Optimal Strategies of Blotto Games: Beyond Convexity

SOHEIL BEHNEZHAD*, University of Maryland
 AVRIM BLUM, TTI-Chicago
 MAHSA DERAKHSHAN*, University of Maryland
 MOHAMMADTAGHI HAJIAGHAYI*, University of Maryland
 CHRISTOS H. PAPADIMITRIOU, Columbia University
 SAEED SEDDIGHIN*, University of Maryland

The *Colonel Blotto* game, first introduced by Borel in 1921, is a well-studied game theory classic. Two colonels each have a pool of *troops* that they divide simultaneously among a set of *battlefields*. The winner of each battlefield is the colonel who puts more troops in it and the overall utility of each colonel is the sum of weights of the battlefields that s/he wins. Over the past century, the Colonel Blotto game has found applications in many different forms of competition from advertisements to politics to sports.

Two main objectives have been proposed for this game in the literature: (i) maximizing the guaranteed expected payoff, and (ii) maximizing the probability of obtaining a minimum payoff u . The former corresponds to the conventional utility maximization and the latter concerns scenarios such as elections where the candidates' goal is to maximize the probability of getting at least half of the votes (rather than the expected number of votes). In this paper, we consider both of these objectives and show how it is possible to obtain (almost) optimal solutions that have few strategies in their support.

One of the main technical challenges in obtaining bounded support strategies for the Colonel Blotto game is that the solution space becomes non-convex. This prevents us from using convex programming techniques in finding optimal strategies which are essentially the main tools that are used in the literature. However, we show through a set of structural results that the solution space can, interestingly, be partitioned into polynomially many disjoint convex polytopes that can be considered independently. Coupled with a number of other combinatorial observations, this leads to polynomial time approximation schemes for both of the aforementioned objectives.

We also provide the first complexity result for finding the maximin of Blotto-like games: we show that computing the maximin of a generalization of the Colonel Blotto game that we call **GENERAL COLONEL BLOTTO** is exponential time-complete.

ACM Reference Format:

Soheil Behnezhad, Avrim Blum, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Christos H. Papadimitriou, and Saeed Seddighin. 2019. Optimal Strategies of Blotto Games: Beyond Convexity. In *ACM EC '19: ACM Conference on Economics and Computation (EC '19), June 24–28, 2019, Phoenix, AZ, USA*. ACM, New York, NY, USA, 20 pages. <https://doi.org/10.1145/3328526.3329608>

Full Version Available on arXiv: [3]

*Supported in part by Guggenheim Fellowship, NSF grants CCF:SPX 1822738, IIS:BIGDATA 1546108, DARPA grant SI3CMD, UMD Year of Data Science Program Grant, and Northrop Grumman Faculty Award.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

EC '19, June 24–28, 2019, Phoenix, AZ, USA

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6792-9/19/06...\$15.00

<https://doi.org/10.1145/3328526.3329608>

1 INTRODUCTION

The *Colonel Blotto* game, originally introduced by Borel in 1921 [5, 6, 9, 10], is among the first mathematically formulated strategic situations and has become one of the game theory classics over the past century. In this game, two players — which we refer to as player 1 and player 2 — are fighting over a set of *battlefields*. Each player has a number of *troops* to distribute across the battlefields and the winner of each of the battlefields is the player who puts more troops in it. The overall utility of each player is the sum of weights of the battlefields that s/he wins. Although originally introduced to consider a war situation (as the terminology implies,) the Colonel Blotto game has found applications in many different forms of competition from advertisements to politics to sports [7, 16–18, 20, 21].

The goal in the Colonel Blotto game is to find optimal (i.e., *maximin*) strategies of the players. A mixed strategy is maximin if it maximizes the guaranteed *expected* utility against any strategy of the opponent. For certain applications of the Colonel Blotto game — such as the United States presidential elections, which we elaborate on in the next paragraph — maximizing the expected utility is not desirable. In such scenarios, we need to maximize the probability of obtaining a minimum utility u . This has been captured by the concept of (u, p) -maximin strategies in the literature [2]. A strategy is (u, p) -maximin if it guarantees receiving a utility of at least u with probability at least p . In this paper we consider both of these objective functions.

Perhaps the most notable application of the Colonel Blotto game is in the U.S. presidential election race. Every state has a number of electoral votes and the winning candidate is the one who receives the majority of these votes. In most¹ of the states, a *winner-take-all* rule determines the electoral votes. That is, the candidate who gets the popular vote in a state receives all the electoral votes of that state. It is quite possible that the winning candidate loses the popular vote over the country; thus, the candidates have to strategically decide on how they spend their resources such as funds, staff, etc., in different states. This can be modeled by a Colonel Blotto instance in which each battlefield corresponds to a state and the colonels' troops correspond to the candidates' resources. Here, maximizing the expected utility (i.e., the expected number of electoral votes) does not necessarily maximize the likelihood of winning the race. Instead, the actual goal is to maximize the probability of obtaining more than half (i.e., at least 270) of the electoral votes. As such, we need to find a $(270, p)$ -maximin strategy with maximum possible p .

Maximin strategies, or equivalently Nash equilibrium for constant-sum games such as Colonel Blotto, are often criticized for that they may be too complicated (see e.g., [19, 24, 26]). That is, even if we are able to find such solution in polynomial time, we may not be able to deploy it since the equilibria can have a large support². In the case of the Blotto game, the potential size of the support is enormous, while every possible pure strategy in the support requires a prior (often costly) setup. Therefore it is tempting to find a near-optimal strategy that uses very few pure strategies, and is near optimum against the opponent's best response — the main goal of the present paper.

However, limiting the support size often renders the game intractable. For instance, the decision problem of existence of a Nash equilibrium when the support size is bounded by a given number is NP-hard even in two-player zero-sum games [11], while this problem is unlikely to be fixed parameter tractable when the problem is parametrized by the support size [8]. These results imply that in order to find optimal strategies with bounded support, we need to use structural properties of the game at hand, even when the players have polynomially many strategies. It becomes even more challenging for succinct games such as Colonel Blotto wherein the strategy space itself is exponentially large.

¹All states except Maine and Nebraska.

²The *support* of a mixed strategy is the set of all pure strategies to which a non-zero probability is associated.

Recent studies have made significant progress in understanding the optimal response problem in Colonel Blotto when the support size is unbounded [1, 2, 4, 12–14, 22, 23, 25]. Note that the unbounded case is also challenging to solve due to the exponential number of pure strategies of the players in the Colonel Blotto game. In spite of that, for maximizing the expected utility, one can obtain optimal strategies in polynomial time by algorithms of Ahmadinejad *et al.* [1] and Behnezhad *et al.* [4]. The case of (u, p) -maximin strategies is generally harder to solve. However, Behnezhad *et al.* [2] show that it is possible to obtain a logarithmic factor approximation in polynomial time.

All of the results mentioned above rely crucially on the fact that the support size is unbounded. The challenges in obtaining bounded support strategies turn out to be entirely different. On one hand, for the choice of each pure strategy in the support we still have exponentially³ many possibilities. On the other hand, we show that bounding the support size makes the solution space non-convex. The latter prevents us from using convex programming techniques in finding optimal strategies — which are essentially the main tools that are used in the literature for solving succinct games in polynomial time [1, 2, 4, 15]. However, we show through a set of structural results that the solution space can, interestingly, be partitioned into polynomially many disjoint convex polytopes, allowing us to consider each of these sub-polytopes independently. Combined with a number of other techniques that are highlighted in Section 3, this leads to polynomial time approximation schemes (PTASs) for both the expectation case and the case of (u, p) -maximin strategies.

We also provide the first complexity result for finding the maximin of Blotto-like games: we show that computing the maximin of a generalization of the Colonel Blotto game that we call GENERAL COLONEL BLOTTO — roughly, the utility is a general function of the two allocations — is exponential time-complete.

2 PRELIMINARIES

Throughout the paper, for any integer n , we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. We denote the vectors by bold fonts and for every vector $\mathbf{v}$, use v_i to denote its i th entry.

The Colonel Blotto game. The Colonel Blotto game is played between two players which we refer to as player 1 and player 2. Any instance of the game can be represented by a tuple $\mathcal{B}(n, m, \mathbf{w})$ where n and m respectively denote the number of *troops* of player 1 and player 2, and $\mathbf{w} = (w_1, \dots, w_k)$ is a vector of length k of positive integers denoting the *weight* of k *battlefields* on which the game is being played.

A pure strategy of each of the players is a partitioning of his troops over the battlefields. That is, any pure strategy of player 1 (resp. player 2) can be represented by a vector $\mathbf{x} = (x_1, \dots, x_k)$ of length k of non-negative numbers where $\sum_{i \in [k]} x_i \leq n$ (resp. $\sum_{i \in [k]} x_i \leq m$). In the *discrete* variant of the game, the number of troops that are assigned to each of the battlefields must be non-negative integers. In contrast, in the *continuous* variant, any assignment with non-negative real values is considered valid. Throughout the paper, we denote respectively by $\mathcal{S}_1$ and $\mathcal{S}_2$ the set of all valid pure strategies of player 1 and player 2.

Let $\mathbf{x}$ and $\mathbf{y}$ be the pure strategies that are played respectively by player 1 and player 2. Player 1 wins battlefield i if $x_i > y_i$ and loses it otherwise. The winner of battlefield i gets a partial utility of w_i on that battlefield, and the overall utility of each player is the sum of his partial utilities on all the battlefields. More precisely, the utilities of players 1 and 2, which we respectively denote by

³Or even an unbounded number of strategies for the continuous variant of Colonel Blotto where the troops are capable of fractional partitioning.

$u_1(\mathbf{x}, \mathbf{y})$ and $u_2(\mathbf{x}, \mathbf{y})$, are as follows:

$$u_1(\mathbf{x}, \mathbf{y}) = \sum_{i \in [k]: x_i > y_i} w_i, \quad u_2(\mathbf{x}, \mathbf{y}) = \sum_{i \in [k]: x_i \leq y_i} w_i.$$

Note that in the definition above, we break the ties in favor of player 2, i.e., when both players put the same number of troops on a battlefield, we assume the winner is player 2.

Also, we define the *uniform* Colonel Blotto game to be a special case of Colonel Blotto in which all of the battlefields have the same weights, i.e., $w_1 = w_2 = \dots = w_k = 1$.

The objective. The guaranteed expected utility of a (possibly mixed) strategy $\mathbf{x}$ of player 1 is u , if against any (possibly mixed) strategy $\mathbf{y}$ of player 2, we have $\mathbb{E}_{\mathbf{x}' \sim \mathbf{x}, \mathbf{y}' \sim \mathbf{y}}[u_1(\mathbf{x}', \mathbf{y}')] \geq u$. A strategy is *maximin* if it maximizes this guaranteed expected utility.

In this paper, we are interested in bounded *support* strategies. The support of a mixed strategy is the set of pure strategies to which it assigns a non-zero probability. We call a strategy c -strategy if its support has size at most c . A maximin c -strategy is a c -strategy that has the highest guaranteed expected utility among all c -strategies. Observe that we do not restrict the adversary to play a c -strategy here. In fact, given the mixed strategy of a player, it is well-known that the best response of the opponent can be assumed to be a pure strategy w.l.o.g. This means that an opponent that can respond by only one pure strategy is as powerful as an opponent that can play mixed strategies as far as maximin strategies are concerned.

Another standard objective for many natural applications of the Colonel Blotto game is to maximize the probability of obtaining a utility of at least u . This has been captured in the literature by the notion of (u, p) -maximin strategies.

DEFINITION 2.1. For any two-player game, a (possibly mixed) strategy $\mathbf{x}$ of player 1 is called a (u, p) -maximin strategy, if for any (possibly mixed) strategy $\mathbf{y}$ of player 2,

$$\Pr_{\mathbf{x}' \sim \mathbf{x}, \mathbf{y}' \sim \mathbf{y}}[u(\mathbf{x}', \mathbf{y}') \geq u] \geq p.$$

In such scenarios, for a given minimum utility u , our goal is to compute (or approximate) a (u, p) -maximin with maximum possible p . Similar to the expected case, we are interested in bounded support strategies. That is, given u and c , our goal is to compute (or approximate) a (u, p) -maximin c -strategy with maximum possible p . Again, we do not restrict the adversary's strategy.

3 HIGH-LEVEL OVERVIEW OF OUR RESULTS & TECHNIQUES

Throughout this paper, we discuss the optimal strategies of the Colonel Blotto game when the support size is small. That is, when player 1 can only randomize over at most $c = O(1)$ pure strategies and wishes to maximize his utility. In this section, we give an overview of our results and the techniques we use to obtain them. It should be noted that the discussions in this section are purposefully informal and intuitive. More details are provided in the forthcoming sections and the full version of the paper [3].

Our main interest is in finding (almost) optimal (u, p) -maximin strategies as discussed above. However, our results carry over to the conventional definition of the maximin strategies and can be used when the goal is to maximize the expected utility.

When randomization is not allowed, any (u, p) -maximin 1-strategy with $p > 0$ is also $(u, 1)$ -maximin. Thus, the problem boils down to finding a pure strategy with the maximum guaranteed payoff. However, when randomization over two pure strategies is allowed, we may play two pure strategies with different probabilities q and $1 - q$ and thus the problem of finding a (u, p) -maximin 2-strategy with $p \neq 1$ does not necessarily reduce to the case where the goal is to find a single pure strategy. As an example, when $n = m = 2$ and we have two battlefields with equal weights ($w_1 = w_2 = 1$), a

$(1, \frac{1}{2})$ -maximin 2-strategy can be obtained by selecting a battlefield at random and placing two troops on the selected battlefield. However, in this example (or in any example with $m = n$), no 2-strategy is $(1, p)$ -maximin for $p > \frac{1}{2}$ since the opponent can just select our higher-probability strategy and copy it, ensuring we get utility 0 with probability at least $\frac{1}{2}$. More generally, for $c = 2$, even if $n \leq m$, a simple observation shows that the only interesting cases are when $p = 1$ (which reduces to the $c = 1$ case) or when $p = \frac{1}{2}$. The idea is that when $p > \frac{1}{2}$ holds, existence of a (u, p) -maximin 2-strategy for an arbitrary u implies that of a $(u, 1)$ -maximin 2-strategy. Similarly, if a 2-strategy is (u, p) -maximin for some u and $0 < p < \frac{1}{2}$, one can modify the same strategy to make it $(u, \frac{1}{2})$ -maximin. Therefore, for $c = 2$, the problem is (computationally) challenging only when a $(u, \frac{1}{2})$ -maximin 2-strategy is desired. Thus, for $c = 2$, the problem essentially reduces to finding two pure strategies $\mathbf{x}, \mathbf{x}'$ such that no strategy of the opponent can prevent *both* $\mathbf{x}$ and $\mathbf{x}'$ from getting a utility of at least u . A similar, but more in-depth analysis gives us the same structure for $c = 3$. That is, in this case, we may look for a $(u, \frac{1}{3})$ -maximin or a $(u, \frac{2}{3})$ -maximin 3-strategy. This implies that in an optimal solution, we look for three strategies $\mathbf{x}, \mathbf{x}', \mathbf{x}''$ such that no strategy of the opponent prevents two of or all of (depending on whether $p = \frac{1}{3}$ or $p = \frac{2}{3}$) $\mathbf{x}, \mathbf{x}'$ and $\mathbf{x}''$ from getting a utility of at least u .

It is surprising to see that this structure breaks when considering more than three pure strategies ($c \geq 4$). For instance, consider an instance of the Colonel Blotto game with 4 battlefields ($k = 4$) in which the players have 4 and 6 troops, respectively ($n = 4, m = 6$). Let the weights of the first 3 battlefields be 5 and the weight of the last battlefield be 10. In this example, the goal of player 1 is to obtain a utility of at least 10 with the highest probability. One can verify with an exhaustive search that if player 1 were to randomize over 4 pure strategies with equal probability, he could guarantee a utility of at least 10 with probability no more than $\frac{1}{4}$.⁴ However, we present in Table 1, a $(10, \frac{2}{5})$ -maximin 4-strategy for player 1 that plays 4 pure strategies with non-uniform probabilities.

	Battlefield 1 $w_1 = 5$	Battlefield 2 $w_2 = 5$	Battlefield 3 $w_3 = 5$	Battlefield 4 $w_4 = 10$
Strategy 1 , played with probability $\frac{1}{4}$.	0	0	0	4
Strategy 2 , played with probability $\frac{1}{4}$.	1	1	2	0
Strategy 3 , played with probability $\frac{1}{4}$.	1	2	1	0
Strategy 4 , played with probability $\frac{1}{4}$.	2	1	1	0

Table 1. A $(10, \frac{2}{5})$ -maximin 4-strategy for player 1 on instance $\mathcal{B}(4, 6, (5, 5, 5, 10))$.

In order to design an algorithm for finding (u, p) -maximin strategies that does not lose on p , it is essential to understand how player 1 randomizes over his strategies in an optimal solution. We begin by showing that when randomization is allowed only on a small number of pure strategies, the number of different ways to distribute the probability over the pure strategies in an optimal solution is limited. That is, one can list a number of probability distributions and be sure that at least one of such probability distributions leads to an optimal solution. This structural property is general and applies to any game so long as (u, p) -maximin strategies are concerned.

THEOREM 3.1 (INFORMAL STATEMENT). *When randomization is only allowed on a constant number of pure strategies, the number of possible probability distributions for an optimal (u, p) -maximin strategies is limited by a constant.*

⁴We have verified this claim with a computer program.

Theorem 3.1 is proven via a combinatorial analysis of the optimal solutions. On one hand, we leverage the optimality of the solution to argue that p cannot be improved. On the other hand, we use the maximality of p to derive relations between the probabilities that the pure strategies are played with. Finally, we use these relations to narrow down the probabilities a to a small set.

Indeed, a consequence of Theorem 3.1 is that when we fix a utility u and wish to find a (u, p) -maximin c -strategy with highest p , p can only take a constant number of values. This observation makes the problem substantially easier as we can iterate over all possible probability profiles and solve the problem separately for each profile. Thus, from now on, we assume that we fix a probability distribution over the pure strategies, and the goal is to select c strategies to be played according to the fixed probabilities. We assume that (i) u and p are given in the input, (ii) we are guaranteed that a (u, p) -maximin c -strategy exists, and (iii) the goal is to compute/approximate such a strategy.

Continuous Colonel Blotto. For the continuous case, we can represent each pure strategy with a vector of size k of real values indicating the number of troops that is placed on each battlefield. Thus, a c -strategy can be represented by c vectors of length k (given that we have fixed the probability distribution over the pure strategies). A simple observation (also pointed out by Behnezhad *et al.* [2]) is that when the goal is to find a pure maximin strategy, the solution is essentially a convex set with respect to the representations. That is, if $\mathbf{x}^1$ and $\mathbf{x}^2$ are both (u, p) -maximin, any strategy whose representation is a convex combination of the representations of $\mathbf{x}^1$ and $\mathbf{x}^2$ is also (u, p) -maximin. The pure maximin problem coincides with our setting when $c = 1$. However, it is surprising to see that for $c > 2$, the solution may not be convex, even though we use the same approach to represent the strategies.

As an example, imagine we have two battlefields with equal weights (say 1) and $n = m = 2$. Indeed a $(1, 1/2)$ -maximin strategy can be obtained for player 1 by randomizing over $(0, 2)$ and $(2, 0)$ uniformly. Similarly, randomizing over $(2, 0)$ and $(0, 2)$ (the order is changed) gives us the same guarantee. However, a convex combination of the two strategies plays $(1, 1)$ deterministically and loses both battlefields against the strategy $(1, 1)$ of player 2. The situation may be even worse as one can construct a delicate instance whose solution set is the union of up to $2^{\Omega(k)}$ convex regions no two of which make a convex set when merged.

A key observation that enables us to compute/approximate the solution is the *partial convexity* of the solution. That is, we show that although the solution set is not necessarily convex, one can identify regions of the space where for each region, the solution is convex. To be more precise, denote by $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^c$ the representations of the strategies. In this representation, x_j^i denotes the number of troops that i 'th pure strategy places on the j 'th battlefield. This gives us a ck dimensional problem space $[0, n]^{ck}$. Divide this space by $k \binom{c}{2}$ hyperplanes each formulated as $x_j^i = x_j^{i'}$ for some $i, i' \in [c]$ and $j \in [k]$. Partial convexity implies that the solution space in each region is convex, and as a result, gives us an exponential time solution for the problem in the continuous setting. Roughly speaking, since the solution is convex in each region, we can iterate over all regions and solve the problem separately using a linear program for each region. However, we have exponentially many regions and therefore the running time of this approach is exponentially large.

The above algorithm can be modified to run in polynomial time in the uniform setting. The high-level idea is that when the battlefields have equal weights, there is a strong symmetry between the solutions of the regions. Based on this, we show that in the uniform setting, one only needs to search a polynomial number of regions for a solution. This idea along with the partial convexity gives us a polynomial time solution for the uniform setting.

Fig. 1. An example of how the solution space is decomposed for the case when $k = 3$ and $c = 2$. The figure on the left illustrates 3 hyperplanes $x_j^1 = x_j^2$. These hyperplanes divide the solution space into 8 regions and for all the points in each region, as illustrated in the figure on right, x_j^1 and x_j^2 for $j \in [3]$ compare in the same way.

THEOREM 3.2 (INFORMAL STATEMENT). *There exists a polynomial time solution for finding a (u, p) -maximin c -strategy for Colonel Blotto in the continuous case when all the battlefields have the same weight.*

Indeed, the uniform setting is a very special case since we are essentially indifferent to the battlefields. When we incorporate the weights of the battlefields, we no longer expect the solutions to be symmetric with respect to the battlefields. However, one may still observe a weak notion of symmetry between the regions. Recall that we denote the weights of the battlefields with $w_1, w_2, \dots, w_k$. Let us lose a factor of $1 + \epsilon$ in the utility and assume w.l.o.g. that each weight w_i is equal to $(1 + \epsilon)^j$ for some integer $j \geq 0$. Assuming that the maximum weight is polynomially bounded (we only make this assumption for the sake of simplicity and our solution does not depend on this constraint), the number of different battlefield weights is logarithmically small. Thus, we expect many battlefield to have equal weights which as a result makes the solution regions more symmetric. However, this idea alone gives us a quasi-polynomial time algorithm for searching the regions as we may have a logarithmic number of different battlefield weights. To reduce the running time to polynomial, we need to further prune the regions of the solution to polynomially many. Indeed, we show that it suffices to search over a polynomial number of regions if we allow an approximate solution. Via this observation, we can design a polynomial time algorithm that approximates the solution within a factor $1 + \epsilon$. This settles the problem for the continuous setting.

THEOREM 3.3 (INFORMAL STATEMENT). *(u, p) -maximin c -strategy strategies of the Colonel Blotto admit a PTAS in the continuous setting.*

Discrete Colonel Blotto. For the discrete setting, we take a rather different approach. The main reason is that in this setting, even if we are guaranteed that the solution is convex, we cannot use LP's to compute/approximate a solution. Behnezhad *et al.* [2] give a 2 approximation algorithm that finds a pure maximin strategy for the discrete setting. Indeed, this solution can be used to get a 2 approximate solution for the case of $c = 1$. We both extend their algorithm to work for $c \geq 2$ and also devise a *heavy-light decomposition* to improve the approximation factor to $1 + \epsilon$. Both our extension and decomposition techniques are novel.

Behnezhad *et al.* [2] introduce the notion of a *weak adversary*. Roughly speaking, they define a relaxed best response for player 2 that does not maximize the utility of player 2, but instead, approximately maximizes his utility. We call a player that plays a relaxed best response, a weak adversary. By proposing a greedy algorithm for the weak adversary, Behnezhad *et al.* [2] show that the payoff of the weak adversary and the actual adversary differ by the value of at most one battlefield. That is, if the weights of the battlefields are bounded by $w_{\max}$, then the difference

between the utility of an adversary and that of a weak adversary is always bounded by $w_{\max}$. Next, they show that a dynamic program can find a pure strategy of player 1 that performs best against a relaxed adversary and turn this algorithm into a 2 approximate solution for the problem, by considering two cases $2w_{\max} \geq u$ and $2w_{\max} < u$.

Indeed losing an additive error of $w_{\max}$ may hurt the approximation factor a lot when the desired utility u is not much more than $w_{\max}$. Thus, in order to improve the approximation factor, one needs to design a separate algorithm for the heavy battlefields. To this end, we introduce our heavy-light decomposition. We define a threshold $\tau \approx \epsilon u$ and call a battlefield i heavy if $w_i > \tau$ and light otherwise. In addition to this, we assume w.l.o.g. that $w_{\max} \leq u$ since otherwise one can set a cap of u on the weights without changing the solution. Therefore, the maximum weight and the minimum weight of heavy battlefields are within a multiplicative factor of $1/\epsilon$. Next, by incurring an additional $1 + \epsilon$ multiplicative factor to the approximation guarantee, we round down the weight of each battlefield to the nearest $(1 + \epsilon)^i$. We show that this leaves us with a constant number of different weights for the heavy battlefields. Next, we state that since the number of different weights for the heavy battlefields is constant, the total number of (reasonable) pure strategies of player 1 over these battlefields can be reduced down to a polynomial. Indeed this also holds for player 2, but for the sake of our solution, we should further bound the number of responses of player 2 over these battlefields. We show in fact, that the number of (reasonable) responses of player 2 over the heavy battlefields against a strategy of player 1 is bounded by a constant! For light battlefields, we use the idea of a weak adversary. However, in order to find a solution that considers both heavy and light battlefields, we need to define multiple weak adversaries each with regard to a response of player 2 on the heavy battlefields.

Let us clarify the challenge of mixing the two solutions via an example. Suppose we have 4 battlefields with weights 10, 8, 7 and 5 and the players' troops are as follows: $n = 5$ and $m = 2$. One can verify that in this case, the following pure strategy of player 1 guarantees him a payoff of 15.

	Battlefield 1 $w_1 = 10$	Battlefield 2 $w_2 = 8$	Battlefield 3 $w_3 = 7$	Battlefield 4 $w_4 = 5$
A (15, 1)-maximin 1-strategy for player 1.	2	2	1	0

Table 2. A (15, 1)-maximin 1-strategy for player 1 on instance $\mathcal{B}(5, 2, (10, 8, 7, 5))$.

In fact, 15 is the highest utility player 1 can get with a single pure strategy as no other pure strategy of player 1 can guarantee a payoff more than 15 for him. Now, assume that we select the first two battlefields with weights 10 and 8 as heavy battlefields and the rest of the battlefields as the light ones. One may think that by taking a maximin approach for the heavy battlefields and solving the problem separately for the light battlefields, we can obtain a correct solution. The above example shows that this is not the case. We show in what follows, that the maximin approach reports a payoff of 17 for player 1 which is more than the actual solution. Fix the strategy of player 1 on the heavy battlefields to be placing 2 troops on battlefield 1 and 1 troop on battlefield 2. As such, the only reasonable responses of player 2 on these battlefields are as shown below.

Response 3 already gives player 1 a payoff of 18 which is more than 17. Also, response 1 of player 2 leaves him with no troops for the light battlefields and thus he loses both light battlefields against strategy (1, 1) of player 1 on the light battlefields. Therefore, this gives player 1 a payoff of 20. Also, if player 2 plays response 2 on the heavy battlefields, player 1 can win the light battlefield w_3 by putting two troops on it. Indeed player 2 has only one troop left and there is no way for him to win this battlefield. Thus, in this case, the payoff of player 1 would be 17. Since we take the maximum solution over all strategies of player 1 for the heavy battlefields, our final utility would be at least 17.

	Battlefield 1 $w_1 = 10$	Battlefield 2 $w_2 = 8$	troops left for player 2
response 1	2	0	0
response 2	0	1	1
response 3	0	0	2

Table 3. The responses of player 2 on the heavy battlefields.

What the above analysis shows is that, if we take a maximin approach on the heavy battlefields and then solve (or approximate) the problem for the light battlefields, we may incorrectly report a higher payoff for player 1. Roughly speaking, this error happens since in this approach, we allow player 1 to have different actions over the light battlefields against different responses of player 2. To resolve this issue, we design a dynamic program that takes into account all responses of player 2 simultaneously. Indeed, to make sure the program can be solved in polynomial time, we need to narrow down the number of responses of player 2 to a constant. Our heavy-light decomposition along with our structural properties of the optimal strategies enables us to reach this goal. This gives us a non-trivial dynamic program that can approximate the solution within a factor $1 + \epsilon$ for the case of a single pure strategy.

THEOREM 3.4 (INFORMAL STATEMENT). *The problem of finding a $(u, 1)$ -maximin strategy for player 1 in discrete Colonel Blotto admits a PTAS.*

To extend the result to the case of $c \geq 2$, we need to design a weak adversary that plays a relaxed best response against more than 1 strategy of player 1. For $c = 1$, the greedy algorithm follows from the well-known greedy solution of knapsack. However, when $c \geq 2$ the best-response problem does not necessarily reduce to knapsack and therefore our greedy solution is much more intricate. Roughly speaking, we design a non-trivial procedure for player 2 that gets c thresholds as input, and based on these thresholds, decides about the response for each battlefield locally. This local decision making is a key property that we later exploit in our dynamic program to find an optimal strategy against a weak adversary. This in addition to the heavy-light decomposition technique gives us a PTAS for (u, p) -maximin c -strategy strategies of Colonel Blotto in the discrete setting.

THEOREM 3.5 (INFORMAL STATEMENT). *The problem of finding a (u, p) -maximin c -strategy for player 1 in discrete Colonel Blotto admits a PTAS.*

Further results. Our techniques also imply PTASs for the Colonel Blotto game when instead of a (u, p) -maximin c -strategy we wish to find a maximin c -strategy. For the continuous case, similar to the case of (u, p) -maximin strategies, we divide the solution space into polynomially many convex subregions and prove that among them a $(1 + \epsilon)$ -approximate solution is guaranteed to exist. The main difference with the case of (u, p) -maximin strategies is in the LP formulation of the problem, but the general approach is essentially the same. For the discrete variant of Colonel Blotto, we also follow a similar approach as in the case of (u, p) -maximin strategies. In more details, we partition the battlefields into heavy and light subsets and define a weaker adversary that is adapted to approximately best respond against maximin strategies. We find it surprising and possibly of independent interest that essentially the same approach (though with minor changes) can be applied to these two variants of Colonel Blotto. Prior algorithms proposed for these two variants were fundamentally different [1, 2, 4].

THEOREM 3.6 (INFORMAL STATEMENT). *The problem of finding a maximin c -strategy for player 1 in both discrete and continuous variants of Colonel Blotto admits a PTAS.*

Finally, recall that motivation for approximate algorithms comes from intractability. In view of all the recent sophisticated algorithmic approaches to solving, approximately, various special cases of the Colonel Blotto game, it is worth asking, what is the computational complexity of the full fledged problem of computing a maximin strategy of the Colonel Blotto game? (Notice that, since the full strategy is too long to return, we should formulate the problem in terms of something succinct, for example *one* component of the maximin.) We present the first complexity results in this area, establishing that an interesting variant of the problem that we call **GENERAL COLONEL BLOTTO** — roughly, the utility is a general function of the two allocations, instead of the probability of winning more than a certain goal of total battlefield weight — is *complete for exponential time*. The precise complexity of the two versions of the original game (computing a maximin of the probability of winning a majority, or of the expectation of the total weights of battlefields won) is left as an open question here. We conjecture that both problems are also exponential time-complete.

4 CONTINUOUS COLONEL BLOTTO

In this section we consider the continuous variant of Colonel Blotto. We start with the case where our goal is to find a (u, p) -maximin 1-strategy and show how we can generalize it to 2-strategies and, further, to any bounded number of strategies.

For the particular case of $c = 1$, our goal is to find a single pure strategy that is (u, p) -maximin. Indeed, since we are playing a single pure strategy with no randomization, the probability p must be 1. One can think of the solution of this case as a vector of non-negative real values that sum up to n and formulate the problem in the following way.

$$\begin{array}{ll}
 \text{find} & \mathbf{x} \\
 \text{subject to} & x_i \geq 0 \quad \forall i \in [k] \\
 & \sum x_i \leq n \\
 & u_1(\mathbf{x}, \mathbf{y}) \geq u \quad \forall \mathbf{y} \in \mathcal{S}_2
 \end{array} \tag{1}$$

To analyze Program 1, we need to better understand the payoff constraints. To this end, we state another interpretation of the payoff constraints in Observation 4.1.

OBSERVATION 4.1 ([2]). *A pure strategy $\mathbf{x}$ of player 1 guarantees a payoff of at least u against any pure strategy of player 2 if and only if $\sum_{i \in S} x_i > m$ for any set S of battlefields with $\sum_{i \notin S} w_i < u$.*

PROOF. ($\Rightarrow$): Suppose that a strategy $\mathbf{x}$ does not get a payoff of u against a strategy $\mathbf{y}$ of player 2. This means that there exists a set S of battlefields that player $\mathbf{x}$ loses. The total payoff of $\mathbf{x}$ is below u , and therefore $\sum_{i \notin S} w_i < u$. In addition to this $\sum_{i \in S} x_i \leq m$ holds since player 2 needs to match player 1's troops in all battlefields of S .

($\Leftarrow$): It is trivial to show that if there exists such a violating set S , then $\mathbf{x}$ cannot promise a payoff of u against any strategy of player 2. The reason is that since $\sum_{i \in S} x_i \leq m$ player 2 can match the troops of player 1 in set S and that suffices to prevent player 1 from winning a payoff of u . $\square$

Via Observation 4.1, we can turn Program 1 into LP 2 where we have a constraint for every possible subset S of battlefields with $\sum_{i \notin S} w_i < u$. Although the number of these subsets can be exponentially large, Behnezhad *et al.* [2] show that one can find a violating constraint of LP 2 in polynomial time and thus find a feasible solution using the ellipsoid method.

$$\begin{array}{ll}
\text{find} & \mathbf{x} \\
\text{subject to} & x_i \geq 0 \quad \forall i \in [k] \\
& \sum x_i \leq n \\
& \sum_{i \in S} x_i > m \quad \text{for every subset } S \text{ of battlefields with } \sum_{i \notin S} w_i < u.
\end{array} \tag{2}$$

The key idea that enables us to solve this variant is that we are playing only one pure strategy. Even the case of having two pure strategies in the support is much more challenging. To illustrate the challenges and ideas, we next focus on how to obtain a 2-strategy and later generalize it to c -strategies.

4.1 The Case of 2-Strategies

Recall that if a 2-strategy is (u, p) -maximin for some $p > 1/2$, then there also exists a pure $(u, 1)$ -maximin strategy that, by aforementioned techniques, can be found in polynomial time. Further, if a 2-strategy is (u, p) -maximin for some $p < 1/2$, we can simply play the two strategies in its support with equal probability $1/2$ to obtain a better $(u, 1/2)$ -maximin strategy. Combining these two observations, we can assume w.l.o.g., that in the case of $c = 2$, we wish to find two pure strategies $\mathbf{x}$ and $\mathbf{x}'$, and play them with equal probabilities such that any strategy of player 2 gives us a payoff of at least u for at least one of $\mathbf{x}$ or $\mathbf{x}'$. A mathematical formulation of the problem is given below.

$$\begin{array}{ll}
\text{find} & \mathbf{x} \text{ and } \mathbf{x}' \\
\text{subject to} & x_i \geq 0 \text{ and } x'_i \geq 0 \quad \forall i \in [k] \\
& \sum x_i \leq n \\
& \sum x'_i \leq n \\
& \text{either } u_1(\mathbf{x}, \mathbf{y}) \geq u \text{ or } u_1(\mathbf{x}', \mathbf{y}) \geq u \quad \forall \mathbf{y} \in \mathcal{S}_2
\end{array} \tag{3}$$

Observe that the fourth constraint of the above program is not linear. In the following, we show that even the polytope that is described by this program is essentially nonconvex.

OBSERVATION 4.2. *Program 3 is not convex.*

PROOF. Suppose $n = m = 2$, $w_1 = w_2 = 1$ and the goal is to find a $(1, 1/2)$ -maximin strategy by randomizing over two pure strategies. A possible solution is to play $\mathbf{x} = (2, 0)$ with probability $1/2$ and play $\mathbf{x}' = (0, 2)$ with probability $1/2$ which guarantees a payoff of 1 with probability $1/2$. An alternative way to achieve this goal is to set $\mathbf{x} = (2, 0)$ and $\mathbf{x}' = (0, 2)$ which is the same strategy except that $\mathbf{x}$ and $\mathbf{x}'$ are exchanged. However, the linear combination of the two strategies results in $\mathbf{x} = (1, 1)$ and $\mathbf{x}' = (1, 1)$ which always loses both battlefields against $\mathbf{y} = (1, 1)$. $\square$

A more careful analysis shows that the feasible region of Program 3 may be the union of up to 2^k convex polytopes which makes it particularly difficult to find a desired solution. In what follows, we present algorithms to overcome this challenge for both the uniform and nonuniform settings.

4.1.1 Uniform Setting. Recall that in order to find a pure $(u, 1)$ -maximin strategy, we proved the linearity of Program 1 by characterizing the optimal solution. Similar to that, we show necessary and sufficient conditions for the solution of Program 3. To this end, we define *critical tuples* as follows.

DEFINITION 4.3. *Let L_1 , L_2 , and L_{12} be three disjoint subsets of battlefields. We call the tuple $\langle L_1, L_2, L_{12} \rangle$ a critical tuple, if critical any of $L_1 \cup L_{12}$ or $L_2 \cup L_{12}$ suffices to prevent player 1 from getting a payoff of u . In other words, $\langle L_1, L_2, L_{12} \rangle$ is a critical tuple, if and only if*

$$\sum_{i \notin L_1 \cup L_{12}} w_i < u, \quad \text{and}, \quad \sum_{i \notin L_2 \cup L_{12}} w_i < u.$$

Via this definition, we can now describe the feasible solutions of Program 3 as follows.

OBSERVATION 4.4. *Two pure strategies $\mathbf{x}$ and $\mathbf{x}'$ of player 1 meet the constraints of Program 3 if and only if for any critical tuple $\langle L_1, L_2, L_{12} \rangle$ we have*

$$\sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} \max\{x_i, x'_i\} > m.$$

PROOF. The proof is similar to that of Observation 4.1. Note that $\mathbf{x}$ and $\mathbf{x}'$ violate a payoff constraint of Program 3 if they both get a payoff less than u against a pure strategy $\mathbf{y}$ of player 2. In this case we define three sets L_1 , L_2 , and L_{12} as

$$L_1 = \{i : x_i \leq y_i \text{ and } x'_i > y_i\}, \quad L_2 = \{i : x_i > y_i \text{ and } x'_i \leq y_i\}, \quad L_{12} = \{i : x_i \leq y_i \text{ and } x'_i \leq y_i\}.$$

Observe that L_1 , L_2 , and L_{12} make a critical tuple since both $\mathbf{x}$ and $\mathbf{x}'$ get a payoff less than u against $\mathbf{y}$. Since L_1 , L_2 , and L_{12} are disjoint and $\sum y_i = m$ we have $\sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} \max\{x_i, x'_i\} \leq m$. A similar argument implies that if this condition does not hold for any critical tuple, then $\mathbf{x}$ and $\mathbf{x}'$ meet the conditions of Program 3. $\square$

Based on Observation 4.4 we rewrite Program 3 in the following way.

$$\begin{array}{ll} \text{find} & \mathbf{x} \text{ and } \mathbf{x}' \\ \text{subject to} & x_i \geq 0 \text{ and } x'_i \geq 0 \quad \forall i \in [k] \\ & \sum x_i \leq n \\ & \sum x'_i \leq n \\ & z_i = \max\{x_i, x'_i\} \quad \forall i \in [k] \\ & \sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} z_i > m \quad \text{for every critical tuple } \langle L_1, L_2, L_{12} \rangle \end{array} \quad (4)$$

Indeed Program 4 is not convex since $z_i = \max\{x_i, x'_i\}$ is not a linear constraint. The naive approach to get around this issue is to consider 2^k possibilities for the assignment of z_i 's. More precisely, if we knew in an optimal strategy for which i 's we have $x_i > x'_i$ and for which $x_i \leq x'_i$, we could turn Program 4 into a linear program by replacing each z_i with either x_i or x'_i . This observation gives us an exponential time solution to find a (u, p) -maximin strategy by trying all 2^k combinations. However, for the uniform case, we can further improve the running time to a polynomial. The overall idea is that when we are indifferent between the battlefields, we do not necessarily need to know for which subset of battlefields $\mathbf{x}$ puts more troops than $\mathbf{x}'$. It suffices to be given the count!

In the uniform setting, let $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ be the actual solution. Count the number of battlefields on which $\hat{\mathbf{x}}$ puts more troops than $\hat{\mathbf{x}}'$ and call this number α . Therefore, on $k - \alpha$ battlefields, $\mathbf{x}'$ puts at least as many troops as $\mathbf{x}$. Since the battlefields are identical, we can rearrange the order of the battlefields to make sure $\mathbf{x}$ puts more troops than $\mathbf{x}'$ in the first α battlefields. If α is given to us, we can formulate the problem as follows.

$$\begin{array}{ll} \text{find} & \mathbf{x} \text{ and } \mathbf{x}' \\ \text{subject to} & x_i \geq 0 \text{ and } x'_i \geq 0 \quad \forall i \in [k] \\ & \sum x_i \leq n \\ & \sum x'_i \leq n \\ & x_i \geq x'_i, z_i = x_i \quad \forall i : 1 \leq i \leq \alpha \\ & x_i \leq x'_i, z_i = x'_i \quad \forall i : \alpha < i \leq k \\ & \sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} z_i > m \quad \text{for every critical tuple } L_1, L_2, L_{12} \end{array} \quad (5)$$

Program 5 is clearly an LP. We show in Theorem 4.5 that LP 5 can indeed be solved in polynomial time. This settles the problem when α is given. Note that α takes an integer value between 0 and k and thus we can iterate over all possibilities and solve the problem in polytime.

THEOREM 4.5. *Given that an instance of continuous Colonel Blotto in the uniform setting admits a $(u, 1/2)$ -maximin 2-strategy for player 1, there exists an algorithm to find one such solution in polynomial time.*

PROOF. As discussed earlier, the solution boils down to solving LP 5. Here we show that LP 5 can be solved in polynomial time using the ellipsoid method. For that, we need a separating oracle that for any given assignment to the variables decides in polynomial time whether any constraint is violated and if so, reports one. LP 5 has polynomially many constraints, except the constraints of form

$$\sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} z_i > m \quad \text{for every critical tuple } L_1, L_2, L_{12},$$

since there may be exponentially many critical tuples. Therefore the only challenge is whether any constraint of this form is violated. That is for given strategies $\mathbf{x}$ and $\mathbf{x}'$ and with $z_i = \max\{x_i, x'_i\}$, we need to design an algorithm that finds a critical tuple $\langle L_1, L_2, L_{12} \rangle$ (if any) for which we have

$$\sum_{i \in L_1} x_i + \sum_{i \in L_2} x'_i + \sum_{i \in L_{12}} \max\{x_i, x'_i\} \leq m. \quad (6)$$

For this, note that as implied by Observation 4.4, it suffices to be able to find a pure strategy $\mathbf{y}$ of player 2 such that

$$u_1(\mathbf{x}, \mathbf{y}) < u, \quad \text{and}, \quad u_1(\mathbf{x}', \mathbf{y}) < u. \quad (7)$$

This is in some sense, equivalent to player 2's best-response which we show can be solved in polynomial time via a dynamic program. Define $D(j, m', v, v')$ to be 1 if and only if player 2 can use up to m' troops in the first j battlefields in a way that prevents $\mathbf{x}$ (resp. $\mathbf{x}'$) from obtaining a payoff of at least v (resp. v') in those battlefields. More precisely, $D(j, m', v, v')$ is 1 if and only if there exists a strategy $\mathbf{y}$ for player 2 such that

$$\sum_{i=1}^j y_i \leq m', \quad \sum_{i: i \in [j], x_i > y_i} w_i \leq v, \quad \text{and}, \quad \sum_{i: i \in [j], x'_i > y_i} w_i \leq v'.$$

Clearly, if we are able to solve $D(j, m', v, v')$ for all possible inputs, then it suffices to check the value of $D(k, m, u, u)$ to see whether we can find a strategy $\mathbf{y}$ satisfying (7). Indeed, we can update the dynamic program in the following way:

$$D(j, m', v, v') = \max_{y_j \in \{0, \dots, m'\}} D(j-1, m' - y_j, v - g(y_j, j), v' - g'(y_j, j)),$$

where,

$$g(y_j, j) = \begin{cases} w_j, & \text{if } x_j > y_j, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and}, \quad g'(y_j, j) = \begin{cases} w_j, & \text{if } x'_j > y_j, \\ 0 & \text{otherwise.} \end{cases}$$

As for the base case, we set $D(0, 0, 0, 0) = 1$. The correctness of the dynamic program is easy to confirm, since we basically check all possibilities for the number of troops that the second player can put on the j th battlefield and update the requirements on the previous battlefields accordingly. A minor issue, here, is that this only confirms whether a strategy $\mathbf{y}$ exists that satisfies (7) or not and does not give the actual strategy. However, one can simply obtain the actual strategy by slightly modifying the dynamic program to also store the strategy itself.

To summarize, we gave a polynomial time separating oracle for LP 5 that gives a polynomial time algorithm to solve it which leads to a $(u, 1/2)$ -maximin 2-strategy for player 1 in polynomial. $\square$

4.1.2 General Weights. Theorem 4.5 shows that the problem of computing a $(u, 1/2)$ -maximin 2-strategy is computationally tractable when the weights are uniform. In this section, we study the general (i.e., non-uniform) setting and show that it is possible to obtain an (almost) optimal solution for this problem in polynomial time.

Recall that we assume there exists a $(u, 1/2)$ -maximin 2-strategy and the goal is to either compute or approximate such a strategy. Fix the pure strategies of the solution to be $\hat{x}$ and $\hat{x}'$. Similar to Section 4.1.1, if we knew for which i 's $\hat{x}_i \geq \hat{x}'_i$ holds and for which i 's it is the opposite, we could model the problem as a linear program and find a solution in polynomial time. Since w_i 's are not necessarily the same, unlike Section 4.1.1, we need to try an exponential number of combinations to make a correct decision. To alleviate this problem, we show a generalized variant of the above argument. Define the status of battlefield i to be compatible with either $\leq$ or $\geq$ (or both in case of equality) based on the comparison of $\hat{x}_i$ and $\hat{x}'_i$. Assume we make a guess for the status of battlefields, which is incorrect for a set S of battlefields but correct for the rest of them. This means that for every battlefield i in set S , if $\hat{x}_i > \hat{x}'_i$, we assume $x_i \leq x'_i$ and vice versa. We show that if the total weight of the battlefields in S is small, there exists an almost optimal solution for the problem whose status is compatible with our guess.

For simplicity, we represent a guess for the status of the battlefields with a vector $g \in \{\leq, \geq\}^k$ of length k in which every entry is either ' $\leq$ ' or ' $\geq$ '. A solution (x, x') is compatible with this guess if g_i correctly compares x_i to x'_i .

LEMMA 4.6. *Let $(\hat{x}, \hat{x}')$ be an optimal (u, p) -maximin solution of the problem and g be a guess for the status of the battlefields. Let S be the set of battlefields for which g makes an incorrect comparison between $\hat{x}$ and $\hat{x}'$ on these battlefields. If $\sum_{i \in S} w_i = \alpha$ then there exists a $(u - \alpha, p)$ -maximin strategy that is compatible with g .*

PROOF. We construct a pair of strategies (x, x') based on $\hat{x}$ and $\hat{x}'$. For every battlefield $i \in S$, we set $x_i = x'_i = 0$ and for every battlefield $i \notin S$ we set $x_i = \hat{x}_i$ and $x'_i = \hat{x}'_i$. If a strategy of player 2 prevents both x and x' from getting a utility of $u - \alpha$, then the same strategy prevents both $\hat{x}$ and $\hat{x}'$ from getting a payoff of u . Therefore, (x, x') is $(u - \alpha, p)$ -maximin. Since for every battlefield outside set S we have $x_i = x'_i = 0$, then both $\leq$ and $\geq$ correctly compare the corresponding values for such battlefields. Therefore, (x, x') is compatible with g . $\square$

A simple interpretation of Lemma 4.6 is that if we make a guess that differs from a correct guess in a subset of battlefields with a total weight of α , we can use this guess to find a solution with an additive error of at most α in the utility. Based on this idea, we present a polynomial time algorithm that for any arbitrarily small constant $\epsilon < 1$ computes a $((1 - \epsilon)u, 1/2)$ -maximin 2-strategy.

δ -Uniform weights. One of the crucial steps of our algorithms, is updating battlefield weights. This step, is indeed used in multiple other places of the paper as well. For a parameter δ , we define a δ -uniform variant of the game to be an instance on which the weight of each of the battlefields is rounded down to be in set $\mathcal{W} = \{1, (1 + \delta)^1, (1 + \delta)^2, \dots\}$. That is, for any $i \in [k]$, we set the updated weight of battlefield i to be $w'_i = \max\{w : w \in \mathcal{W}, w \leq w_i\}$. The following observation implies that we can safely assume the game is played on the updated weights without losing a considerable payoff.

OBSERVATION 4.7. *For any u' , any (u', p) -maximin strategy of the game instance $\mathcal{B}(n, m, w')$ with the updated weights is a $((1 - \delta)u', p)$ -maximin strategy of the original instance $\mathcal{B}(n, m, w)$. Similarly, any (u', p) -maximin strategy of the original instance $\mathcal{B}(n, m, w)$ is a $((1 - \delta)u', p)$ -maximin for instance $\mathcal{B}(n, m, w')$.*

PROOF. Let $\mathbf{x} = (x_1, \dots, x_k)$ be any pure strategy in the support of the (u', p) -maximin strategy of $\mathcal{B}(n, m, \mathbf{w}')$. Consider any arbitrary strategy $\mathbf{y}$ of player 2, it suffices to show that $\mathbf{x}$ gets a payoff of at least $(1 - \delta)u'$ against $\mathbf{y}$ in the original instance $\mathcal{B}(n, m, \mathbf{w})$. Note that for any $i \in [k]$ we have $w'_i \geq w_i/(1 + \delta) \geq (1 - \delta)w_i$ by the way that we round down the weights; therefore, we have that

$$\sum_{i: x_i > y_i} w'_i \geq \sum_{i: x_i > y_i} (1 - \delta)w_i \geq (1 - \delta) \sum_{i: x_i > y_i} w_i \geq (1 - \delta)u',$$

completing the proof for the first part.

Similarly, since the weight of each battlefield is multiplied by a factor of at most $1/(1 + \delta)$, any (u', p) -maximin strategy for the original instance is a $(u'/(1 + \delta), p)$ -maximin or simply a $((1 - \delta)u', p)$ -maximin strategy for instance $\mathcal{B}(n, m, \mathbf{w}')$. $\square$

The algorithm in a nutshell. We first update the weight of every battlefield i to be $\min\{u, w_i\}$. This, in fact, does not change the game instance for player 1 since his only objective is to guarantee a payoff of at least u . Now, for $\delta = \epsilon^3/10$ which is a relatively smaller error threshold than ϵ , we consider the δ -uniform variant of the game. In the δ -uniform instance, since the weights change exponentially in $1 + \delta$, we have at most $O(1/\delta \cdot \log u)$ distinct weights. We put the battlefields with the same weight into a bucket and denote the sizes of the buckets by $k_1, k_2, \dots, k_b$ where b is the number of buckets. Recall from Section 4.1.1 that if a set of battlefields have the same weight, then we are indifferent between these battlefields and thus the only information relevant to these battlefields is on how many of them $\hat{\mathbf{x}}$ puts more troops than $\hat{\mathbf{x}}'$. Therefore one way to make a correct guess is to try all $\prod(k_i + 1)$ possibilities for all of the buckets. Unfortunately, $\prod(k_i + 1)$ is not polynomial since the number of buckets is not constant. In order to reduce the number of possibilities to a polynomial, we make a number of observations.

First, since the weights decrease exponentially between the buckets, the number of distinct weights that are larger than $\delta u/k$ (and smaller than u as described above,) is at most $\log_{1+\delta} k/\delta = O_\delta(\log k)$. Observe that we can safely ignore (i.e., make a wrong guess for) all the buckets with weight less than $\delta u/k$ since sum of the weights of all battlefields in such buckets is at most δu and by Lemma 4.6 it causes us to lose an additive error of at most δu . Although this reduces the number of buckets down to $O_\delta(\log k)$, it is still more than we can afford to try all $\prod(k_i + 1)$ possibilities.

Second, instead of trying $k_j + 1$ possibilities for bucket j , we reduce it down to only $O(1/\delta)$ options. More precisely, let t_j be the number of battlefields i in bucket j such that $\hat{x}_i \geq \hat{x}'_i$. For any bucket j with $k_j > 1/\delta$, if we only consider t_j to be in set $\{0, \lfloor \delta k_j \rfloor, \lfloor 2\delta k_j \rfloor, \lfloor 3\delta k_j \rfloor, \dots, k_j\}$ one of the realizations of t_j makes at most $\delta k_j + 1$ incorrect guesses for bucket j . We use this later to argue that we do not lose a significant payoff by considering only $O(1/\delta)$ possibilities per bucket. As a result, we reduce the size of the cartesian product of all possibilities over all the buckets down to a polynomial.

Third, we show that if $n \geq (1 + \epsilon)m/2$, we can safely assume that losing the value of at most $\lfloor \delta k_j \rfloor$ battlefields of buckets with more than $1/\delta$ battlefields does not hurt the payoff significantly. In other words, when $n \geq (1 + \epsilon)m/2$, the optimal utility u is much larger than the total sum of the payoff we lose by only trying $1/\delta$ possibilities for every bucket j (proven in Lemma 4.8).

Based on the above ideas, we outline our PTAS as follows: (i) We first set a cap of u for the weight of the battlefields. (ii) Let $\delta = \epsilon^3/10$. Next, we round down the weight of the battlefields to be powers of $(1 + \delta)$. (iii) We put the battlefields with the same weights in the same bucket and remove the buckets whose battlefield weights are smaller than $\delta u/k$. (iv) Finally, we try $O(1/\delta)$

possibilities for the status of the battlefields within each bucket and check its feasibility using LP 5.⁵ We return the first feasible solution that we find. The formal algorithm is given as Algorithm 1.

Algorithm 1 Algorithm to find a (u, p) -maximin 2-strategy for general weights.

Input: A payoff u for which existence of a $(u, 1/2)$ -maximin 2-strategy is guaranteed.

Output: Two pure strategies $\mathbf{x}$ and $\mathbf{x}'$ that form a $(u, 1/2)$ -maximin when played with equal prob. $1/2$.

- 1: For every battlefield i , update w_i to be $\min\{u, w_i\}$.
 - 2: For $\delta = \epsilon^3/10$, we further update the battlefield weights and consider its δ -uniform variant.
 - 3: Ignore every battlefield i with weight less than $\delta u/k$ (i.e., naively guess $x_i \geq x'_i$).
 - 4: Put all battlefields of the same weight into the same bucket and denote by k_j the number of battlefields in bucket j .
 - 5: For each bucket j with $k_j \leq 1/\delta$, let $G_j = \{\leq, \geq\}^{k_j}$ be the set of all possible guesses for it.
 - 6: For each bucket j with $k_j > 1/\delta$, let G_j be the set of all guesses where for any $d \in \{0, \lfloor \delta k_j \rfloor, \lfloor 2\delta k_j \rfloor, \lfloor 3\delta k_j \rfloor, \dots, k_j\}$ we have $x_i \geq x'_i$ for any $i \leq d$ and $x_i \leq x'_i$ for any $i > d$.
 - 7: Let $G = G_1 \times \dots \times G_b$ be the cartesian product of the partial guesses of the buckets.
 - 8: For any guess $\mathbf{g} \in G$, construct an instance of LP 5 and return the first found feasible solution $(\mathbf{x}, \mathbf{x}')$.
-

Before we present a formal proof, we state an auxiliary lemma to show a lower bound on the value of u when $n \geq (1 + \epsilon)m/2$.

LEMMA 4.8. *Let α be the total sum of the weights of the battlefields whose buckets have a size of at least $1/\delta$. If $n > (1 + \epsilon)m/2$ then there exists a $(\epsilon\alpha/8, 1/2)$ -maximin strategy for player 1 that randomizes over two pure strategies.*

PROOF. Let $\mathcal{B} = \{B_1, \dots, B_b\}$ be the set of all buckets with at least $1/\delta$ battlefields. We slightly abuse the notation and respectively denote by w_i and k_i the weight and the number of battlefields in B_i . This means we have $\sum_{i=1}^b k_i w_i = \alpha$. We construct a 2-strategy $(\mathbf{x}, \mathbf{x}')$ where both $\mathbf{x}$ and $\mathbf{x}'$ are played with equal probability $1/2$ and claim that it is $(\epsilon\alpha/8, 1/2)$ -maximin. To that end, for any bucket $B_i \in \mathcal{B}$ with an odd number of battlefields, we ignore one battlefield (i.e., we put zero troops in it in both $\mathbf{x}$ and $\mathbf{x}'$) to consider only an even number of battlefields for each bucket. Denote by α' the total weight of the remaining battlefields, i.e., the battlefields in some $B_i \in \mathcal{B}$ that are not ignored. It is easy to confirm that

$$\alpha' \geq (1 - \delta)\alpha \quad (8)$$

since from each of the buckets in $\mathcal{B}$, at most one battlefield is ignored, which is only a δ fraction of the battlefields in that bucket since all buckets in $\mathcal{B}$ are assumed to have at least $1/\delta$ battlefields. Now, since only an even number of battlefields remain in each bucket B_i , we can partition them into two disjoint subsets $B_i^{(1)}$ and $B_i^{(2)}$ of equal size. Strategies $\mathbf{x}$ and $\mathbf{x}'$ are then constructed as follows:

- In strategy $\mathbf{x}$, for any $i \in [b]$, we put exactly $2w_i n / \alpha'$ troops in each battlefield in $B_i^{(1)}$. We put zero troops in all other battlefields.
- In strategy $\mathbf{x}'$, for any $i \in [b]$, we put exactly $2w_i n / \alpha'$ troops in each battlefield in $B_i^{(2)}$. We put zero troops in all other battlefields.

⁵As a minor technical detail, since our goal is to guarantee a payoff of at least $(1 - \epsilon)u$ instead of u , we need to update the definition of losing tuples accordingly for this case.

Observe that the total number of troops that we use in each of the strategies $\mathbf{x}$ and $\mathbf{x}'$ is exactly n as required. To see this, in strategy $\mathbf{x}$ for instance, the number of troops that are used is

$$\sum_{i \in [b]} \frac{2w_i n}{\alpha'} |B_i^{(1)}| = \frac{2n \sum_{i \in [b]} w_i |B_i^{(1)}|}{\alpha'} = \frac{2n \sum_{i \in [b]} w_i |B_i^{(1)}|}{\sum_{i \in [b]} w_i \cdot 2|B_i^{(1)}|} = n.$$

It only remains to prove that this strategy is $(\epsilon\alpha/8, 1/2)$ -maximin. Assume for the sake of contradiction that player 2 has a strategy $\mathbf{y}$ that prevents both $\mathbf{x}$ and $\mathbf{x}'$ from achieving a payoff of at least $\epsilon\alpha/8$. We can assume w.l.o.g., that for every battlefield i , we have $y_i \in \{0, x_i, x'_i\}$, thus, on all battlefields that are ignored (i.e., $x_i = x'_i = 0$), we have $y_i = 0$. Further, note that because of the special construction of strategies $\mathbf{x}$ and $\mathbf{x}'$, in each battlefield, at most one of $\mathbf{x}$ or $\mathbf{x}'$ put non-zero troops; therefore, on every battlefield i where $y_i > 0$, either we have $x_i > 0$ or $x'_i > 0$. Define

$$m_1 = \sum_{i \in [k]: x_i > 0} y_i, \quad \text{and,} \quad m_2 = \sum_{i \in [k]: x'_i > 0} y_i.$$

Note that $m_1 + m_2 \leq m$ since it cannot be the case that both x_i and x'_i are non-zero at the same time as described above. Therefore at least one of m_1 or m_2 is not more than $m/2$. Assume w.l.o.g., that $m_1 \leq m/2$. One can think of m_1 as the number of troops that are spent by player 2 to prevent strategy $\mathbf{x}$ from getting a payoff of at least $\epsilon\alpha/8$. To obtain the contradiction, we prove that player 2 cannot use only $m/2$ troops to prevent $\mathbf{x}$ from obtaining a payoff of $\epsilon\alpha/8$. Recall that we denote the total sum of battlefields on which we put a non-zero number of troops either in $\mathbf{x}$ or $\mathbf{x}'$ by α' . Strategy $\mathbf{x}$ puts non-zero troops in half of these battlefields, and therefore sum of their weights is at least $\alpha'/2$. To prevent $\mathbf{x}$ from getting a payoff of at least $\epsilon\alpha/8$, player 2 can lose a weight of less than $\epsilon\alpha/8$ on these battlefields. Let $\mathcal{Y}$ be the subset of battlefields on which $\mathbf{y}$ wins $\mathbf{x}$ and let $w(\mathcal{Y})$ be the total weight of all these battlefields. We need to have

$$w(\mathcal{Y}) > \alpha'/2 - \epsilon\alpha/8. \quad (9)$$

Since the number of troops that is put on the battlefields in $\mathbf{x}$ is proportional to the battlefield weights on which $\mathbf{x}$ puts non-zero troops, we have

$$\sum_{i \in \mathcal{Y}} x_i \geq \left(\frac{w(\mathcal{Y})}{\alpha'/2} \right) n \stackrel{\text{By (9)}}{\geq} \left(\frac{\alpha'/2 - \epsilon\alpha/8}{\alpha'/2} \right) n \geq \left(1 - \frac{\epsilon\alpha}{4\alpha'} \right) n \stackrel{\text{By (8)}}{\geq} \left(1 - \frac{\epsilon}{4(1-\delta)} \right) n \stackrel{\text{Since } \delta = \epsilon^3/10}{\geq} \frac{n}{1+\epsilon}.$$

Therefore, to be able to match the troops of $\mathbf{x}$ in every battlefield in $\mathcal{Y}$, using only $m/2$ troops, we have

$$m/2 \geq \sum_{i \in \mathcal{Y}} x_i \geq \frac{n}{1+\epsilon}. \quad (10)$$

The last inequality contradicts the assumption of the lemma that $n > (1+\epsilon)m/2$. Therefore, there exists no such strategy $\mathbf{y}$ for player 2. That means, the constructed strategy is indeed a $(\epsilon\alpha/8, 1/2)$ -maximin 2-strategy. $\square$

THEOREM 4.9. *Let $\epsilon > 0$ be an arbitrarily small constant and suppose that we have an instance of continuous Colonel Blotto in which $n \geq (1+\epsilon)m/2$. Given that player 1 has a $(u, 1/2)$ -maximin 2-strategy, there exists an algorithm that finds a $((1-\epsilon)u, 1/2)$ -maximin 2-strategy for him in polynomial time.*

PROOF. The algorithm to achieve this strategy is given as Algorithm 1. We first analyze the approximation factor of Algorithm 1 and then prove that it runs in polynomial time.

Approximation factor. Let $\mathbf{w}'$ denote the updated battlefield weights by the end of Line 2 of Algorithm 1. First note that setting a cap of u for the battlefield weights in the first line of algorithm does not change the game instance at all since the only goal of player 1 is to guarantee a payoff of

at least u . Second, we know by Observation 4.7 that for any u' , any (u', p) -maximin strategy for the δ -uniform variant is a $((1 - \delta)u', p)$ -maximin strategy for the original (i.e., not δ -uniform) instance. Roughly speaking, since δ is relatively smaller than ϵ , we still get a $((1 - \epsilon)u, 1/2)$ -maximin strategy for the original instance if we achieve a good approximation on the δ -uniform variant.

Consider an optimal $(u, 1/2)$ -maximin 2-strategy for player 1 on the original instance (which recall is guaranteed to exist) and assume that it randomizes over two pure strategies $\hat{x}$ and $\hat{x}'$. By Observation 4.7, this is a $((1 - \delta)u, 1/2)$ -maximin strategy for the δ -uniform variant. Let us denote by vector $\mathbf{c} \in \{\leq, \geq\}^k$ the comparison between the entries of $\hat{x}$ and $\hat{x}'$. That is, c_i is ' $\geq$ ' if and only if $\hat{x}_i \geq \hat{x}'_i$ and it is ' $\leq$ ' otherwise. Our goal is to argue that there exists a guess $\mathbf{g} \in G$ that is *sufficiently close* to $\mathbf{c}$ – where by sufficiently close we mean sum of weights of all battlefields with $g_i \neq c_i$ is very small. We will later combine this with Lemma 4.6 to obtain the desired guarantee.

We first assume w.l.o.g. that for any two battlefields i and j with $i \leq j$ that have the same weight, if $c_i \neq c_j$, then $c_i = \leq$ and $c_j = \geq$ (otherwise we swap these two battlefields without changing the payoff guaranteed by the strategy). We have two relaxations over the guesses in Lines 3 and 6 of Algorithm 1.

First, in Line 3 of Algorithm 1, sum of weights of all light battlefields with weight at most $\delta u/k$ is not more than δu since k is the total number of battlefields. Thus, even if $g_i \neq c_i$ on these battlefields, their total sum is less than δu .

Second, in Line 6 of Algorithm 1, let us denote by α the sum of weights of all battlefields whose bucket contains more than $1/\delta$ battlefields. Observe that we check almost all possibilities of guesses on these buckets, except on δ fraction of their battlefields. More precisely, the total sum of weights of such battlefields on which our guess is wrong is at most $\delta\alpha$. It only remains to argue that $\delta\alpha$ cannot be very large. Note that by Lemma 4.8, we can obtain a simple $(\epsilon\alpha/8, 1/2)$ -maximin strategy since the condition of $n \geq (1 + \epsilon)m/2$ is also satisfied here. Thus, we can assume $u \geq \epsilon\alpha/8$ or simply $\alpha \leq 8u/\epsilon$ (otherwise instead of Algorithm 1 we return the strategy of Lemma 4.8). Therefore the total weight of battlefields of this type, on which we guess wrong is no more than

$$\delta\alpha \leq \frac{\delta 8u}{\epsilon} \stackrel{\text{Since } \delta = \epsilon^3/10}{\leq} \frac{4}{5}\epsilon^2 u.$$

Combining these two, we show that there exists a guess $\mathbf{g} \in G$ for which sum of battlefields with $g_i \neq c_i$ is at most

$$\delta u + \frac{4}{5}\epsilon^2 u = \frac{1}{10}\epsilon^3 u + \frac{4}{5}\epsilon^2 u < \epsilon^2 u.$$

By Lemma 4.6, this implies a $((1 - \delta)u - \epsilon^2 u, 1/2)$ -maximin 2-strategy for the δ -uniform variant; and by Observation 4.7, guarantees a utility of at least

$$(1 - \delta - \epsilon^2)(1 - \delta)u = (1 - \frac{\epsilon^3}{10} - \epsilon^2)(1 - \frac{\epsilon^3}{10})u \geq (1 - \epsilon)u$$

for the original instance with probability at least $1/2$, which in other words, gives a $((1 - \epsilon)u, 1/2)$ -maximin 2-strategy as desired.

Running time. It is easy to confirm that the running time of Algorithm 1 is $\text{poly}(|G|)$. Thus, it suffices to show that the total number of guesses in G is polynomial. Observe that for any $i \in [b]$, the total number of partial guesses for bucket i is $O(1)$ (though dependent on δ). On the other hand, the total number of buckets as mentioned before is at most $O(\log k)$ (we hide the dependence on δ) therefore $|G| \leq O(1)^{O(\log k)}$ which is polynomial. $\square$

Notice that when $n \leq m/2$, there is no chance for player 1 to get a nonzero utility by randomizing over two pure strategies since player 2 can always play $y_i = \max\{x_i, x'_i\}$ troops on every every battlefield and win all of them.

By generalizing the ideas mentioned above, we can extend our approach to the case when $c > 2$. For space limits, we defer the details of this generalization as well as the algorithms for the discrete variant and our complexity results to the full version of the paper [3].

5 CONCLUSION

In this paper, we presented efficient algorithms to find arbitrarily good approximate solutions for the variant of the Colonel Blotto game where we seek to find strategies whose support is bounded by a given parameter c . This upper bound on the support size radically changes the problem. We believe it is important to study such variants of other well-studied games and see whether they can also be well-approximated when the support size is bounded.

REFERENCES

- [1] AmirMahdi Ahmadinejad, Sina Dehghani, MohammadTaghi Hajiaghayi, Brendan Lucier, Hamid Mahini, and Saeed Seddighin. 2016. From Duels to Battlefields: Computing Equilibria of Blotto and Other Games.. In *AAAI*. 376–382.
- [2] Soheil Behnezhad, Avrim Blum, Mahsa Derakhshan, MohammadTaghi HajiAghayi, Mohammad Mahdian, Christos H Papadimitriou, Ronald L Rivest, Saeed Seddighin, and Philip B Stark. 2018. From Battlefields to Elections: Winning Strategies of Blotto and Auditing Games. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2291–2310.
- [3] Soheil Behnezhad, Avrim Blum, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Christos H Papadimitriou, and Saeed Seddighin. 2019. Optimal Strategies of Blotto Games: Beyond Convexity. *arXiv preprint arXiv:1901.04153* (2019).
- [4] Soheil Behnezhad, Sina Dehghani, Mahsa Derakhshan, MohammadTaghi HajiAghayi, and Saeed Seddighin. 2017. Faster and Simpler Algorithm for Optimal Strategies of Blotto Game.. In *AAAI*. 369–375.
- [5] Émile Borel. 1921. La théorie du jeu et les équations intégrales à noyau symétrique. *Comptes Rendus de l'Académie* 173, 13041308 (1921), 97–100.
- [6] Émile Borel. 1953. The theory of play and integral equations with skew symmetric kernels. *Econometrica* 21 (1953), 97–100.
- [7] Subhasish M Chowdhury, Dan Kovenock, and Roman M Sheremeta. 2009. An experimental investigation of Colonel Blotto games. (2009).
- [8] Vladimir Estivill-Castro and Mahdi Parsa. 2009. Computing Nash equilibria gets harder: new results show hardness even for parameterized complexity. In *Proceedings of the Fifteenth Australasian Symposium on Computing: The Australasian Theory-Volume 94*. Australian Computer Society, Inc., 83–90.
- [9] M. Fréchet. 1953. Commentary on the three notes of Emile Borel. *Econometrica* 21 (1953), 118–124.
- [10] M. Fréchet. 1953. Emile Borel, initiator of the theory of psychological games and its application. *Econometrica* 21 (1953), 95–96.
- [11] Itzhak Gilboa and Eitan Zemel. 1989. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior* 1, 1 (1989), 80–93.
- [12] Oliver Alfred Gross and RA Wagner. 1950. A continuous Colonel Blotto game. (1950).
- [13] Sergiu Hart. 2008. Discrete Colonel Blotto and general lotto games. *International Journal of Game Theory* 36, 3 (2008), 441–460.
- [14] Rafael Hortala-Vallve and Aniol Llorente-Saguer. 2012. Pure strategy Nash equilibria in non-zero sum colonel Blotto games. *International Journal of Game Theory* 41, 2 (2012), 331–343.
- [15] Nicole Immorlica, Adam Tauman Kalai, Brendan Lucier, Ankur Moitra, Andrew Postlewaite, and Moshe Tennenholtz. 2011. Dueling algorithms. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*. ACM, 215–224.
- [16] Dan Kovenock and Brian Roberson. 2010. Conflicts with multiple battlefields. (2010).
- [17] Dan Kovenock and Brian Roberson. 2012. Coalitional Colonel Blotto games with application to the economics of alliances. *Journal of Public Economic Theory* 14, 4 (2012), 653–676.
- [18] Jean-Francois Laslier and Nathalie Picard. 2002. Distributive politics and electoral competition. *Journal of Economic Theory* 103, 1 (2002), 106–130.
- [19] Richard J Lipton, Evangelos Markakis, and Aranyak Mehta. 2003. Playing large games using simple strategies. In *Proceedings of the 4th ACM conference on Electronic commerce*. ACM, 36–41.
- [20] Jennifer Merolla, Michael Munger, and Michael Tofias. 2005. In play: A commentary on strategies in the 2004 us presidential election. *Public Choice* 123, 1-2 (2005), 19–37.
- [21] Roger B Myerson. 1993. Incentives to cultivate minorities under alternative electoral systems. *American Political Science Review* 87, 4 (1993), 856–869.

- [22] Brian Roberson. 2006. The colonel blotto game. *Economic Theory* 29, 1 (2006), 1–24.
- [23] Brian Roberson and Dmitriy Kvasov. 2012. The non-constant-sum Colonel Blotto game. *Economic Theory* 51, 2 (2012), 397–433.
- [24] Ariel Rubinstein. 1998. *Modeling bounded rationality*. MIT press.
- [25] Martin Shubik and Robert James Weber. 1981. Systems defense games: Colonel Blotto, command and control. *Naval Research Logistics Quarterly* 28, 2 (1981), 281–287.
- [26] Herbert Alexander Simon. 1982. *Models of bounded rationality: Empirically grounded economic reason*. Vol. 3. MIT press.