *Public Choice* 138(3-4):279–299.
- [21] Gross, O. A., and Wagner, R. 1950. A continuous colonel blotto game. *RAND Corporation* RM-098.
- [22] Hart, S. 2008. Discrete colonel blotto and general lotto games. *International Journal of Game Theory* 36(3-4):441–460.
- [23] Insider, B. 2016. Scientists say they can predict two-party outcomes after solving a 95-year-old game theory problem. <http://www.businessinsider.com.au/scientists-say-they-can-predict-two-party-outcomes-after-solving-the-95-year-old-colonel-blotto-game-theory-problem-2016-2>.
- [24] Kovenock, D., and Roberson, B. 2010. Conflicts with multiple battlefields. CESifo Working Paper Series 3165, CESifo Group Munich.
- [25] 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.
- [26] Kvasov, D. 2007. Contests with limited resources. *J. Econ. Theory* 136(1):738–748.
- [27] Laslier, J.-F., and Picard, N. 2002. Distributive politics and electoral competition. *J. Econ. Theory* 103(1):106–130.
- [28] Martin, R. K. 1991. Using separation algorithms to generate mixed integer model reformulations. *Operations Research Letters* 10(3):119–128.
- [29] Merolla, J.; Munger, M.; and Tofias, M. 2005. In play: A commentary on strategies in the 2004 us presidential election. *Public Choice* 123(1-2):19–37.
- [30] Myerson, R. B. 1993. Incentives to cultivate favored minorities under alternative electoral systems. *Am. Polit. Sci. Rev.* 856–869.
- [31] MyInforms. 2016. Scientists say they can predict two-party outcomes after solving a 95-year-old game theory problem. <http://myinforms.com/en-au/a/24189185-scientists-say-they-can-predict-two-party-outcomes-after-solving-a-95-year-old-game-theory-problem/>.
- [32] Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge.
- [33] NSF. 2016. Umd-led team first to solve well-known game theory scenario. https://www.nsf.gov/news/news_summ.jsp?cntn_id=137734&org=NSF&from=news.
- [34] Roberson, B. 2006. The colonel blotto game. *Economic Theory* 29(1):1–24.
- [35] Rothvoß, T. 2013. Some 0/1 polytopes need exponential size extended formulations. *Mathematical Programming* 142(1-2):255–268.
- [36] Rothvoß, T. 2014. The matching polytope has exponential extension complexity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, 263–272. ACM.

- [37] Science Newslne. 2016. Umd-led team first to solve well-known game theory scenario. <http://www.sciencenewslne.com/summary/2016021212280019.html>.
- [38] ScienceDaily. 2016. Well-known game theory scenario solved: Colonel blotto: New algorithm could help political strategists, business leaders make better decisions. <https://www.sciencedaily.com/releases/2016/02/160211190010.htm?utm>.
- [39] Scientific Computing. 2016. After nearly a century, colonel blotto game theory scenario solved. <http://www.scientificcomputing.com/news/2016/02/after-nearly-century-colonel-blotto-game-theory-scenario-solved>.
- [40] Shubik, M., and Weber, R. J. 1981. Systems defense games: Colonel blotto, command and control. *Nav. Res. Logist. Q.* 28(2):281–287.
- [41] Tukey, J. W. 1949. A problem of strategy. *Econometrica* 17:73.
- [42] von Neumann, J. 1953. Communication on the borel notes. *Econometrica* 21:124–127.
- [43] Weinstein, J. 2005. Two notes on the blotto game. *Manuscript, Northwestern University*.
- [44] Yannakakis, M. 1988. Expressing combinatorial optimization problems by linear programs. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, 223–228. ACM.