From Battlefields to Elections: Winning Strategies of Blotto and Auditing Games[*]

Soheil Behnezhad[†] Avrim Blum[‡] Mahsa Derakhshan[†] MohammadTaghi HajiAghayi[†]
Mohammad Mahdian[§] Christos H. Papadimitriou[¶] Ronald L. Rivest[||]
Saeed Seddighin[†] Philip B. Stark[**]

Abstract

Mixed strategies are often evaluated based on the *expected payoff* that they guarantee. This is not always desirable. In this paper, we consider games for which maximizing the expected payoff deviates from the actual goal of the players. To address this issue, we introduce the notion of a (u, p)-maxmin strategy which ensures receiving a minimum utility of u with probability at least p. We then give approximation algorithms for the problem of finding a (u, p)-maxmin strategy for these games.

The first game that we consider is *Colonel Blotto*, a well-studied game that was introduced in 1921. In the Colonel Blotto game, two colonels divide their troops among a set of battlefields. Each battlefield is won by the colonel that puts more troops in it. The payoff of each colonel is the weighted number of battlefields that she wins. We show that maximizing the expected payoff of a player does not necessarily maximize her winning probability for certain applications of Colonel Blotto. For example, in presidential elections, the players' goal is to maximize the probability of winning more than half of the *votes*, rather than maximizing the expected number of votes that they get. We give an exact algorithm for a natural variant of *continuous* version of this game. More generally, we provide constant and logarithmic approximation algorithms for finding (u, p)-maxmin strategies.

We also introduce a *security game* version of Colonel Blotto which we call *auditing game*. It is played between two players, a *defender* and an *attacker*. The goal of the defender is to prevent the attacker from changing the outcome of an instance of Colonel Blotto. Again, maximizing the expected payoff of the defender is not necessarily optimal. Therefore we give a constant approximation for (u, p)-maxmin strategies.

1 Introduction

Pure strategies are often not desirable as they typically allow the opponent to exploit the chosen strategy. Randomization through mixed strategies has shown to be effective in addressing this issue. In the literature, mixed strategies are mostly evaluated based on the *expected payoff* that they guarantee. For example, a *maxmin* strategy maximizes the minimum possible expected payoff of a player. This is misleading when the tail behavior of a strategy is particularly important. In the following paragraphs, we start by the definition of some natural games for which the objective is a more complicated function than the expected payoff.

Colonel Blotto. Colonel Blotto was first introduced by Borel in 1921 [6]. In the *Colonel Blotto* game, two colonels each have a pool of troops and fight against each other over a set of battlefields. The colonels simultaneously divide their troops between the battlefields. A colonel wins a battlefield if the number of her troops dominates the number of troops of her opponent. Each battlefield has an associated weight, and the final payoff of each colonel is the sum of the weights of the battlefields that she wins. The *continuous* variant of the game corresponds to the case where the resources (troops) are capable of continuous partition; whereas the *discrete* version considers the case where the partitions are natural numbers.

Recent studies have made significant progress in understanding the optimal strategies of Colonel Blotto when the players are expectation maximizers [1, 4, 25, 16, 17, 28, 18, 26]. Note that even the problem of maximizing the expected payoff in the Colonel Blotto game is quite a challenge since the players have a huge number of pure strategies. The pioneering work of Immorlica *et al.* [19] initiated the study of large constant-sum games (such as Colonel Blotto) and showed that in some cases finding the equilibria of these games is tractable. The first polynomial time algorithm to find the equilibria of Colonel Blotto was presented by Ahmadinejad *et al.* [1].

[*]This work was conducted in part while the authors were visiting the Simons Institute for the Theory of Computing.

[†]University of Maryland, College Park. 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.

[‡]TTI-Chicago and CMU. Supported in part by NSF CCF-1331175.

[§]Google Research, New York

[¶]Computer Science Division, UC Berkeley

[||]CSAIL, MIT, Cambridge

[**]Department of Statistics, UC Berkeley

Later, Behnezhad *et al.* [4] improved upon this algorithm to obtain a "faster and simpler" solution.

Although the Colonel Blotto game was initially proposed to study a war situation, it has found applications in the analysis of many different forms of competition. Perhaps the most notable application of Colonel Blotto is in the U.S. presidential election where 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 states, 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.[1] It might happen that the winning candidate receives fewer votes than her opponent. Therefore, the candidates strategize their resources (e.g., money, staff, etc.) to maximize the number of electoral votes they win and as such, their policies might undermine the popular vote.

This form of election can be modeled as a Colonel Blotto game by corresponding each state to a battlefield and modeling the candidates' resources with the troops of the colonels. If the candidates were to maximize the *expected number* of electoral votes, the optimal strategies could be characterized and computed via the known techniques [19, 1, 4, 15, 35, 34]. However, in the U.S. election, the goal of the parties is to maximize the likelihood of winning the race which is the probability that their candidate wins the majority of the electoral votes. To illustrate how different the two objectives could be, imagine that a strategy of a candidate secures an expected number of 280 electoral votes out of 538 votes in total. Now, if this solution guarantees 270 electoral votes with probability 0.5 and receives 290 electoral votes with probability 0.5, then the corresponding strategy always wins the race. Another (artificial) possibility is that this strategy receives 260 electoral votes with probability 9/10 and 460 votes with probability 1/10 and thus losing the race with probability 9/10 despite receiving more than half of the electoral votes in expectation.

Although expectation maximizer strategies of Blotto have received a lot of attention over the past few decades [6, 7, 13, 14], prior to this work, not much was known for the case where the goal is to maximize the likelihood of winning a certain amount of payoff. In this work, we study this problem for both the discrete and continuous variants of Colonel Blotto. In particular, for the discrete variant of Colonel Blotto, we present a logarithmic approximation algorithm and improve this result for the continuous case to a constant approximation algorithm. We also give an exact algorithm for the guaranteed payoff setting (when the goal is to obtain a utility u with probability 1) in the continuous case. Moreover, we provide improved algorithms for the uniform case (when all the battlefields have the same weight). an al

Auditing game. This game could be viewed as a *security game* version of the Colonel Blotto game. A security game has a *defender* and an *attacker* and the goal of the defender is to *protect* a set of targets from possible attacks of the attacker using her limited resources. Many different variants of security games, targeting different applications (e.g., protecting a set of nodes in a graph, scheduling a set of patrols in space and time, etc.) have been considered in the literature [5, 11, 36, 8, 20].

For a given instance of Colonel Blotto, the goal of the *defender* in the auditing game is to prevent an *attacker* from changing the outcome of the game by protecting the battlefields. Since we consider the problem in the full information setting, we assume both players have full information of the game including the winner of each battlefield.[2] Let us again focus on the presidential election. In this setting, a *hacker* plays the role of the attacker and an *auditor* plays the role of a defender. The hacker tries to change the winner by *hacking* some states and changing the outcome of those states in favor of a player (an actual loser of the election). The auditor, on the other hand, is able to audit a limited number of states to prevent this. If the auditor protects a state that the hacker is trying to attack, the auditor catches the hacker. If the hacker is caught by the auditor, she gets utility 0 but otherwise, her utility would be the total number of electoral votes that she hacks. The game is constant-sum and the sum of the payoffs of the players is always the total number of electoral votes. Similar to Colonel Blotto, maximizing the expected payoff of the auditor does not necessarily maximize the chance of preventing the hacker from changing the winner of the race. We show in Section 7 that given a threshold u for the utility of the hacker, we can compute a constant approximation algorithm that approximately maximizes the likelihood of the auditor to prevent the hacker from obtaining a payoff more than u.

We refer the interested readers to the rich literature on different auditing strategies and methodologies in

[1] All states except Maine and Nebraska choose their electors on a winner-take-all basis. In this work, we consider an idealized electoral vote system where a state may not split its electoral votes.

[2] In real world, an estimation of the votes could be learned from internal polls.

protecting election results [10, 9, 32, 30, 29, 31, 33, 23, 24, 22].

An alternative to expectation. One way to tackle this problem is to change the payoff function of the players. For example, in Colonel Blotto, if we change the utility of the players such that a player gets a utility 1 if and only if she wins more than half of the electoral votes, maximizing her expected payoff would indeed maximize her winning chance. However, instead of changing the game setting (which breaks the linearity assumption and causes a combinatorial explosion), we take a rather different (and generalizable) approach.

Consider a two-player game between player A and player B.[3] We call a strategy of player A a (u,p)-maxmin strategy, if it guarantees a utility of at least u for her with probability at least p, regardless of the strategy that player B chooses. In other words, a strategy X is (u,p)-maxmin if for every (possibly mixed) strategy Y of the opponent we have

$$\Pr_{x\sim X, y\sim Y}[\mathsf{u}^{\mathsf{A}}(x,y) \geq u] \geq p,$$

where $\mathsf{u}^{\mathsf{A}}(x,y)$ denotes the payoff of player A if she plays strategy x and player B plays strategy y. Now for a given required payoff u and probability p, the problem is to find a (u,p)-maxmin strategy or report that there is no such strategy.

For many natural games, solving (u,p)-maxmin (for any given u and p) is computationally harder than the case where we focus on expected payoff (e.g., see Section 4.2 for a discussion about why it seems to be computationally harder to solve (u,p)-maxmin rather than maximizing expected payoff for Colonel Blotto). Therefore, it is reasonable to look for approximation algorithms. An approximate solution may relax the given probability, the given payoff, or both.

2 Our Results and Techniques

Throughout this paper, we consider the *discrete* and *continuous* variants of Colonel Blotto, as well as the auditing game and provide approximately optimal (u,p)-maxmin strategies for these games. Our main result is an algorithm with logarithmic approximation factor for the discrete Colonel Blotto game. Next, we provide a constant approximation algorithm for the *continuous* variant of Colonel Blotto and finally we provide a constant approximation algorithm for the auditing game.

At a high level, our techniques are inspired by recent developments in game theory and optimization. For instance, when the goal is to maximize the guaranteed payoff of a player (finding a $(u,1)$-maxmin strategy) our problem settings generalizes Stackelberg games. In addition to this, when the goal is to find a (u,p)-maxmin strategy for an arbitrary $0 \leq p \leq 1$, the problem extends the robust optimization problem studied in [12]. We also devise a decomposition technique inspired by [3, 21, 2, 27].

We recall that a plethora of studies have analyzed and characterized the equilibria of structured zero-sum games such as Colonel Blotto [19, 1, 4, 15, 35, 34]. In particular, Ahmadinejad *et al.* [1] and Behnezhad *et al.* [4] present polynomial time algorithms to compute the maxmin strategies of Colonel Blotto. Provided that the maxmin strategies of Colonel Blotto are available, it is crucial to understand how well such strategies perform when the objective is *not* to maximize the expected payoff but to approximate a (u,p)-maxmin strategy. We begin in Section 4, by illustrating the difference between the maxmin strategies and (u,p)-maxmin strategies. Although we show that in special cases, a maxmin strategy provides a decent approximation of a (u,p)-maxmin strategy, we present an example to show that in general, maxmin strategies are not competitive to the (u,p)-maxmin ones. Our counter-example is a Colonel Blotto game in which the number of troops of player B is many times more than the troops of player A.

Theorem 4.1 [restated]. *For any given u, p and arbitrarily small constants $0<\alpha<1$ and $0<\beta<1$, there exists an instance of Colonel Blotto (both discrete and continuous), where for any approximate $(\alpha' u, \beta' p)$-maxmin strategy that an expectation maximizer algorithm returns, either $\alpha' < \alpha$ or $\beta' < \beta$.*

Theorem 4.1 states that in order to provide an exact or even an approximation algorithm for (u,p)-maxmin strategies, one needs to go beyond the expectation maximizer algorithms. Following this observation, we begin our results by studying the special case of $(u,1)$-maxmin or in other words the *u-guaranteed payoff* strategies for the discrete variant of Colonel Blotto game.

For the special case of $p=1$, one possible strategy of the opponent is to randomize over all pure strategies. Thus, any strategy of player A that guarantees a payoff of at least u with probability 1, must obtain a payoff of at least u against any pure strategy of the opponent. Indeed this condition is sufficient to declare a strategy $(u,1)$-maxmin or in other words, any strategy of player A that obtains a payoff of at least u against any pure strategy of player B is $(u,1)$-maxmin. Moreover, one can show that randomization offers no benefit to player A when the objective is to find a $(u,1)$-maxmin strategy. Therefore, the definition of $(u,1)$-maxmin strategies coincides with the notion of *pure maxmin* strategies.

[3]Our definition could easily be generalized to multi-player games.

Based on this, our objective is to find a pure strategy for player A that obtains the maximum payoff against any best response of the opponent. This is very similar to Stackelberg games with the exception that here we only incorporate the pure strategies of player A.

For a fixed strategy of player A, the best response of player B can be modeled as a knapsack problem. Let a_i denote the number of troops of player A in battlefield i. In order for player B to maximize her payoff (or equivalently minimize player A's payoff) she needs to find a subset of battlefields S and put a_i troops in every battlefield i in this subset. The constraint is that she can only afford to put M troops in those battlefields and therefore $\sum_{i \in S} a_i$ should be bounded by M. Therefore the problem is to find a subset S of battlefields with the maximum total weight subject to $\sum_{i \in S} a_i \leq M$. This problem can be solved in time $\mathsf{poly}(N, M, K)$ (where K is the number of battlefields) with a classic knapsack algorithm. However, a polytime algorithm for best response does not lead to a polytime solution since player A has exponentially many pure strategies and verifying all such strategies takes exponential time.

To overcome this challenge, we relax the best response algorithm of player B to an almost best response greedy algorithm. Let $\mathsf{W}_{\mathsf{max}} = \max w_i$ be the maximum weight of a battlefield or equivalently the maximum profit of an item in the knapsack problem. It is well-known that the following greedy algorithm for knapsack guarantees an additive error of at most $\mathsf{W}_{\mathsf{max}}$ in comparison to the optimal solution: sort the items based on the ratio of profit over size and put these items into the knapsack accordingly. Based on this observation, if we restrict the opponent to play according to the greedy algorithm, the performance of our solution drops by an additive factor of at most $\mathsf{W}_{\mathsf{max}}$. Once we replace the strategy of player B by the greedy knapsack algorithm, finding a maxmin strategy of player A becomes tractable. More precisely, we show that the problem of finding an optimal strategy for player A against the greedy knapsack algorithm boils down to a dynamic program that can be solved in polynomial time.

In order to turn the $\mathsf{W}_{\mathsf{max}}$ additive error into a $1/2$ multiplicative error, we also consider a strategy of player A that puts all her troops in the battlefield with the highest weight. We show that the better of the two strategies guarantees a profit of at least $u/2$ against any strategy of the opponent where u is the maximum possible guaranteed payoff of player A.

Theorem 5.1 [restated]. There exists a polynomial time algorithm that gives a $(u/2, 1)$-maxmin strategy of player A, assuming that u is the maximum guaranteed payoff of player A.

In Section 5.2, we consider the problem of approximating a (u, p)-maxmin strategy for an arbitrary u and $0 \leq p \leq 1$. This case is more challenging than the case of guaranteed payoff since (1) the solution is not necessarily a pure strategy; and (2) the knapsack modeling for the best response of player B is no longer available. We begin by considering the special case of uniform weights (wherein all the weights are equal to 1) and providing an algorithm for approximating a (u, p)-maxmin strategy in this setting. Later, we reduce instances with general weights to this case. Since all the weights are equal to 1, we are able to characterize the optimal strategies of the players and based on that we provide simple strategies that obtain a fraction of the guarantees that the optimal strategies provide.

Theorem 5.2 [restated]. *Given that there exists a (u, p)-maxmin strategy for player A in an instance of discrete Colonel Blotto with uniform weights, there exists a polynomial time algorithm that provides a $(u/8, p/2)$-maxmin strategy.*

The more technically involved result of Section 5.2 concerns the case where the weights are not necessarily uniform. We show a reduction from the case of non-uniform weights to the case of uniform weights that loses an $O(\log K)$ on the payoff of the algorithm. The high-level idea is as follows: In order to approximate a (u, p)-maxmin strategy, we separate the battlefields into two categories *high-value* and *low-value*. High-value battlefields have a weight of at least $u/O(\log K)$ and the payoff of the low-value battlefields is below this threshold. The idea is that in order for player A to obtain a payoff of at least $u/O(\log K)$ it only suffices to win a high-value battlefield. Moreover, if the number of low-value battlefields is considerable, player A may distribute her troops over those battlefields and obtain a payoff of at least $u/O(\log K)$. Since any high-value battlefield provides a payoff of at least $u/O(\log K)$, we can ignore the weights and play on these battlefields as if all their weights were equal. For the low-value battlefields, on the other hand, we take advantage of the fact that any battlefield of this type contributes a small payoff to the optimal solution and thus via a combinatorial argument we reduce the problem to the case of uniform weight. We then state that if player A flips a coin and plays on each set of battlefield with probability $1/2$ she can obtain a payoff of at least $u/O(\log K)$ with probability at least $p/O(1)$ given that there exists a (u, p)-maxmin strategy for player A. This method is similar to the *core-tail decomposition* technique used in [3, 21, 2, 27] to design approximately optimal mechanisms in the worst case scenarios.

Theorem 5.3 [restated]. *Given that a (u,p)-maxmin strategy exists for player A in an instance of discrete Colonel Blotto, there exists a polynomial time algorithm that provides a $(u/(16(\lceil \log K \rceil + 1)), p/4)$-maxmin strategy.*

We also consider the continuous Colonel Blotto problem in Section 6. In the continuous variant of the problem, the players may allocate a real number of troops to a battlefield. We show that this enables us to exactly compute the optimal guaranteed payoff of player A with an LP. Recall that the best response of player B in the guaranteed payoff case can be modeled via a knapsack problem. Let a_i denote the number of troops that player A allocates to a battlefield i. Based on the knapsack model, player B's best response is to select a subset S of battlefields with the maximum possible payoff subject to $\sum_{i \in S} a_i \leq M$. Indeed this is a linear constraint and thus the problem of computing a $(u,1)$-maxmin strategy can be formulated as a linear program as follows: define K variables $a_1, a_2, \ldots, a_K$ to denote the number of troops of player A in each of the K battlefields. Every strategy of player B cannot get a payoff more than $\sum w_i - u$ and thus any subset of battlefields with a total weight of at least $\sum w_i - u$ should have a total number of troops more than M. Of course, this adds an exponential number of linear constraints to the LP, nonetheless, we show that ellipsoid method can solve this program in polynomial time.

Theorem 6.1 [restated]. For any given instance of continuous Colonel Blotto and any given u, there exists a polynomial time algorithm to either find a $(u,1)$-maxmin strategy or report that no $(u,1)$-maxmin strategy exists.

Furthermore, similar to the high-value, low-value decomposition of the battlefields we described above, we show that the problem for the case of (u,p)-maxmin reduces to the case of uniform battlefield weights and as a result, one can design a polynomial time algorithm to provide a constant approximation of a (u,p)-maxmin strategy.

Theorem 6.2 [restated]. Given that a (u,p)-maxmin strategy exists for player A in an instance of continuous Colonel Blotto, Algorithm 5 provides a $(u/8, p/8)$-maxmin strategy.

Finally, in Section 7 we study the notion of (u,p)-maxmin strategies for the auditing game. In the auditing game, an instance of the Colonel Blotto game (such as the US presidential election) is given and a hacker is trying to meddle in the game in favor of one of the players, say player A. Therefore, each strategy of the hacker is to choose a subset of the battlefields in which player A loses and flip the results of those battlefields by hacking the system. The auditor, on the other hand, wants to secure the game by establishing extra security for up to m battlefields. If the auditor protects a battlefield that the hacker attacks, she'll catch the attacker and thus the attacker receives a payoff of 0. Otherwise, the payoff of the hacker is the total sum of the weights of the states that she hacks. The game is constant-sum and the summation of the payoffs of the players is always the total number of electoral votes. Note that both the auditor and the hacker are aware of the strategies in the Colonel Blotto instance.

In Section 7, we seek to approximate a (u,p)-maxmin strategy for the auditor in this game. We show that for a given threshold utility u, one can find in polynomial time a strategy for the auditor which is at least $(u,(1-1/e)p)$-maxmin where for any (u,p')-maxmin strategy that exists for the auditor, we have $p' \leq p$ (i.e., p is an upper bound on the probability of achieving minimum utility u). To this end, we define a benchmark LP and make a connection between the optimal solution of this LP and the highest probability for which the auditor can obtain a payoff of at least u. Next, we take the dual of the program and based on a primal-dual argument, provide a strategy for the auditor that guarantees a payoff of at least u with at least a probability of q. Finally, we make a connection between q and the solution of the benchmark LP and argue that q is at least a $1-1/e$ fraction of p. This yields the following theorem.

Theorem 7.1 [restated]. Given a minimum utility u and an instance of the auditing game, there exists a polynomial time algorithm to find a $(u,(1-1/e)p)$-maxmin strategy for the auditor; where for any $p' > p$, no (u,p')-maxmin strategy exists for the auditor.

Finally, we show a reduction from the auditing game to an instance of the Colonel Blotto game when the winner of each battlefield is specified by a given function.

3 Preliminaries

Colonel Blotto. In the Colonel Blotto game, two players A and B are competing over a number of battlefields. We denote the number of battlefields by K and denote by N and M the total troops of players A and B, respectively. Associated to each battlefield i is a weight w_i which shows the amount of profit a player wins if she wins that battlefield. This way, every strategy of a player is a partitioning of her troops over the battlefields. In the *discrete* version of Colonel Blotto, the number of troops that the players put in the battlefields must be an integer. In contrast, in the *continuous* version, a battlefield may contain any fraction of the troops. Player A wins a battlefield i if she puts more troops in that battlefield than her opponent. For simplicity, we break the ties in favor of player B, that is, if player B puts as many troops as player A's troops in a battlefield i, then she wins that battlefield and receives a payoff of w_i and player A receives a payoff of 0 on that battlefield. The final payoff of the players in this game is the total payoff that they receive over all battlefields. For a pair of pure strategies x and y, we denote by $\mathsf{u}^{\mathsf{A}}(x, y)$ and $\mathsf{u}^{\mathsf{B}}(x, y)$ the payoff of the players if they play x and y respectively. Similarly, for a pair of mixed strategies X and Y we have

$$\mathsf{u}^{\mathsf{A}}(X, Y) = \mathop{\mathbb{E}}_{x\sim X, y\sim Y}[\mathsf{u}^{\mathsf{A}}(x, y)],$$

$$\mathsf{u}^{\mathsf{B}}(X, Y) = \mathop{\mathbb{E}}_{x\sim X, y\sim Y}[\mathsf{u}^{\mathsf{B}}(x, y)].$$

Auditing game. Suppose there are K states in a presidential election race, each corresponding to a number of electoral votes. An outsider wants to hack into the system and change the outcome of the election in favor of the losing candidate. Moreover, an auditor wants to make recounts to avoid possible frauds. Refering to the players by the *hacker* and the *auditor*, we assume the simplest and full information case, that is both the hacker and the auditor have access to the exact results.[4] If the auditor conducts an inspection in a state whose winner is manipulated by the hacker, she catches the hacker and thus she wins the game (i.e., receives the maximum possible utility). On the other hand, if the hacker survives the inspection, her utility would be the number of electoral votes that she hacks in favor of her candidate.

We formally define the game as follows. Given in the input, is a set $\{s_1, s_2, \ldots, s_K\}$ of K states. For each state s_i, a value v_i is specified in the input, which is the number of its electoral votes if it is won by the losing candidate (i.e., the hacker's candidate) and zero otherwise. A limit m on the number of states that can be inspected by the auditor is also given in the input. A strategy of the hacker is a subset H of the states to hack, and a strategy of the auditor is a set A of size at most m of the states to audit. The game is constant sum and the sum of utilities is always $\sum v_i$. If the attacker is caught (i.e., if $H \cap A \neq \emptyset$), the auditor receives utility $\sum v_i$ and the attacker receives utility 0. However, if the attacker is not caught (i.e., if $H \cap A = \emptyset$), she receives utility $\sum_{s_i \in H} v_i$ and the auditor receives utility $\sum v_i - \sum_{s_i \in H} v_i$.

[4]In practice, this could be obtained by polls.

Similar to the notation that we use for the Colonel Blotto problem, for (possibly mixed) strategies x and y, we denote by $\mathsf{u}^{\mathsf{A}}(x, y)$ and $\mathsf{u}^{\mathsf{B}}(x, y)$ the payoff of the auditor and the hacker if they play x and y respectively.

(u, p)-maxmin strategies. We call a strategy of a player, a (u, p)-maxmin, if it guarantees a utility of at least u for her with probability at least p, regardless of her opponent's strategy. In other words, a strategy X is (u, p)-maxmin if for every (possibly mixed) strategy Y of the opponent we have

$$\mathop{\mathsf{Pr}}_{x\sim X, y\sim Y}[\mathsf{u}^{\mathsf{A}}(x, y) \geq u] \geq p.$$

4 Maximizing Expectation vs (u, p)-maxmin Strategies

In this section, we compare algorithms that are designed to maximize the expected payoff to the algorithms that are specifically designed to approximate a (u, p)-maxmin strategy from two perspectives: (i) the approximation factor that they guarantee; (ii) their computational complexity.

4.1 Comparison of the Approximation Factors

As it was already mentioned, our main results are algorithms that approximate the problem of finding a (u, p)-maxmin strategy. Given that at least for the Colonel Blotto problem, exact algorithms that maximize the expected payoff exist [1, 4], one might be interested in the possible approximation factor that an expectation maximizer algorithm guarantees, or to see whether designing new algorithms is really needed.

Conceptually, the main difference between an expectation maximizer algorithm and a (u, p)-maxmin optimizer, is in that the (u, p)-maxmin optimizer, in addition to the instance of the game, takes u and p as extra parameters in the input and designs a strategy accordingly, whereas the expectation maximizer returns a single strategy for the game instance. Therefore an expectation maximizer would achieve a relatively good approximation of our problem, only if it does so for all possible values of u and p.

Unsurprisingly, this is not the case. The following theorem implies that the expectation maximizers do not guarantee any constant approximation for (u, p)-maxmin problem.

THEOREM 4.1. *For any given u, p and arbitrarily small constants $0 < \alpha < 1$ and $0 < \beta < 1$, there exists an instance of Colonel Blotto (both discrete and continuous), where for any approximate $(\alpha' u, \beta' p)$-maxmin strategy that an expectation maximizer algorithm returns, either*

$\alpha' < \alpha$ *or* $\beta' < \beta$.

Proof. We construct a Colonel Blotto instance in such a way that guarantees existence of a (u, p)-maxmin strategy for player A. Then show that an expectation maximizer algorithm does not achieve anything strickly better than an $(\alpha u, \beta p)$-maxmin solution.

Construct the following Colonel Blotto instance: as usual, denote the troops of player A by N and assume that player B has $M = N/(\beta p) - 1$ troops.[5] The game has $K = 1/(\beta p) + M/(1-p)$ battlefields, where $1/(\beta p)$ of which are called *high-value* battlefields and the rest are called *low-value* battlefields. The weight of a high-value battlefield is a sufficiently large number which we denote by ∞ (suffices if $\infty > Nu$) and the weight of a low-value battlefield is u.

We first show that in the mentioned instance, player A has a (u, p)-maxmin strategy. The strategy is as follows: choose one low-value battlefield uniformly at random and put N troops in it. Since there are $M/(1-p)$ low-value battlefields, in any pure strategy, player B can put a non-zero number of troops in at most a $(1-p)$ fraction of the low-value battlefields. Hence player A wins a low-value battlefield with probability at least $1-(1-p) = p$. Since the low-value battlefields have weight u, this is a (u, p)-maxmin strategy.

However, if the goal of player A is to achieve the maximum expected payoff she will end up playing a totally different strategy. Since $M < N/(\beta p)$, there exists at least one high-value battlefield in which player B puts less than N troops, so the strategy that maximizes the expected payoff of player A is to choose a high-value battlefield uniformly at random and put N troops in it. This guarantees that with probability at least βp, player A wins a high-value battlefield, which achieves an expected payoff of at least $\beta p \cdot \infty$. It is easy to see that no other strategy of player A achieves this expected payoff. Note that this strategy gets a non-zero payoff with probability at most βp. Hence for any $\alpha u > 0$, it does not achieve any strategy that is strictly better than $(\alpha u, \beta p)$-maxmin. $\square$

However, we show that for the special case where a (u, p)-maxmin strategy, for sufficiently large values of u and p is guaranteed to exist, the solution of an expectation maximizer is a good approximation of it.

LEMMA 4.1. *Let $W := \Sigma_{i=1}^{k} w_i$ denote the total weight of the battlefields. Given that there exists a (u, p)-maxmin strategy of player A where $u \geq \frac{W}{\alpha}$ and $p \geq \frac{1}{\beta}$ for $\alpha, \beta \geq 1$ the expected maximizer returns a $(\frac{u}{2\beta}, \frac{p}{2\alpha})$-maxmin strategy.*

Proof. Let U denote the maximum expected utility of player A. Since there exist a (u, p)-maxmin strategy of player A, $U \geq u \cdot p$. Note that $u \cdot p \geq \frac{W}{\alpha\beta}$, and the maximum payoff that a player achieves from playing a strategy is W. Let q denote the probability with which player A achieves more than $\frac{u}{2\beta}$ utility in the strategy with expected utility of U. Therefore, $U < (1-q) \cdot \frac{u}{2\beta} + qW$. Assume that $q < \frac{p}{2\alpha}$ then we obtain a contradiction. In this case,

$$U < (1 - \frac{p}{2\alpha}) \cdot \frac{u}{2\beta} + \frac{p}{2\alpha} W \leq (1 - \frac{p}{2\alpha}) \frac{W}{2\alpha\beta} + \frac{W}{2\alpha\beta} < \frac{W}{\alpha\beta}$$

holds which contradicts $U \geq \frac{W}{\alpha\beta}$, so the expected maximizer strategy is a $(\frac{u}{2\beta}, \frac{p}{2\alpha})$-maxmin strategy of player A. $\square$

It needs to be mentioned that our approximation algorithms for the Colonel Blotto game take both u and p in the input, with the assumption that a (u, p)-maxmin strategy is guaranteed to exist. A more constructive approach (as taken in Theorem 7.1 for the auditing game) is to only take u (and not p) along with the game instance in the input and approximate the maximum probability with which one can achieve a guaranteed utility of u.

4.2 Comparison of their Computational Complexity

Let us again focus on the Colonel Blotto problem. It has been shown in the literature that the problem of finding a maxmin strategy that maximizes the expected payoff could be efficiently solved in polynomial time [1, 4]. The goal of this section is to illustrate why the problem of finding a (u, p)-maxmin strategy seems to be computationally harder.

In particular, we show that while the "best response" could easily be computed in polynomial time when the goal is to maximize the expected payoff [4], it is NP-hard to find the best-response for the case where the goal is to give a (u, p)-maxmin strategy.

Although the hardness of finding the best response does not necessarily imply any hardness result for the actual game, it is often a good indicator of how hard the game is to solve. We refer interested readers to the paper of Xu [35] which, for a large family of games, proves solving the actual game is as hard as finding the best response.

Assume a mixed strategy s_B of player B (as a list of pure strategies in its support and their associated probabilities), and a minimum utility u are given; the best response problem for the Colonel Blotto game,

[5]Assume for ease of exposition that all the fractions that are used in constructing the strategy are integers, otherwise consider a smaller value for β for which that holds.

denoted by $\mathcal{BR}$, is to find a pure strategy s_A of player A which maximizes $\Pr[\mathsf{u}^{\mathsf{A}}(s_A, s_B) \geq u]$.

THEOREM 4.2. *There is no polynomial time algorithm to solve $\mathcal{BR}$ unless P=NP.*

Proof. To prove this hardness, we reduce max-coverage problem to $\mathcal{BR}$. A number k and a collection of sets $S = \{S_1, S_2, \dots, S_n\}$ are given. The maximum coverage problem is to find a subset $S' \subseteq S$ of the collection, such that $|S'| \leq k$ and the number of covered elements (i.e., $|\cup_{S_i \in S'} S_i|$) is maximized.

Consider an instance of Colonel Blotto game with $|S|$ battlefields of the same weight where the first player has k troops and the second player has a sufficiently large number of troops.

Let $E = \cup_{S_i \in S} S_i$ denote the set of all items in the given max coverage instance. The support of the mixed strategy s_B of player B contains $|E|$ pure strategies, each corresponding to an item and player B plays one of these pure strategies uniformly at random (i.e., all of the pure strategies are played with the same probability). For the corresponding pure strategy to an item $e \in E$, we put $k+1$ troops in each battlefield that its corresponding set does not contain e and put zero troops in all other battlefields.

Assume that our goal is to find the best response of player A that maximizes the probability of winning at least one battlefield. Let p denote this probability. We claim $p|E|$ is indeed the solution of the max-coverage instance. Since player B puts either 0 troops or $k+1$ troops in each battlefield, it suffices for player A to put either 0 or 1 troops in each battlefield. To see this recall that player A has only k troops and clearly cannot win the battlefields in which player B puts $k+1$ troops. If player A puts more than zero troops in a battlefield, we choose its corresponding set in the max-coverage problem. Since there are at most k such battlefields, it is indeed a valid solution and it is easy to see that it maximizes the number of covered elements. □

5 Discrete Colonel Blotto

5.1 Approximating $(u, 1)$-maxmin

In this section we study the problem of finding a $(u, 1)$-maxmin strategy for player A with the maximum possible u. In this case, u is also called the guaranteed payoff that player A achieves in an instance of Colonel Blotto game. It is easy to see that the maximum guaranteed payoff of player A in a Colonel Blotto game, denoted by OPT is equal to the minimax strategy in a slightly modified version of this game which is as follows: player A first chooses a pure strategy and reveals it to her opponent, then player B, based on this observation, plays a pure strategy that maximizes her payoff. In this game if $N \leq M$, there exists no strategy of player A that guarantees a payoff more than zero for this player, therefore in this section we assume that $N > M$.

Let S_A be an arbitrary pure strategy of player A, and let strategy R_B be a best response of player B to S_A. We first give an algorithm that finds a response R'_B of player B such that $\mathsf{u}^{\mathsf{B}}(S_A, R'_B) \geq \mathsf{u}^{\mathsf{B}}(S_A, R_B) - \max_{i=1}^{K} w_i$. Then, by fixing this algorithm for player B, we are able to find a strategy of player A that guarantees at least half of the maximum guaranteed payoff of player A.

Algorithm 1 An approximation algorithm for the best response of player B

```
1: Let $w_i$ denote the weight of the $i$-th battlefield and
   let $s_i$ denote the number of troops that player A has
   in this battlefield.
2: Sort the battlefields such that for any $i \in [K-1]$,
   $\frac{w_i}{s_i} \geq \frac{w_{i+1}}{s_{i+1}}$
3: $i \leftarrow 1$
4: while $M \geq s_i$ do
5:     Player B puts $s_i$ troops in the $i$-th battlefield
6:     $M \leftarrow M - s_i$
7:     $i \leftarrow i + 1$
```

LEMMA 5.1. *Let R'_B denote the strategy that Algorithm 1 provides for player B, and let R_B denote her best response against the strategy of player A, which is denoted by S. Thus, we have $\mathsf{u}^{\mathsf{B}}(S, R'_B) \geq \mathsf{u}^{\mathsf{B}}(S, R_B) - \max_{i=1}^{K} w_i$.*

Proof. Let s_i denote the number of troops that player A puts in the i-th battlefield while playing strategy S, and let $r_i := w_i/s_i$ denote the *value* of a battlefield.

The algorithm first sorts the battlefields in the decreasing order of their values. Assume w.l.g. that the initial order is the desired one, i.e., $r_{i-1} \geq r_i$. Starting from the first battlefield in the sorted order, player B puts as many troops as player A has until there are no more troops left for player B. Let the k-th battlefield be the stopping point of the algorithm. Clearly $\mathsf{u}^{\mathsf{B}}(S, R'_B) = \Sigma_{i=1}^{k-1} w_i$. Moreover, one can easily see that $\mathsf{u}^{\mathsf{B}}(S, R_B) \leq \Sigma_{i=1}^{k} w_i$. Therefore $\mathsf{u}^{\mathsf{B}}(S, R'_B) \geq \mathsf{u}^{\mathsf{B}}(S, R_B) - \max_{i=1}^{K} w_i$ as desired. □

THEOREM 5.1. *There exists a polynomial time algorithm that gives a $(u/2, 1)$-maxmin strategy of player A, assuming that u is the maximum guaranteed payoff of player A.*

Proof. We first define a new optimization problem, then we prove that the solution to that problem is also a

2-approximation solution for the maximum guaranteed payoff of player A (i.e., OPT). For any strategy s of player A, let $U(s)$ denote the payoff that player A achieves if the response of player B to strategy s is determined by Algorithm 1. The optimization problem is to find a strategy s of player A that maximizes $U(s)$. Let S denote the set of all possible strategies of player A, and let $\text{OPT}' = \max_{s \in S} U(s)$. Since this is a constant-sum game, by Lemma 5.1, $\text{OPT}' \leq \text{OPT} + \max_{i=1}^{K} w_i$. Moreover, $\text{OPT} \geq \max_{i=1}^{K} w_i$ since $N > M$, and player A can win the battlefield with maximum weight by putting all her troops in that battlefield. Therefore, $\text{OPT}' \leq 2\text{OPT}$, and to prove this lemma it suffices to give an algorithm that finds OPT'. Algorithm 2 finds OPT' via dynamic programming.

Algorithm 2 A 2-approximation algorithm for the guaranteed payoff of player A

```
1: function ApproximateGuaranteedPayoff
2:   Let $w_k$ denote the weight of $k$-th battlefild.
3:   $s \leftarrow -\infty$
4:   for $k$ in $[K]$ do
5:     for $m$ in $[N]$ do
6:       $s \leftarrow \max(s, \text{FindBestPayoff}(m, k))$
7:     return s
8: function FindBestPayoff$(m, k)$
9:   $r \leftarrow w_k/m$
10:  $U[0][0][0] \leftarrow 0$
11:  for any $i$ in $[K]$, $a$ in $[0, N-m]$ and $b$ in $[0, M]$ do
12:    $U[i][a][b] \leftarrow -\infty$
13:    if $i = k$ then
14:      $U[i][a][b] \leftarrow U[i-1][a-m][b] + w_i$
15:    else
16:      for $t$ in $\{0, \ldots, \min(a, b, \lceil w_i/r \rceil - 1)\}$ do
17:        $U[i][a][b] \leftarrow$
               $\max(U[i][a][b], U[i-1][a-t][b-t])$
18:      if $a \geq \lceil w_i/r \rceil$ then
19:        $U[i][a][b] \leftarrow$
               $\max(U[i][a][b], U[i-1][a - \lceil w_i/r \rceil][b] + w_i)$
20:  return $\max_{i=0}^{m-1} U[K][N][M-i]$
```

For any strategy s of player A, let s_i denote the number of troops that player A puts in the i-th battlefield and let $B(s)$ denote the set of battlefields that player A wins given that the response of player B is determined by Algorithm 1. In addition let $b(s) := \arg\max_{i \in B(s)} \frac{w_i}{s_i}$ be the first battlefield in which player B loses (in the sorted list of battlefields in Algorithm 1) and let $t(s) := s_{b(s)}$. Furthermore, let $S(k, m)$ be a subset of strategies of player A where $b(s) = k$ and $t(s) = m$ for any $s \in S(k, m)$. Function FindBestPayoff, for given inputs $1 \leq k \leq K$ and $0 \leq m \leq N$ finds $\max_{s \in S(k,m)} U(s)$. Finally, in function ApproximateGuaranteedPayoff, we find OPT' by calling function FindBestPayoff for all sets $S(k, m)$ such that $1 \leq k \leq$ and $0 \leq m \leq N$, and returning the maximum answer.

Let $U^A(j, s)$ denote the payoff that player A achieves in the j-th battlefield if the response of player B to strategy s is determined by Algorithm 1, and let $P(i, a, b)$ denote the set of strategies of player A such that for any $s \in P(i, a, b)$, $\Sigma_{j=1}^{i} s_j = a$, and in the response to s that is determined by Algorithm 1, player B puts exactly b troops in the first i battlefields.

CLAIM 5.1. *In Algorithm 2,*

$$U[i][a][b] = \max_{s \in S(k,m) \cap P(i,a,b)} \Sigma_{j=1}^{i} U^A(j, s).$$

Proof. We use induction on i.

(i). Induction hypothesis: for any $1 \leq i \leq K$, and any arbitrary a' and b' such that $0 \leq a' \leq N$ and $0 \leq b' \leq M$, $U[i][a'][b'] = \max_{s \in S(k,m) \cap P(i,a',b')} \Sigma_{j=1}^{i} U^A(j, s)$.

(ii). Base case: $U[0][0][0] = 0$ and all the other cells for $i = 0$ are undefined (we assume that the value of any undefined cell is equal to $-\infty$).

(iii). For the induction step we prove the correctness of hypothesis for $i+1$: It is easy to verify if $i + 1 = k$ since by the constraints, player A puts exactly m troops and player B puts no troops in the k-th battlefield, therefore $U[i][a][b] = U[i-1][a-m][b] + w_i$. However, if $i + 1 \neq k$, there are different possible cases for the number of troops that player A puts in this battlefield, denoted by s_i. As a result she either gains w_{i+1} or 0 utility in this battlefield. By Algorithm 1, if $\frac{w_{i+1}}{s_{i+1}} \geq \frac{w_k}{m}$ player B wins this battlefield, otherwise she loses it. In other words, if $s_{i+1} < \frac{m \times w_{i+1}}{w_k}$, by Algorithm 1, player B, puts s_{i+1} troops in this battlefield and wins it. These cases are handled in line 17 of the function FindBestPayoff. In addition, if $s_{i+1} \geq \frac{m \times w_{i+1}}{w_k}$, by Algorithm 1, player B puts no troop in it so player A wins it. Also, player A never puts more than $\lceil \frac{m \times w_{i+1}}{w_k} \rceil$ in this battlefield since any number of troops more than this results in the same payoff (winning w_{i+1}). This case is handled in line 19 of the algorithm. To sum up, the induction step is proved.

□

To complete the proof, it suffices to prove that function FindBestPayoff correctly finds $\max_{s\in S(k,m)} U(s)$. Note that the output of function FindBestPayoff is $\max_{i=0}^{m-1} U[K][N][M-i]$, hence we need to prove

$$\max_{s\in S(k,m)} U(s) = \max_{i=0}^{m-1} U[K][N][M-i]. \tag{5.1}$$

Claim 5.1 is indeed the main technical ingredient that we use to prove (5.1). The first equality in the following equation comes from Claim 5.1.

$$\begin{aligned} &\max_{i=0}^{m-1} U[K][N][M-i] \\ &= \max_{i=0}^{m-1}\Big(\max_{s\in S(k,m)\cap P(K,A,B-i)} \Sigma_{j=1}^{K} U^A(j,s)\Big) && (5.2)\\ &= \max_{i=0}^{m-1}\Big(\max_{s\in S(k,m)\cap P(K,A,B-i)} U(s)\Big) && (5.3)\\ &= \max_{s\in S(k,m)\cap(\cup_{i=0}^{m-1}P(K,A,B-i))} U(s). && (5.4)\end{aligned}$$

Note that player B may have at most $m-1$ unused troops since otherwise she could use them to win battlefield k which contradicts the assumption of this function. This implies $S(k,m) \subseteq \cup_{i=0}^{m-1} P(K,A,B-i)$ and therefore by (5.4) we obtain (5.1) as desired. □

5.2 Approximating (u,p)-maxmin

In this section, we present a polynomial time algorithm for approximating a (u,p)-maxmin strategy in the Colonel Blotto game. More precisely, given that there exists a (u,p)-maxmin strategy for player A, we present a polynomial time algorithm to find a $(O(u/(\log K)), O(p))$-maxmin strategy. In Section 5.2.1 we study the problem for a special case where $w_i = 1$ for all $i \in [K]$. We show that in this case, if K and N are large enough then player A can win a fraction of the battlefields proportional to the ratio of N over M. We also argue that in some cases, no strategy can be (u,p)-maxmin for player A with $u,p>0$. We then use these observations to obtain a $(u/8, p/4)$-maxmin strategy for player A in the uniform setting and a $(u/(16(\lceil\log K\rceil+1)), p/8)$-maxmin strategy for the general setting.

5.2.1 The Case of Uniform Weights

A special case of the problem is when all weights are uniform. We study this case in this section. We assume w.l.g. that all weights are equal to 1 since one can always satisfy this condition by scaling the weights. Given that there exists a (u,p)-maxmin strategy for player A, we present a strategy for player A that is at least $(u/8, p/4)$-maxmin. Recall that we denote the number of troops of players A and B by N and M, respectively. Our first observation is that if both u and p are non-zero then $p \le 4(K - \lfloor M/N\rfloor)/K$ holds.

Lemma 5.2. *Given that there exists a (u,p)-maxmin strategy for player A with $u,p>0$, then $p \le 2(K-\lfloor M/N\rfloor)/K$ holds.*

Proof. We assume w.l.g. that $\lfloor M/N\rfloor \ge 1$ (otherwise $p \le 2(K-\lfloor M/N\rfloor)/K = 2$ trivially holds). Also $K \ge 2\lfloor M/N\rfloor$ implies $2(K-\lfloor M/N\rfloor)/K \ge 1$ which yields $p \le 2(K-\lfloor M/N\rfloor)/K$. Suppose for the sake of contradiction that the conditions doesn't hold. Therefore we have $K < 2\lfloor M/N\rfloor$ and also $p > 2(K-\lfloor M/N\rfloor)/K$. We show that in this case, no strategy of player A can be (u,p)-maxmin for $u>0$.

If $K \le \lfloor M/N\rfloor$ then player B can put N troops in all battlefields and always prevent player A from winning any battlefield. Thus, $K > \lfloor M/N\rfloor$. Now if player B plays the following strategy, the probability that player A wins a single battlefield is smaller than p: randomly choose $2(K-\lfloor M/N\rfloor)$ battlefields and put $N/2$ ($=\lfloor N/2\rfloor$ in the discrete version of the game) troops in them and put N troops in the rest of the battlefields. Recall that $K < 2\lfloor M/N\rfloor$ and thus $2(K-\lfloor M/N\rfloor)$ does not exceed the number of battlefields. This requires at most the total number of

$$\begin{aligned} 2(K-\lfloor M/N\rfloor)N/2 + (2\lfloor M/N\rfloor - K)N &\\ \le \lfloor M/N\rfloor(2N-2N/2) + K(2N/2-N) &\\ = \lfloor M/N\rfloor(2N-N) + K(N-N) &\\ = \lfloor M/N\rfloor N &\\ \le M &\end{aligned}$$

troops. Notice that in order for a strategy of player A to win a battlefield, it needs to put more than $N/2$ troops in that battlefield. Moreover, each pure strategy of player A can put more than $N/2$ troops in at most one battlefield. Thus, a pure strategy of player A gains a non-zero payoff only if player B puts at most $N/2$ troops in that chosen battlefild. This probability is bounded by $2(K-\lfloor M/N\rfloor)/K$ for each battlefield due to the strategy of player B. Therefore, player A can get a non-zero payoff with probability no more than $2(K-\lfloor M/N\rfloor)/K$. This contradicts the existence of a (u,p)-maxmin strategy for player A with $u>0$ and $p > 2(K-\lfloor M/N\rfloor)/K$. □

Although we consider Lemma 5.2 in the uniform and discrete setting, the proof doesn't rely on any of these conditions. Thus, Lemma 5.2 holds for the general setting (both continuous and discrete). Based on Lemma 5.2, we present a simple algorithm and show that the strategy obtained from this algorithm is at least $(u/8, p/4)$-maxmin. In our algorithm, if

$K \leq 2\lfloor M/N \rfloor$ then we randomly select a battlefield and put N troops in it. Otherwise, we find the smallest t such that $t(\lfloor K/2 \rfloor + 1) > M$ and put t troops in $\lfloor N/t \rfloor$ battlefields uniformly at random. The logic behind this is that we choose a large enough t to make sure player B can put t troops in no more than $\lfloor K/2 \rfloor$ battlefields. Therefore, when player A puts t troops in a random battlefield, we can argue that she wins that battlefield with probability at least $1/2$ regardless of player B's strategy. We use this fact to show that player A wins at least $\lceil 1/8 \min\{N, K, K(N/M)\} \rceil$ battlefields with probability at least $1/2$. Finally we provide almost matching upper bounds to show the tightness of our solution. We first provide a lower bound in Lemma 5.3 on the payoff of this strategy against any response of player B.

Algorithm 3 An algorithm to find a $(u/8, p/4)$-maxmin strategy for player A

```
 1: if $K < 2\lfloor M/N \rfloor$ then
 2:     Choose a battlefield $i$ uniformly at random.
 3:     Put $N$ troops in battlefield $i$.
 4: else
 5:     $t \leftarrow 0$.
 6:     while $t(\lfloor K/2 \rfloor + 1) \leq M$ do
 7:         $t \leftarrow t + 1$
 8:     if $N \geq Kt$ then
 9:         put $t$ troops in all battlefields
10:     else
11:         Choose $\lfloor N/t \rfloor$ battlefields $a_1, a_2, \ldots, a_{\lfloor N/t \rfloor}$ uniformly at random.
12:         Put $t$ troops in every battlefield $a_i$.
```

LEMMA 5.3. *The strategy of Algorithm 3 provides the following guarantees for player A in any instance of discrete Colonel Blotto game:*

- *If $K < 2\lfloor M/N \rfloor$, player A wins a battlefield with probability at least $(K - \lfloor M/N \rfloor)/K$.*
- *If $K \geq 2\lfloor M/N \rfloor$, player A wins $\lceil 1/8 \min\{N, K, K(N/M)\} \rceil$ battlefields with probability at least $1/2$.*

Proof. We prove each of the cases separately. If $K < 2\lfloor M/N \rfloor$, player A's strategy is to randomly choose a battlefield and put all her troops in it. Notice that any strategy of player B can put N troops in at most $\lfloor M/N \rfloor$ battlefields. Thus, with probability at least $(K - \lfloor M/N \rfloor)/K$, player B puts fewer than N troops in the selected battlefield of player A and thus player A wins that battlefield.

If $K \geq 2\lfloor M/N \rfloor$, Algorithm 3 finds the smallest t such that $t(\lfloor K/2 \rfloor + 1) > M$ and puts t troops in $\min\{K, \lfloor N/t \rfloor\}$ randomly selected battlefields. As we mentioned earlier, since $t(\lfloor K/2 \rfloor + 1) > M$, player B can put t troops in no more than $\lfloor K/2 \rfloor$ battlefields and thus she puts fewer than t troops in at least half of the battlefields. If $K \leq \lfloor N/t \rfloor$, player A puts t troops in all battlefields and since player B can protect at most $\lfloor K/2 \rfloor$ of the battlefields, player A wins at least $\lceil K/2 \rceil$ battlefields with probability 1. If $K > \lfloor N/t \rfloor$, player A wins any of the selected battlefields with pobability at least $1/2$ and therefore, with probability at least $1/2$, player A wins at least $\lceil \lfloor N/t \rfloor /2 \rceil$ of the $\lfloor N/t \rfloor$ battlefields wherein she puts t troops. The rest of the proof follows from a mathematical observation. In the interest of space we omit the proof of Observation 5.1 here.

OBSERVATION 5.1. *Let N, M, and K be three positive integer numbers such that $K \geq 2\lfloor M/N \rfloor$ and t be the smallest integer number such that $t(\lfloor K/2 \rfloor + 1) > M$. Then we have*

$$\lceil \lfloor N/t \rfloor /2 \rceil \geq \lceil 1/8 \min\{N, K(N/M)\} \rceil.$$

□

To show that Algorithm 3 provides a strategy competitive to that of the optimal, we present two upper bounds for each of the cases separately.

LEMMA 5.4. *Given that there exists a (u, p)-maxmin strategy for player A with non-zero u and p in an instance of discrete Colonel Blotto with uniform weights, for $K < 2\lfloor M/N \rfloor$ we have $u \leq 2$ and $p \leq 2(K - \lfloor M/N \rfloor)/K$.*

Proof. $p \leq 2(K - \lfloor M/N \rfloor)/K$ follows directly from Lemma 5.2. Next we argue that in this case u is also bounded by 2. To this end, suppose that player B puts $\lfloor M/K \rfloor$ troops in every battlefield. This way, in order for player A to win a battlefield, she should put at least $\lfloor M/K \rfloor + 1 \geq \lceil M/K \rceil$ troops in that battlefield. Since $K < 2\lfloor M/N \rfloor$, then $\lceil M/K \rceil \geq N/2$ and thus player A can never achieve a payoff more than 2. □

LEMMA 5.5. *Given that there exists a (u, p)-maxmin strategy for player A with non-zero u and p in an instance of discrete Colonel Blotto with uniform weights, for $K \geq 2\lfloor M/N \rfloor$ we have $u \leq \min\{K, N, K(N/M)\}$.*

Proof. $u \leq K$ and $u \leq N$ hold since there are at most K battlefields to win and player A can put non-zero troops in at most N of them. Therefore, the only non-trivial part is to show $u \leq K(N/M)$. We show that no strategy of player A can achieve a payoff more than $K(N/M)$ with non-zero probability. To this end, suppose that player B puts $\lfloor M/K \rfloor$ troops in every battlefield. This way, in order for player A to win a battlefield, she has to put at least $\lfloor M/K \rfloor + 1 \geq \lceil M/K \rceil$ troops in that battlefield. Thus, any strategy of player A wins no more than $N/\lceil M/K \rceil \leq K/(M/N)$ battlefields. Therefore, u is bounded by $K/(M/N)$. □

Lemma 5.3 along with the upper bounds provided in Lemmas 5.4 and 5.5 proves that the strategy of Algorithm 3 is competitive with the optimal strategy of player A. As a corollary of Lemmas 5.3, 5.4, and 5.5, we get the following theorem.

THEOREM 5.2. *Given that there exists a (u, p)-maxmin strategy for player A in an instance of discrete Colonel Blotto with uniform weights, Algorithm 3 provides a $(u/8, p/2)$-maxmin strategy.*

5.2.2 The General Setting We showed in Section 5.2.1 that when all the weights are equal to 1, there is a polynomial time solution for finding an approximate (u, p)-maxmin solution for a given u and p. In this section, we extend this result to the case of non-uniform weights. The main ingredient of our proposal is a mathematical argument which we state in Lemma 5.6.

LEMMA 5.6. *Given n non-negative values $a_1 \geq a_2 \geq a_3 \geq \ldots \geq a_n$, there exists a k with $1 \leq k \leq n$ such that*

$$ka_k \geq 1/(\lceil \log n \rceil + 1) \sum_{i=1}^{n} a_i.$$

Proof. We assume w.l.g. that $n = 2^k - 1$ for some k (otherwise we add enough 0's to the end of the sequence to satisfy this condition). We divide the sequence into $\lceil \log n \rceil$ buckets as follows: The first bucket contains only a_1. The second bucket contains a_2 and a_3. More precisely, the i'th bucket contains all variables from $a_{2^{i-1}}$ to $a_{2^i - 1}$. Since the variables are non-decreasing, for each bucket i, sum of the variables inside it is upper bounded by $2^{i-1}a_{2^{i-1}}$. Since we have $\lceil \log n \rceil$ buckets, the total sum of the values of at least one bucket is no less than $(\sum a_i)/\lceil \log n \rceil$ and therefore at least for one i we have $2^{i-1}a_{2^{i-1}} \geq (\sum a_i)/\lceil \log n \rceil$. An extra $+1$ appears in the guarantee because of the adjustment to n that we made in the beginning of the proof. □

Given that player A has a (u, p)-maxmin strategy, we present a randomized strategy for player A and show that this strategy is at least $(u/(16(\lceil \log K \rceil + 1)), p/8)$-maxmin. In our strategy, we split the battlefields into two categories *high-value* and *low-value*. A battlefield is called high-value if the winner of that battlefield obtains a payoff of at least $u/(16(\lceil \log n \rceil + 1))$ and low-value otherwise. We denote the high-value battlefields by $A = \{a_1, a_2, \ldots, a_{|A|}\}$ and the low-value battlefields by $B = \{b_1, b_2, \ldots, b_{|B|}\}$. We assume w.l.g. that both a_i's and b_i's are sorted in non-decreasing order according to the weights of the battlefields. In other words, $w_{a_1} \geq w_{a_2} \geq \ldots \geq w_{a_{|A|}}$ and $w_{b_1} \geq w_{b_2} \geq \ldots \geq w_{b_{|B|}}$. Let M' be the smallest number of troops that player B needs to put in the high-value battlefields to make sure player A wins any of such battlefields with probability less than p due to the upperbound of Lemma 5.2 . In our proposal, with probability 1/2 we play Algorithm 3 on the high-value battlefields with the assumption that player B has $M' - 1$ troops. Also, with probability 1/2 we play Algorithm 3 on a prefix of battlefields $b_1, b_2, \ldots, b_k$ as if player B had $M - M'$ troops. Note that in both cases, we assume that the weights of all battlefields are equal to 1 when using Algorithm 3. If any of A or B is empty, we only play on the non-empty set. If $M' > M$, we only play on battlefields of set A. A formal description of our proposal is given in Algorithm 4.

Algorithm 4 An algorithm to find a $(u/(16(\lceil \log K \rceil + 1)), p/4)$-maxmin strategy for player A

```
 1: $A = \{a_1, a_2, \ldots, a_{|A|}\} \leftarrow$ the set of battlefield with weight at least $u/(16(\lceil \log n \rceil + 1))$
 2: $B = \{b_1, b_2, \ldots, b_{|B|}\} \leftarrow$ the set of battlefield with weight less than $u/(16(\lceil \log n \rceil + 1))$
 3: $M' \leftarrow 0$
 4: while $2(|A| - \lfloor M'/N \rfloor)/|A| \geq p$ do
 5:     $M' \leftarrow M' + 1$
 6: coin $\leftarrow$ either 0 or 1 with equal probability
 7: if coin $= 0$ and $|A| = 0$ then
 8:     coin $\leftarrow 1$
 9: if coin $= 1$ and ($|B| = 0$ or $M' > M$) then
10:     coin $\leftarrow 0$
11: if coin $= 0$ then
12:     Run Algorithm 3 on the battlefields $a_1, a_2, \ldots, a_{|A|}$ with $N$ and $M' - 1$ troops for the players.
13: else
14:     if $N \geq M - M'$ then
15:         $k \leftarrow \arg\max_{i=1}^{\min\{|B|, N\}} iw_{b_i}$
16:     else
17:         $k \leftarrow \arg\max_{i=1}^{\min\{|B|, M-M'\}} iw_{b_i}$
18:     Run Algorithm 3 on the battlefields $b_1, b_2, \ldots, b_k$ with $N$ and $M - M'$ troops for the players.
```

Our claim is that Algorithm 4 is $(u/(16(\lceil \log K \rceil + 1)), p/8)$-maxmin. Before we provide a formal proof, we mention the high level idea briefly. Notice that in Algorithm 4 we flip a coin and attack each set of the battlefields with probability 1/2. The best response of player B is always a pure strategy. Such a pure strategy either puts fewer than M' troops in the high-value battlefields or no more than $M-M'$ troops in the low-value battlefields. In each case, we argue that the strategy of Algorithm 4 performs well with probability at least $p/4$.

THEOREM 5.3. *Given that a (u,p)-maxmin strategy exists for player A in an instance of discrete Colonel Blotto, Algorithm 4 provides a $(u/(16(\lceil\log K\rceil+1)), p/4)$-maxmin strategy.*

Proof. In order to show that the strategy obtained from Algorithm 4 is $(u/(16(\lceil\log K\rceil+1)), p/4)$-maxmin, we show that it achieves a payoff at least $(16(\lceil\log K\rceil+1))$ with probability at least $p/4$ against any pure strategy of player B. To this end, we consider two cases. Either the pure strategy of player B puts fewer than M' troops in the high-value battlefields, or puts no more than $M-M'$ troops in the low-value battlefields. We investigate each of the possibilities in the following:

Fewer than M' troops in the high-value battlefields: Line 4 of Algorithm 4 terminates right after $2(|A|-\lfloor M'/N\rfloor)/|A| < p$ happens. Therefore, for $M'-1$ troops we have $2(|A|-\lfloor(M'-1)/N\rfloor)/|A| \ge p$. It follows from Lemma 5.3 that if player B puts $M'-1$ (or fewer) troops in these battlefields and player A plays according to Algorithm 3, she wins at least a battlefield with probability at least $\min\{1/2, (|A|-\lfloor(M'-1)/N\rfloor)/|A|\}$ which is at least $p/2$. Moreover, the payoff she achieves from winning any of the battlefields is at least $u(16(\lceil\log K\rceil+1))$. Since the strategy of Algorithm 4 plays on the high-value battlefields with probability (at least) $1/2$, this guarantees a payoff of $u/(16(\lceil\log K\rceil+1))$ with probability at least $p/4$.

No more than $M-M'$ troops in the low-value battlefields: We first provide some lower bounds on sum of the weights of the low-value battlefields. We show that unless the total weight of certain battlefields is lower bounded by fixed values, player B can play in a way to prevent player A from obtaining a payoff of at least u with probability at least p. Recall that due to Line 4 of Algorithm 4, $2(|A|-\lfloor M'/N\rfloor)/|A| < p$ holds. It follows from Lemma 5.2 that player B can put M' troops in the high-value battlefields to make sure player A wins no high-value battlefield with probability at least $1-p$. Therefore, efficient allotcations of the remaining $M-M'$ troops of player B to the low-value battlefields imply:

- If $N \le |B|$ then $w_{b_1}+w_{b_2}+\ldots+w_{b_N} \ge u$ since a (u,p)-maxmin stratey of the player A should obtain at least a payoff of u from the low-value battlefields.
- $w_{b_1}+w_{b_2}+\ldots+w_{b_{|B|}} \ge u(M-M')/N$ since otherwise the following strategy of player B can prevent player A from achieving a payoff of u with non-zero probability: Let $W=\sum w_{b_i}$ be sum of the weights of the low-value battlefields. Put $\lfloor(M-M')w_{b_i}/W\rfloor$ troops in every battlefield b_i. Note that this requires no more than $M-M'$ troops since
$$\begin{aligned}\sum(M-M')w_{b_i}/W &= (M-M')(\sum w_{b_i})/W\\ &= (M-M')W/W = M-M'.\end{aligned}$$
In addition to this, in order for player A to win a battlefield b_i, she has to put at least $\lfloor(M-M')w_{b_i}/W\rfloor+1 \ge \lceil(M-M')w_{b_i}/W\rceil$ troops in that battlefield. Therefore, the ratio of the payoff over the number of necessary troop to win for each battlefield is at least $W/(M-M')$ and thus player A can obtain no more than $WN/(M-M')$ payoff. This implies $W \ge u(M-M')/N$ provided that there exists a (u,p)-maxmin strategy for player A.
- If $N \le M-M' \le |B|$ then $w_{b_1}+w_{b_2}+\ldots+w_{b_{|M-M'|}} \ge u(M-M')/N$: This follows from an argument similar to the one just stated. Suppose for the sake of contradiction that $w_{b_1}+w_{b_2}+\ldots+w_{b_{|M-M'|}} < u(M-M')/N$. We define $W=\sum_{i=1}^{M-M'} w_{b_i}$ and propose the following strategy for player B. For each $1 \le i \le M-M'$, put $\lfloor(M-M')w_{b_i}/W\rfloor$ troops in battlefield b_i and put no troops in the rest of the low value battlefields. This way, for each such battlefield the ratio of payoff per troop necessary to win that battlefield is bounded by $W/(M-M')$ for player A. Moreover, since we assume the weights of the battlefields are non-decreasing, we have $w_{b_i} \le W/(M-M')$ for $i > M-M'$ and thus winning each of those battlefields has a payoff of at most $W/(M-M')$ for player A. Therefore, no strategy of player A can achieve a payoff more than $NW/(M-M')$ against such a strategy of player B. This implies that $w_{b_1}+w_{b_2}+\ldots+w_{b_{|M-M'|}} \ge u(M-M')/N$ provided that there exists a (u,p)-maxmin strategy for player A in this game.

The above lower bounds along with Lemma 5.6 imply that
$$kw_{b_k} \ge u\max\{1,(M-M')/N\}/(\lceil\log|B|\rceil+1).$$
Moreover, $k < 2\lfloor(M-M')/N\rfloor$ cannot hold since the weight of each low-value battlefield is bounded by $u/(16\lceil\log K\rceil+1)$. Thus, if player A plays Algorithm 3 on battlefields $b_2, b_2, \ldots, b_k$ she wins at least $\lceil 1/8\min\{N,k,k(N/(M-M'))\}\rceil$ of them with probability at least $1/2$ due to Lemma 5.3. This provides

player A with a payoff of at least $u/(16(\lceil \log K \rceil + 1))$ against any pure strategy of player B that puts no more than $M - M'$ troops in the low-value battlefields. Since Algorithm 4 plays on the low-value battlefields with probability at least 1/2, this guarantees a payoff of $u/(16(\lceil \log K \rceil + 1))$ with probability at least 1/4. □

6 Continuous Colonel Blotto

In this section we study the *continuous* version of the Colonel Blotto game. In this version we relax the assumption that the number of troops that a player puts in a battlefield is an integer. In fact, for certain applications (e.g., where money is the resource that is to be distributed among battlefields) the continuous model is more realistic.[6] We first show in Section 6.1 that it is possible to find a $(u, 1)$-maxmin strategy for each player in the continuous Colonel Blotto in polynomial time or report that no such strategy exists. We also give an approximation algorithm for (u, p)-maxmin in the continuous version of the game in Section 6.2. Our algorithm provides a $(u/8, p/8)$-maxmin strategy for any instance of continuous Colonel Blotto, given that there exists a (u, p)-maxmin for that instance.

6.1 An Exact Algorithm for $(u, 1)$-maxmin In this section we provide a polynomial time algorithm to find a $(u, 1)$-maxmin strategy for player A. The formal statement of the theorem is as follows.

Theorem 6.1. *For any given instance of continuous Colonel Blotto and any given u, there exists a polynomial time algorithm to either find a $(u, 1)$-maxmin strategy or report that no $(u, 1)$-maxmin strategy exists.*

The algorithm is a linear program. It is worth mentioning that using this LP, one can search over u to find the maximum payoff that can be guaranteed for player A (i.e., her pure maxmin strategy).

Let $W := \sum_{i=1}^{k} w_i$ denote the total weight of all battlefields. We define a subset $S = \{b_1, \ldots, b_k\}$ of the battlefields to be *critical* if the total weight of the battlefields in it is more than $W - u$ (i.e., $\sum_{i \in S} w_i > W - u$). The following lemma is the main observation behind the LP.

Lemma 6.1. *A strategy S^A is an $(u, 1)$-maxmin strategy of player A if and only if for any critical subset S of the battlefields, strategy S^A puts more than M (the total troops of player B) troops into the battlefields in S.*

[6]To remain consistent to the rest of the paper, we use "troops" to refer to the resources even for the continuous version of the game.

Proof. Assume S^A is a $(u, 1)$-maxmin strategy of player A and assume for the sake of contradiction that there exists a critical subset S of the battlefields in which S^A does not put more than M troops. Clearly, player B is able to win any battlefield $i \in S$ by putting the same number of troops that S^A puts in it. This means player A is only able to win the battlefields that are not in S, but since S is a critical subset of the battlefields, the total weight of the battlefields that are not in S is less than u — which contradicts the assumption that S^A is an $(u, 1)$-maxmin strategy.

For the other direction, consider a strategy S^A that puts more than M troops in any critical subset of the battlefields, we prove S^A is indeed an $(u, 1)$-maxmin strategy. Again, for the sake of contradiction, assume this is not the case and there exists a strategy S^B of player B that gets a payoff of more than $W - u$ agains S^A. The contradiction is that the subset of battlefields that player B wins is a critical subset in which player A has put at most M troops. □

We are now ready to explain the LP. There are K variables $x_1, \ldots, x_K$ where variable x_i denotes the number of troops that we put in the i-th battlefield. Apart from the constraints that enforce these variables correspond to a valid K-partitioning of N, for each critical subset, there is a constraint that ensures the total number of troops in this subset of battlefields is more than M. Since there maybe exponentially many critical subsets, we use the ellipsoid method to solve it. The linear program is formally given as LP 1.

$$\text{(LP 1)} \qquad \begin{aligned} & x_i \geq 0 \qquad \forall i : 1 \leq i \leq K \\ & x_1 + x_2 + \ldots + x_K = N \\ & \sum_{j \in S} x_j > M \qquad \forall \text{ critical subset } S \end{aligned}$$

To apply the ellipsoid method, we need a separation oracle that decides whether a given assignment $\hat{x} = \langle x_1, \ldots, x_K \rangle$ is a valid solution of LP 1, and if not, finds a violated constraint. The separation oracle first verifies whether the first two constraints are violated or not, and if not runs the following instance of knapsack. The knapsack has capacity M, and for any battlefield i, there is an item with volume x_i and value w_i. Clearly, the solution of this knapsack problem is the best response of player B to the strategy of player A that corresponds to $\hat{x}$. It suffices to check whether in this best response, the battlefields that player B wins form a critical subset or not. If they do not form a critical subset, by Lemma 6.1, $\hat{x}$ is a valid solution of LP 1, and if they form a critical subset, the constraint that corresponds to this critical subset is violated.

6.2 An Approximation Algorithm for (u, p)-maxmin In this section we present a polynomial time algorithm to find a $(u/8, p/8)$-maxmin algorithm of player A in any instance of continuous Colonel Blotto given that there exists a (u, p)-maxmin strategy. In this algorithm we partition the battlefields into two sets of low-value and high-value battlefields. If both sets are non-empty, player A either puts all her troops in high-value battlefields with probability 1/4 or she puts all the troops in low-value battlefields with probability 3/4. The strategies that she plays in any of these cases are different. Algorithm 5 contains the details of the algorithm. We also prove that this algorithm gives a $(u/8, p/8)$-maxmin strategy of player A in Theorem 6.2.

Algorithm 5 An algorithm to find a $(u/8, p/8)$-maxmin strategy of player A in continuous Blotto

```
1: $A = \{a_1, a_2, \ldots, a_{|A|}\} \leftarrow$ the set of battlefield with weight at least $u/8$
2: $B = \{b_1, b_2, \ldots, b_{|B|}\} \leftarrow$ the set of battlefield with weight less than $u/8$
3: $s \leftarrow 0$
4: $\delta \leftarrow \{\}$
5: $\Delta \leftarrow \{\}$
6: for $b$ in $B$ do
7:     if $s \geq u/2$ then
8:         Add $\delta$ to $\Delta$.
9:         $\delta \leftarrow \{\}$
10:        $s \leftarrow 0$
11:    Add b to set $\delta$.
12:    $s \leftarrow s +$ weight of battlefield $b$
13: $M' \leftarrow N(|A|(1 - p/2) + 1)$
14: $coin \leftarrow 0$
15: With 3/4 probability set $coin$ to 1.
16: if $coin = 0$ and $|A| = 0$ then
17:     $coin \leftarrow 1$
18: if $coin = 1$ and ($|B| = 0$ or $M' > M$) then
19:     $coin \leftarrow 0$
20: if $coin = 0$ then
21:     Choose a battlefield $a$ uniformly at random from set $A$.
22:     Put $N$ troops in battlefield $a$.
23: else
24:     Choose a set $\delta$ from $\Delta$ uniformly at random.
25:     for $b$ in $\delta$ do
26:         Put $N/w_b$ troops in battlefield $b$.
```

Theorem 6.2. *Given that a (u, p)-maxmin strategy exists for player A in an instance of continuous Colonel Blotto, Algorithm 5 provides a $(u/8, p/8)$-maxmin strategy.*

Proof. The proof is structurally similar to that of Theorem 5.3. However, because of the inherent differences of the continuous case and the discrete case, we are able to get a much better guarantee on the guaranteed utility for the continuous case.

To prove that Algorithm 5 achieves a $(u/8, p/8)$-maxmin strategy, we would have to show that with probability at least $p/8$, it gets a payoff of at least $u/8$ against any pure strategy of player B. To do so, we consider two cases. The first case is that player B puts fewer than M' troops in *high-value* battlefields. The second case is that player B puts no more than $M - M'$ troops in the *low-value* battlefields. The following two paragraphs consider these two cases.

Case 1 (fewer than M' troops in the high-value battlefields): Denote by M_h the number of troops that player B puts in high-value battlefields. Note that $M' = N(|A|(1 - p/2) + 1)$ and in this case $M_h < M'$, which implies $M_h < N(|A|(1 - p/2) + 1)$ and that there at most $|A|(1 - p/2)$ battlefields that player B can put at least N troops. As such, when player A chooses, uniformly at random, a high-value battlefield with probability at least $p/2$, player B puts less than N troops in that battlefield and player A wins it. This means that player A wins at least a battlefield with probability at least $p/2$. Moreover, the payoff of each of these battlefields is at least $u/8$, therefore, the minimum payoff of $u/8$ with probability at least $p/8 = (2/4)(1/4)$ is guaranteed. The latter 1/4 term comes from the fact that player A puts her troops in high-value battlefields with 1/4 probability (Line 15).

Case 2 (no more than $M - M'$ troops in the low-value battlefields): At first, we show that the sum of the weights of low-value battlefields should be at least $u(M - M')/N$. Otherwise, we give a strategy for player B that prevents player A from obtaining a (u, p)-maxmin strategy which is assumed to exist. Note that by Line 13 of Algorithm 5, $M' = N(|A|(1-p/2)+1)$, which means $2(|A|-\lfloor M'/N \rfloor)/|A| < p$. Also note that by Lemma 5.2, player B can put M' troops in the high-value battlefields to make sure that player A wins no high-value battlefield with probability at least $1 - p$. Therefore, there exists a strategy of player A that achieves a payoff of at least u from the low-value battlefields with non-zero probability. However, in the case that

$$w_{b_1} + w_{b_2} + \ldots + w_{b_{|B|}} \geq u(M - M')/N$$

does not hold, the describe a strategy of player B that prevents player A from having a (u, p)-maxmin strategy. Let $W = \sum w_{b_i}$ be the total sum of low-value battlefields' weights. Put $(M - M')w_{b_i}/W$ troops in every battlefield b_i. Note that this requires no more than $M - M'$ troops since

$$\begin{aligned}\sum (M-M')w_{b_i}/W &= (M-M')(\sum w_{b_i})/W \\ &= (M-M')W/W \\ &= M-M'.\end{aligned}$$

In addition to this, note that in order for player A to win a battlefield b_i, she has to put more than $(M-M')w_{b_i}/W$ troops in that battlefield. Therefore, the *ratio of the payoff over the number of necessary troop* to win for each battlefield is at least $W/(M-M')$ and thus player A can obtain no more than $WN/(M-M')$ payoff. This implies that $W \geq u(M-M')/N$ since a (u,p)-maxmin is guaranteed to exist for player A.

This lower bound on the total weights of low-value battlefields implies that $|\Delta| \geq \frac{8}{5}(M-M')/N$. To see this, note that by Line 7, for any bundle of battlefields $\delta \in \Delta$, the total weight of the battlefields in δ should be less than $\frac{5}{8}u$ $(= \frac{1}{2}u + \frac{1}{8}u)$. Also, $|\Delta|$ is bounded as follows:

$$|\Delta| \geq \frac{u(M-M')/N}{u5/8} \geq \frac{8}{5}(M-M')/N.$$

By Algorithm 5, player A plays in the low-value battlefields with probability at least 3/4. Choosing a bundle $\delta \in \Delta$ uniformly at random, she distributes her money over all the battlefields of the bundle proportional to their weights. In this case, if player B puts at most $\frac{3}{4}N$ troops in this bundle player A wins at least $\frac{1}{4}$ fraction of the overall weight of the battlefields in bundle δ which is at least $u/8$. By the fact that $|\Delta| \geq \frac{8}{5}(M-M')/N$, and that player B is able to put in at most $\frac{(M-M')}{3N/4}$ bundles at least $\frac{3}{4}N$ troops, there are at least

$$\frac{8(M-M')}{5N} - \frac{4(M-M')}{3N} \leq \frac{4(M-M')}{15N}$$

bundles that player B puts at most $\frac{3}{4}N$ troops in, which is a $\frac{1}{6}$ fraction of all the bundles. As a result, in this case, with probability at least $\frac{1}{8} = \frac{1}{6} \cdot \frac{3}{4}$, player A gains $\frac{u}{8}$ payoff since by Line 15 of the algorithm she plays this strategy on the low-value battlefields with probability at least $\frac{3}{4}$. □

As a remark, our algorithms could be modified to have no dependence on p which results in constructive variants of Theorem 5.3 and Theorem 6.2. More precisely, one can obtain an algorithm that only takes u in addition to an instance $\mathcal{C}$ of discrete (resp. continuous) Colonel Blotto in the input, and outputs a $(u/(16(\lceil \log K \rceil + 1)), p/4)$-maxmin (resp. $(u/8, p/8)$-maxmin) strategy where p is the maximum possible probability for which a (u,p)-maxmin strategy exists for instance $\mathcal{C}$.

7 Auditing Game

In what follows we propose an approximation solution for the auditor. We first begin by showing an upper-bound on the highest protection that the auditor can provide, and then proceed by proposing a strategy to obtain a fraction of that.

Note that if the hacker chooses a set of states with total value less than $\sum v_i - u$, the auditor is guaranteed to receive a payoff of at least u. Hence we assume throughout that any valid strategy of the hacker chooses a set of states that have a total value of more than $\sum v_i - u$.

We show an upper bound on the best that the auditor can do via a linear program. In this linear program, for every pure strategy x of the hacker, there is a variable $f_x \geq 0$ which is a real value corresponding to this strategy. Intuitively, the value of f_x, after being normalized, determines the probability that the hacker plays strategy x. We refer to f_x as the flow of strategy x. Moreover, for every state v_i, we have a constraint to ensure that the total flow of the strategies of the hacker that change the result of state v_i is bounded by 1. The objective of the program is to maximize the total flow of all strategies.

$$\text{(7.5)} \qquad \begin{aligned} &\text{maximize:} && \sum_{x \in X} f_x \\ &\text{subject to:} && \sum_{s_i \ni x} f_x \leq 1 \qquad \forall i \in [n] \end{aligned}$$

Let OPT be the optimal solution of LP 7.5. We show that no strategy of the auditor can achieve a utility of u with probability more than m/OPT. Of course, if $\text{OPT} \leq m$, this bound is meaningless. To this end, let f^* be an optimal solution of LP 7.5 and consider a mixed strategy of the hacker that plays every strategy x with probability f^*_x/OPT (notice that the probabilities sum up to 1 since $\text{OPT} = \sum f^*_x$).

Recall that the total flow of the strategies that hack every state is bounded by 1, and the hacker plays every strategy with probability f^*_x/OPT, thus the probability that any strategy of the hacker changes the outcome of any state is bounded by $1/\text{OPT}$. Moreover, every strategy of the auditor is to inspect at most m states, thus she catches the hacker with a probability of at most m/OPT. Hence, no strategy of the auditor can protect the election with a probability more than m/OPT.

LEMMA 7.1. *There is no (u,p')-maxmin strategy for the auditor for $p' > m/\text{OPT}$, where OPT is the optimal solution of LP 7.5.*

Now, in order to find an approximation solution for the auditor, we take the dual of LP 7.5 to obtain the following linear program.

$$\text{(7.6)} \qquad \text{minimize:} \qquad \sum_{i \in [n]} g_i$$
$$\text{subject to:} \ \sum_{s_i \in x} g_i \geq 1 \qquad \forall x \in X$$

By the strong duality theorem, the optimal solutions of LP 7.6 and LP 7.5 are equal. Consider an optimal solution g^* of LP 7.6. Based on this solution, we construct a strategy for the auditor that protects the election with probability at least $(1 - 1/e)m/\text{OPT}$. Notice that in LP 7.6, for every state s_i, we have a variable g_i and the total sum of the variables is equal to OPT. Moreover, for every strategy x of the hacker, we have a constraint to ensure that the sum of the g_i^*'s corresponding to x is at least 1. Consider a probability distribution D over the states, such that for every state s_i, $D_{s_i} = g_i^*/\text{OPT}$. Trivially, $\sum D_{s_i} = 1$ holds. Now, we define a strategy for the auditor and show its approximation factor is bounded by $1-1/e$. In this strategy, we draw m states from the probability distribution D and investigate the results of those states. Notice that some states might appear several times in our solution, but we can ignore repetitions and consider each just once.

Now, let us analyze this strategy. Fix a pure strategy x for the hacker. We know that the sum of $\{g_i^*\}$ for the states corresponding to strategy x is at least 1. Therefore, every time we draw a state according to the probability distribution D, one of the states of x is audited with probability at least $1/\text{OPT}$. We draw m different states, therefore, the probability that one of the states of x appears in the strategy of the auditor is at least

$$1 - (1 - 1/\text{OPT})^m \geq (1 - 1/e)m/\text{OPT}.$$

If $m \geq \text{OPT}$, then

$$\begin{aligned} 1 - (1 - 1/\text{OPT})^m &\geq 1 - (1 - 1/\text{OPT})^{\text{OPT}} \\ &\geq (1 - 1/e) \end{aligned}$$

and hence the proof is trivial. Otherwise,

$$1 - (1 - 1/\text{OPT})^m \geq (1 - 1/e)m/\text{OPT}$$

Thus, if the auditor follows this strategy, she can protect the election with probability at least $(1 - 1/e)m/\text{OPT}$. Notice that solving the linear programs can be done in polynomial time via the ellipsoid method, and thus we can find this strategy in polynomial time. This result married with the upper-bound mentioned earlier gives us the following theorem.

THEOREM 7.1. *Given a minimum utility u and an instance of the auditing game, there exists a polynomial time algorithm to find a $(u, (1 - 1/e)p)$-maxmin strategy for the auditor; where for any $p' > p$, no (u, p')-maxmin strategy exists for the auditor.*

7.1 Discussion and Practical Issues Our model for the auditing game simplifies some real-world constraints. We discuss these constraints and their implications in this section.

First, our full-information assumption about the outcome of the election is not completely the case in reality. One workaround to get this information is by polls. However, the accuracy of polls, whether prospective or exit, is sometimes worse than the margin of victory.

Second, we assume only one limit for the auditor and that is the number of states that she can audit. The current situation in real life is more complicated. First, many states do not have technology that is auditable or laws that would allow audits. Second, the cost of auditing different states is not necessarily the same, e.g., it might be proportional to the population of that state (if the audit comprises a full hand count), or approximately inversely proportional to the margin or the square of the margin (if the audit is statistical).

Third, we assume if the auditor audits a hacked state, she catches the hacker. In reality, it would be more realistic to assume that the auditor will use a method that relies on sampling ballots rather than on a full recount, and that has some known minimum probability of detecting that the results are wrong.

7.2 A Reduction From Generalized Blotto to Auditing Game In a generalized version of Colonel Blotto which we call Threshold Blotto game, we have two players A and B, each with a given number of troops. Both of the players distribute their troops over k battlefields and the payoff of each battlefield i is determined based on a specific function h_i with respected to the troops of each side in battlefield i. If the total payoff of player A is more than a threshold τ then player A wins the game, otherwise player B is declared the winner.

More precisely, let x and y specify the number of troops of the players. A set of k battlefields with payoff functions $\{h_1, h_2, \ldots, h_k\}$, where each function h_i admits two inputs α and β corresponding to the number of troops of the players in that battlefield and determines the payoff to player A based on these values. A threshold τ denoting the minimum payoff for A to win the game. Colonel A wins if $\sum h_i(\alpha_i, \beta_i) \geq u$ where α and β are vectors of size k for players strategies. If A wins, she receives a payoff of 1, otherwise a payoff of -1. Colonel B's payoff is the negation of colonel A's payoff and thus the game is zero-sum.

In what follows, we show a reduction from Threshold Blotto to the auditing game. Suppose we have an Auditing game with n states $s_1, s_2, \ldots, s_n$ and the hacker needs to flip the results of some states with a total number of electoral votes of at least t and the auditor audits the results of at most m states. Based on this, we construct an instance of the Threshold Blotto game with $k = n+1$ battlefield. We associate colonel A to the hacker and colonel B to the auditor. We set the number of troops of colonel A equal to the total number of electoral votes of all states ($\sum v_i$) and the number of troops of Colonel B equal to m.

To make the reduction cleaner, we set the payoff function of each battlefield i (h_i) in such a way that the optimal strategies meet the following conditions:

- For every battlefield $1 \le i \le n$, player A either puts 0 or v_i troops (the number of electoral votes of state s_i) in the i'th battlefield.

- For every battlefield $1 \le i \le n$, player B either allocated 0 or 1 troop to that battlefield.

- The total number of troops of player A in the first n battlefields is at least t.

If all these conditions are met then, every time colonel A puts v_i troops in some battlefield $1 \le i \le n$, we can think of that as if the hacker hacks state s_i's results. Similarly, whenever colonel B puts a troop in a battlefield $1 \le i \le n$, we can interpret that as the auditor issuing a recount for state s_i. Therefore, we can define the payoff functions accordingly: For $1 \le i \le n$ we set

$$h_i(\alpha_i, \beta_i) = \begin{cases} -\infty, & \text{if } \beta_i \notin \{0,1\} \\ \infty, & \text{if } \alpha_i \notin \{0, v_i\} \\ 0 & \text{if } \alpha_i = 0 \text{ or } \beta_i = 0 \\ 1 & \text{if } \alpha_i > 0 \text{ and } \beta_i > 0 \end{cases}$$

where α_i and β_i denote the number of troops of colonels A and B in battlefield i respectively. This payoff function specifies the payoff of colonel B. The payoff of colonel A is the nagation of that. This guarantees that in an NE we always have $\alpha_i = \{0, v_i\}$ and $\beta_i = \{0,1\}$ for all $1 \le i \le n$. Notice that colonel B gets a payoff of 1 if she puts a troop in a battlefield and colonel A also puts v_i troops in that battlefield. To make sure the total number of electoral votes of the hacker is at least t, we set the following payoff function for the last battlefield:

$$h_k(\alpha_i, \beta_i) = \begin{cases} -\infty, & \text{if } \beta_i \neq 0 \\ \infty, & \text{if } \alpha_i > \sum v_i - t \\ 0 & \text{otherwise} \end{cases}$$

Notice that if $\alpha_i \le \sum v_i - t$, then the total number of troops of colonel A in the first n battlefield is at least t and thus the corresponding strategy of the hacker is valid. Now, colonel B gets a non-zero payoff in this game if and only if she puts some troops in a battlefield which also contains some troops of colonel A as well. Therefore, if we set $\tau = 1$ an NE of this game corresponds to an NE of the auditing game.

THEOREM 7.2. *Threshold Blotto game is computationally harder than the auditing game.*

8 Conclusion and Open Problems

In this paper, we went beyond strategies that maximize the expected payoff and introduced the general notion of (u,p)-maxmin strategies where receiving a payoff of u with probability at least p is guaranteed. We then gave approximation algorithms for the problem of finding (u,p)-maxmin strategies for two games, Colonel Blotto, and auditing games.

8.1 Open Problems

Four main theoretical problems remain open.

(i). We proposed several approximation algorithms. Is it possible to improve the approximation factors or give exact solutions for any of them? In particular, Theorem 6.1 gives an exact $(u,1)$-maxmin solution for continuous Colonel Blotto, however for the discrete case, we only approximated $(u,1)$-maxmin by a factor of 2 (Theorem 5.1). How tight is this approximation factor? Is it possible to give a constant approximation for discrete Colonel Blotto using a better approximation of $(u,1)$-maxmin strategies?

(ii). Our approximation algorithms for the Colonel Blotto problem are bi-criteria, i.e., they approximate both u and p at the same time. Is there any algorithm that approximates only the probability? That is, is there any algorithm that for any given u, outputs a (u,p)-maxmin strategy where p approximates the highest possible probability of guaranteeing a payoff of u?

(iii). We gave evidence in Section 4.2 that it seems to be computationally hard to give exact (u,p)-maxmin strategies for Colonel Blotto instances based on the hardness of best response. There are, however, rare cases where computing the best response is actually harder than solving the actual game. Filling this gap, by giving a direct hardness result is left open.

(iv). We consider a (u', p')-maxmin strategy to be a good approximation of (u, p)-maxmin if u'/u and p'/p are large (e.g., constant) fractions. One might be interested in bounding the failure probability instead. Consider for example a $(u, 0.96)$-maxmin strategy. A constant bound of 1/8 (as in Theorem 7.1) on p'/p achieves a $(u', 0.12)$-maxmin strategy. The failure probability in the original strategy is 0.04 $(= 1 - 0.96)$, whereas, the guaranteed failure probability in the given approximation is not less than 0.88 $(= 1 - 0.12)$. Our approximation algorithms do not guarantee any bound on the failure ratio (0.04/0.88). Giving algorithms that bound this ratio is left open.

References

[1] A. Ahmadinejad, S. Dehghani, M. Hajiaghay, B. Lucier, H. Mahini, and S. Seddighin. From Duels to Battlefields: Computing Equilibria of Blotto and Other Games. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[2] M. Babaioff, N. Immorlica, B. Lucier, and S. M. Weinberg. A Simple and Approximately Optimal Mechanism for an Additive Buyer. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 21–30. IEEE, 2014.

[3] M. Bateni, S. Dehghani, M. Hajiaghayi, and S. Seddighin. Revenue maximization for selling multiple correlated items. In *Algorithms-ESA 2015*, pages 95–105. Springer, 2015.

[4] S. Behnezhad, S. Dehghani, M. Derakhshan, M. Haji-Aghayi, and S. Seddighin. Faster and Simpler Algorithm for Optimal Strategies of Blotto Game. *arXiv preprint arXiv:1612.04029*, 2016.

[5] S. Behnezhad, M. Derakhshan, M. T. Hajiaghayi, and A. Slivkins. A Polynomial Time Algorithm for Spatio-Temporal Security Games. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017*, pages 697–714, 2017.

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

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

[8] B. Bošanskỳ, V. Lisỳ, M. Jakob, and M. Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 989–996. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[9] J. Bretschneider, S. Flaherty, S. Goodman, M. Halvorson, R. Johnston, M. Lindeman, R. L. Rivest, P. Smith, and P. B. Stark. Risk-limiting Post-election Audits: Why and How, 2012.

[10] B. Chilingirian, Z. Perumal, R. L. Rivest, G. Bowland, A. Conway, P. B. Stark, M. L. Blom, C. Culnane, and V. Teague. Auditing Australian Senate Ballots. *CoRR*, abs/1610.00127, 2016.

[11] F. Fang, A. X. Jiang, and M. Tambe. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 957–964. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[12] U. Feige, K. Jain, M. Mahdian, and V. Mirrokni. Robust Combinatorial Optimization with Exponential Scenarios. *Integer programming and combinatorial optimization*, pages 439–453, 2007.

[13] M. Fréchet. Commentary on the three notes of Emile Borel. *Econometrica*, 21:118–124, 1953.

[14] M. Fréchet. Emile Borel, initiator of the theory of psychological games and its application. *Econometrica*, 21:95–96, 1953.

[15] J. Garg, A. X. Jiang, and R. Mehta. Bilinear games: Polynomial time algorithms for rank based subclasses. In *International Workshop on Internet and Network Economics*, pages 399–407. Springer, 2011.

[16] O. A. Gross and R. Wagner. A continuous Colonel Blotto game. 1950.

[17] S. Hart. Discrete Colonel Blotto and general lotto games. *International Journal of Game Theory*, 36(3):441–460, 2008.

[18] R. Hortala-Vallve and A. Llorente-Saguer. Pure strategy Nash equilibria in non-zero sum colonel Blotto games. *International Journal of Game Theory*, 41(2):331–343, 2012.

[19] N. Immorlica, A. T. Kalai, B. Lucier, A. Moitra, A. Postlewaite, and M. Tennenholtz. Dueling algorithms. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 215–224. ACM, 2011.

[20] J. Letchford and V. Conitzer. Solving Security Games on Graphs via Marginal Probabilities. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[21] X. Li and A. C.-C. Yao. On Revenue Maximization for Selling Multiple Independently Distributed Items. *Proceedings of the National Academy of Sciences*, 110(28):11232–11237, 2013.

[22] M. Lindeman, P. B. Stark, and V. S. Yates. BRAVO: Ballot-polling Risk-limiting Audits to Verify Outcomes. In *EVT/WOTE*, 2012.

[23] S. Park and R. L. Rivest. Towards secure quadratic voting. *Public Choice*, pages 1–25, 2016.

[24] R. L. Rivest. DiffSum-A Simple Post-Election Risk-

Limiting Audit. *arXiv preprint arXiv:1509.00127*, 2015.

[25] B. Roberson. The colonel blotto game. *Economic Theory*, 29(1):1–24, 2006.

[26] B. Roberson and D. Kvasov. The non-onstant-sum Colonel Blotto game. *Economic Theory*, 51(2):397–433, 2012.

[27] A. Rubinstein and S. M. Weinberg. Simple Mechanisms for a Subadditive Buyer and Applications to Revenue Monotonicity. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 377–394. ACM, 2015.

[28] M. Shubik and R. J. Weber. Systems defense games: Colonel Blotto, command and control. *Naval Research Logistics Quarterly*, 28(2):281–287, 1981.

[29] P. B. Stark. A Sharper Discrepancy Measure for Post-election Audits. *The Annals of Applied Statistics*, pages 982–985, 2008.

[30] P. B. Stark. Conservative statistical post-election audits. *The Annals of Applied Statistics*, pages 550–581, 2008.

[31] P. B. Stark. Auditing a Collection of Races Simultaneously. *arXiv preprint arXiv:0905.1422*, 2009.

[32] P. B. Stark. Efficient Post-election Audits of Multiple Contests: 2009 California Tests. 2009.

[33] P. B. Stark. Risk-limiting Post-election Audits: P-values From Common Probability Inequalities. Department of Statistics. *University of California, Berkeley. http://www. stat. berkeley. edu/˜ stark/Preprints/pvalues09. pdf*, 2009.

[34] S. Wang and N. Shroff. Security Game with Non-additive Utilities and Multiple Attacker Resources. *arXiv preprint arXiv:1701.08644*, 2017.

[35] H. Xu. The Mysteries of Security Games: Equilibrium Computation Becomes Combinatorial Algorithm Design. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 497–514. ACM, 2016.

[36] H. Xu, F. Fang, A. X. Jiang, V. Conitzer, S. Dughmi, and M. Tambe. Solving Zero-Sum Security Games in Discretized Spatio-Temporal Domains. In *AAAI*, pages 1500–1506. Citeseer, 2014.