

Efficient Method for Finding Optimal Strategies in Chopstick Auctions with Uniform Objects Values

Stanisław Kaźmierowski
University of Warsaw
Warsaw, Poland
s.kaźmierowski@uw.edu.pl

Marcin Dziubiński
University of Warsaw
Warsaw, Poland
m.dziubinski@mimuw.edu.pl

ABSTRACT

We propose an algorithm for computing Nash equilibria (NE) in a class of conflicts with multiple battlefields with uniform battlefield values and a non-linear aggregation function. By expanding the symmetrization idea of Hart [9], proposed for the Colonel Blotto game, to the wider class of symmetric conflicts with multiple battlefields, we reduce the number of strategies of the players by an exponential factor. We propose a *clash matrix algorithm* which allows for computing the payoffs in the *symmetrized* model in polynomial time. Combining symmetrization and clash matrix algorithm with the double oracle algorithm we obtain an algorithm for computing NE in the models in question that achieves a significant speed-up as compared to the standard, LP-based, approach. We also introduce a heuristic to further speed up the process. Overall, our approach offers an efficient and novel method for computing NE in a specific class of conflicts, with potential practical applications in various fields.

KEYWORDS

chopstick auction; Nash Equilibrium; conflicts with multiple battlefields; zero-sum game; optimal strategies

ACM Reference Format:

Stanisław Kaźmierowski and Marcin Dziubiński. 2024. Efficient Method for Finding Optimal Strategies in Chopstick Auctions with Uniform Objects Values. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Real-life scenarios of rivalry between two or more players often involve competition in more than one area at a time. Examples include firms competing simultaneously for multiple markets, military commanders deploying troops to the front lines, political parties distributing campaign funds among different regions of the country, or airport security having to assign a number of police dogs among security checkpoints. Game theoretic models used to formally capture such competition form a class called *conflicts with multiple battlefields* [13]. A famous example of such a model is the Colonel Blotto game [4]. In these models, two players distribute their limited resources across a number of battlefields. Assignment

of resources determines the outcome at each battlefield and the outcome of the game is an aggregate of the outcomes at the individual battlefields. The way the outcomes at individual battlefields are aggregated depends on a particular application. A well-known aggregation function is taking the sum of outcomes across all the battlefields. This leads to the Colonel Blotto game.

In this paper we address the problem of computing Nash equilibria under two natural aggregation functions called *more than opponent* and *majoritarian*. The more than opponent aggregation takes value 1, if the number of battlefields won is strictly greater than the number of battlefields lost, 0, if it is equal, and -1 , if it is less. It is suitable for military conflicts where the fighting parties care not about how many battles they win but about winning more battles than the opponent in order to win the war. It is also applicable to political competitions where candidates are rather win-motivated than vote-motivated, caring predominantly about winning the competition and not about the margin at which they win. The majoritarian aggregation takes a value of 1, if the number of battlefields won exceeds half of the total number, 0 if it is equal, and -1 if it is less. It is applicable to single-member districts voting rivalry between two political parties. It is also considered in a model of simultaneous standard auctions with externalities, called chopstick auctions [6, 19, 20], where only winning more than half of the objects bears any value to each bidder. Unlike in the Colonel Blotto game, the aggregation functions in question do not result in bilinear payoffs and, therefore, we can not apply the polynomial time algorithms designed for conflicts with multiple battlefields and bilinear payoffs [1, 3].

In this paper, we give a homogeneous formulation of conflicts with multiple battlefields. The main obstacles in computing the NE in these models are the exponential (wrt the model parameters) size of the strategy sets of the players and non-bilinear payoffs. Generalizing the idea of Hart [9], we use a reduction of the strategy sets of the models in question by an exponential factor, called *symmetrization*. The reduction of the strategy sets makes the computation of payoffs in the *symmetrized* model challenging. Our main contribution is a *clash matrix algorithm* which allows for computing the payoffs in the *symmetrized* models in polynomial time. We combine the polynomial time algorithm for computing payoffs with the *double oracle algorithm* [15], used to compute equilibria in zero-sum games with large strategy spaces. We propose a heuristic, based on the natural monotonicity properties of the aggregation functions, that allows us to further speed up this algorithm. We show, via computational experiments, that thus obtained algorithm achieves a significant speed-up as compared to any approach that requires calculating the entire payoff matrix.



This work is licensed under a Creative Commons Attribution International 4.0 License.

1.1 Related literature

Conflicts with multiple battlefields, considered since the beginning of modern game theory, were first introduced by Borel [4], where the Colonel Blotto game is defined. Since then, many game theoretic models that fall into this category were studied, including the hide-and-seek game [21], chopstick auctions [20], audit games [2], as well as some types of security games [12, 16]. See [13] for an excellent survey of this type of models.

The literature on NE computation is vast, and we restrict attention to the closest related works. The models we consider are two-player zero-sum games. The computation of NE in such games can be reduced to solving an LP problem [18] and requires polynomial time with respect to the number of strategies of both players. An exponential number of pure strategies with respect to the model parameters makes those methods inapplicable in the case of conflicts with multiple battlefields. In the case of bilinear payoffs, like in Colonel Blotto game, NE can be computed in polynomial time [1, 3]. This is also possible for some types of security games [12].

The *double oracle algorithm (DOA)* was first proposed as a method for solving zero-sum games in [15]. It was later used for solving zero-sum security games on graphs in [10], which showed that it is worth considering as an alternative for LP-solver-based algorithms. Games considered in this paper fall into a broader category of integer programming games (IPGs) [11]. [5] proposed a *sampled generation method (SGM)* which is a significant generalization of DOA to multiple players and arbitrary payoffs. They show that SGM allows for finding NE in a finite number of steps. SGM (and, in particular, DOA) proceeds by iteratively expanding the sets of strategies of the players and solving thus obtained sampled games. Since conflicts with multiple battlefields we consider are zero-sum games, we solve the sample games using the LP solvers facilitated with the clash matrix algorithm. In the case of the general IPGs the PNS algorithm can be used [5, 17].

The rest of the paper is organized as follows. In Section 2 we define the model of conflicts with multiple battlefields. Section 3 provides the complexity result of finding the best response in chopstick auctions and majoritarian auctions. In Section 4 we define the symmetrized version of the model. Computation of payoffs from symmetrized strategy profiles and the clash matrix algorithm are given in Section 5. Section 6 presents the double oracle algorithm and the heuristic for improving its performance in the case of models in question. We provide a computational evaluation of our approach in Section 7 and conclude in Section 8.

2 MODEL

Conflicts with multiple battlefields model a competition between two players, whom we will denote A and B. Each player has a number of discrete resources, e.g. military units or coins, described by D_A and D_B , respectively. Players compete on the set $[n] = \{1, 2, \dots, n\}$ of n battlefields. We assume that $n \geq 2$. Each battlefield $i \in [n]$ has value 1. The player's strategy is to distribute her resources across battlefields. The set of strategies of player $C \in \{A, B\}$ is:

$$S_C = \left\{ \mathbf{s} \in \mathbb{N}^n : \sum_{i=1}^n s_i = D_C \right\}. \quad (1)$$

A pair of strategies $(\mathbf{s}^A, \mathbf{s}^B)$ is called a *strategy profile*.

Payoff from a single battlefield $i \in [n]$ is defined as

$$u_i^A(\mathbf{s}^A, \mathbf{s}^B) = -u_i^B(\mathbf{s}^A, \mathbf{s}^B) = \text{sign}(s_i^A - s_i^B), \quad (2)$$

meaning that a battlefield is won by the player who assigned more resources to it.

Applying (2) to a strategy profile we get a battlefield outcomes vector

$$\mathbf{u}^A(\mathbf{s}^A, \mathbf{s}^B) = -\mathbf{u}^B(\mathbf{s}^A, \mathbf{s}^B) = (u_1^A(\mathbf{s}^A, \mathbf{s}^B), \dots, u_n^A(\mathbf{s}^A, \mathbf{s}^B)). \quad (3)$$

Values of vector \mathbf{u} , i.e. the result of (3), are in $\{-1, 0, 1\}^n$.

Definition 2.1 (Aggregation function). Function $f: \{-1, 0, 1\}^n \rightarrow R$ is an aggregation function if f returns the same value for all permutations of every vector $\mathbf{u} \in \{-1, 0, 1\}^n$.

OBSERVATION 1 (AGGREGATION FUNCTIONS). *Each function f satisfying Definition 2.1 can be defined as $f(\mathbf{u}) = f_n(k_W, k_L)$, where k_W and k_L describe the number of 1s and -1s in vector \mathbf{u} .*

Given the aggregation function f we define the payoffs of each of the players by:

$$\pi_f^A(\mathbf{s}^A, \mathbf{s}^B) = -\pi_f^B(\mathbf{s}^A, \mathbf{s}^B) = f(\mathbf{u}^A(\mathbf{s}^A, \mathbf{s}^B)). \quad (4)$$

As the payoffs of two players always sum up to 0, the defined class of models describes zero-sum games. Some of the most commonly used aggregation functions are:

$$f_{\text{blotto}}(\mathbf{u}) = \sum_{i=1}^n u_i, \quad (5)$$

$$f_{\text{mto}}(\mathbf{u}) = \text{sign} \left(\sum_{i=1}^n u_i \right), \quad (6)$$

$$f_{\text{cho}}(\mathbf{u}) = \left[\sum_{i=1}^n [u_i = 1] > n/2 \right] - \left[\sum_{i=1}^n [u_i = -1] > n/2 \right], \quad (7)$$

where, given a condition φ , $[\varphi]$ is the Iverson bracket taking value 1 if φ is satisfied and value 0 otherwise.

Aggregation function (5) is the linear function used in Colonel Blotto game [4], (6) is the “more than opponent” function, and (7) is the “majoritarian” function. In general, many more aggregation functions generate nontrivial games.

Definition 2.2 (Conflict with multiple battlefields). Quadruple (D_A, D_B, n, f) defines a zero-sum two-player game with strategies sets defined by (1) and payoffs defined by (4).

We allow the players to make randomized choices. A *mixed strategy* of player $C \in \{A, B\}$ is a probability distribution on S_C . Given a non-empty set X , let $\Delta(X)$ denote the set of all probability distributions on X . The expected payoff to player $C \in \{A, B\}$ from a pair of mixed strategies $(\xi^A, \xi^B) \in \Delta(S_A) \times \Delta(S_B)$ is equal to

$$\Pi_f^C(\xi^A, \xi^B) = \sum_{(x,z) \in S_A \times S_B} \xi_x^A \xi_z^B \pi_f^C(x, z).$$

We assume that the players make their decisions “simultaneously”, i.e. each player chooses her strategy without observing the choice of the opponent. We are interested in (mixed strategy) Nash equilibria (NE) of the game, i.e. mixed strategy profiles $(\xi^A, \xi^B) \in \Delta(S_A) \times$

$\Delta(S_B)$ such that no player can improve her expected payoff by changing her strategy unilaterally, i.e. for each $C \in \{A, B\}$ and all $\zeta \in \Delta(S_C)$,

$$\Pi_f^C(\xi^A, \xi^B) \geq \Pi_f^C(\zeta, \xi^{-C}),$$

where $-C$ denotes the player other than C and (ζ, ξ^{-C}) is the strategy profile obtained from (ξ^A, ξ^B) by replacing ξ^C with ζ .

3 COMPLEXITY OF FINDING THE BEST PURE RESPONSE

Although the complexity of computing Nash equilibria of the games in question (defined by aggregation functions (6) and (7)) remains unknown, we show that finding a pure strategy best response to a given mixed strategy is a computationally hard problem. It is in stark contrast to the Colonel Blotto game, where finding a pure strategy best response is solvable in polynomial time. We prove the result using a reduction from the max coverage problem. This follows the idea in [2], where computing a pure strategy response to a mixed strategy, that maximizes the probability of obtaining payoff above a given threshold in the Colonel Blotto game is shown to be computationally hard.

The hardness result of finding the best response does not necessarily imply any hardness result for the considered game, however, it can be considered an insightful indicator of how hard the game is to solve. See [22] for results about a class of games where the hardness of finding the best response implies the hardness of finding a Nash equilibrium of the game.

Assume a mixed strategy ξ^B of player B, given as a list of pure strategies in its support, and their associated probabilities. The best response problem for a given aggregation function f , denoted by BR_f , is to find a pure strategy s^A of player A that maximizes the $\Pi_f^A(s^A, \xi^B)$.

PROPOSITION 3.1. *For aggregation functions f given by (6) and (7), there is no polynomial time algorithm to solve BR_f unless $P = NP$.*

PROOF. To prove this hardness we provide a reduction from the max-coverage problem to BR_f . A number k and a collection of sets $S = \{S_1, S_2, \dots, S_m\}$ are given. The maximum coverage problem is to find a collection $S' \subseteq S$, such that $|S'| \leq k$ and the number of covered elements (i.e. $|\bigcup_{S_i \in S'} S_i|$) is maximized.

Let $E = \bigcup_{S_i \in S} S_i$ denote the set of all elements in the given max-coverage instance. Assume, that the number of sets in S in which a given element appears is the same for every element in E and denote this number by t . If that is not the case, a polynomial number (with respect to $|E| \cdot |S|$) of additional singleton sets can be added to collection S , without interfering with the max-coverage complexity, or the resulting collection S' as no singleton is more desired than any other sets containing the only element of the considered singleton.

Consider a game with aggregation function f defined by (7), $2 \cdot |S|$ battlefields, where the player A has $|S| + k$ resources and player B has $(|S| + k + 1) \cdot (|S| - t) \cdot |E|$ resources. The first $|S|$ of $2 \cdot |S|$ battlefields correspond to the sets in S . For every element $e \in E$, we denote a corresponding pure strategy of player B, denoted by $s^{B,e}$, by assigning $|S| + k + 1$ resources to each battlefield that its corresponding set does not contain e and assigning 0 resources in

all other battlefields. Assume that player B uses a mixed strategy defined by choosing uniformly from the set of strategies $\bigcup_{e \in E} s^{B,e}$.

Assume that the best response of player A that maximizes her expected payoff is given. Note, that as player B always assigns either 0 or $|S| + k + 1$ resources to a single battlefield, it is sufficient for the player A to assign either 0 or 1 resources on every battlefield.

We claim that the best response of player A assigns $|S|$ resources to the remaining $|S|$ battlefields (one to each), which are always assigned 0 by player B. This guarantees that player A does not lose with probability 1, as it always wins at least half of the battlefields. The remaining k resources are assigned between the first $|S|$ battlefields, yielding a max-coverage of size k from sets in $|S|$, where each battlefield corresponding to the set in the considered max-coverage will be assigned exactly one resource.

The proof can be easily adapted for the aggregation function (6). Instead of assigning additional $|S|$ battlefields that are assigned more than 0 by player B, we only add $|S| - t$ additional battlefields and we reduce the number of resources of player A from $|S| + k$ to $|S| - t + k$. \square

4 SYMMETRIZED MODEL

A player with D_C resources to distribute over n battlefields has

$$\binom{n + D_C - 1}{n - 1} \leq (n + D_C - 1)^{\min(n-1, D_C)} \quad (8)$$

pure strategies. Thus, when the number of battlefields or the number of resources of the players are fixed, both players have polynomial size strategy sets and, since the game is zero-sum, a NE can be found in polynomial time [18]. It is important to note that the degree of the polynomial that describes the time complexity is limited by the parameter value (the number of battlefields or the number of resources of the two players) which can be arbitrarily large. Moreover, when we consider a model with n battlefields and a fixed parameter $d \in N$, such that the number of resources of both players is $(n + d)$, the number of strategies of the players grows exponentially (c.f. (8)). For example, with 20 battlefields and 25 resources, the number of ways in which a player can distribute the resources across the battlefields is more than 10^{12} . With such a number of pure strategies, LP solvers, which are standard tools used for solving zero-sum games, become impossible to use in practice.

To address the problem of large strategy sets, we follow the idea of Hart [9] and consider a symmetrized variant of the model, defined as follows. Let P_n denote the set of all permutations over $[n]$. Given a strategy $s = (s_1, \dots, s_n) \in S_C$ of player C let $\sigma(s)$ be the mixed strategy which for each permutation $p \in P_n$ over $[n]$ chooses, with probability $1/n!$, an assignment $p(s) = (s_{p(1)}, \dots, s_{p(n)})$ of resources.

Definition 4.1 (Symmetrized conflict with multiple battlefields). A symmetrized conflict with multiple battlefields is the variant of conflict with multiple battlefields with the same set of players, each player $C \in \{A, B\}$ having a set of strategies $\sigma(S_C) = \{\sigma(s) : s \in S_C\}$ and payoff from strategy profiles $(\sigma(s^A), \sigma(s^B)) \in \sigma(S_A) \times \sigma(S_B)$ defined by $\Pi_f^C(\sigma(s^A), \sigma(s^B))$.

The elements of $\sigma(S_C)$ are called *symmetric strategies*. Since symmetric strategies are order-invariant, when referring to a symmetric

strategy $\sigma(s)$ we will assume that $s_1 \geq \dots \geq s_n$. Both models (defined in Definitions 2.2 and 4.1) describe finite games and therefore, by the Nash theorem, each of these games has a Nash equilibrium in mixed strategies.

Given a mixed strategy $\xi \in \Delta(S_C)$ of player C in the original game, let $\sigma(\xi)$ be the mixed strategy that for each permutation $p \in P_n$ and each $s \in S_C$ chooses, with probability $\xi_s/n!$, assignment $(s_{p(1)}, \dots, s_{p(n)})$. Notice that for any mixed strategy $\zeta \in \Delta(\sigma(S_C))$ in the symmetrized game there exists a mixed strategy ξ in the original game such that $\zeta = \sigma(\xi)$. We will call the mixed strategies of the symmetrized game *symmetric mixed strategies*.

The following proposition implies that any NE of any game that matches Definition 4.1 is also an NE of a corresponding game described by Definition 2.2.

PROPOSITION 4.2. *For any mixed strategy profile (ξ^A, ξ^B) of the symmetric conflict with multiple battlefields, if $(\sigma(\xi^A), \sigma(\xi^B))$ is an NE of the symmetrized variant of the game then it is also an NE of the original game.*

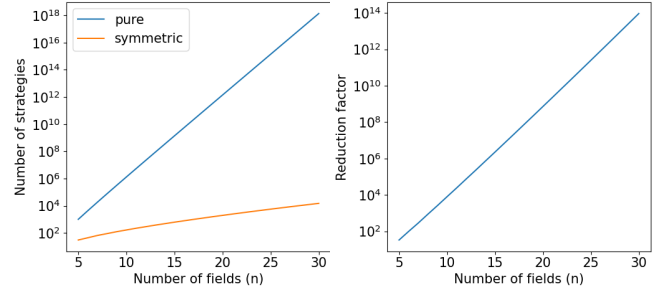
PROOF. First, observe that for any player $C \in \{A, B\}$, any aggregation function f , any two mixed strategies $\xi \in \Delta(S_C)$, of C and $\zeta \in \Delta(S_{-C})$ of the other player and any permutation $q \in P_n$,

$$\begin{aligned} \Pi_f^C(\sigma(\xi), q(\zeta)) &= \sum_{p \in P_n} \sum_{x \in \Delta(S_C)} \sum_{z \in \Delta(S_{-C})} \frac{\xi_x}{n!} \zeta_z \pi_f^C(p(x), q(z)) \\ &= \sum_{p \in P_n} \sum_{x \in \Delta(S_C)} \sum_{z \in \Delta(S_{-C})} \frac{\xi_x}{n!} \zeta_z \pi_f^C(p(q^{-1}(x)), z) \\ &= \sum_{p \in P_n} \sum_{x \in \Delta(S_C)} \sum_{z \in \Delta(S_{-C})} \frac{\xi_x}{n!} \zeta_z \pi_f^C(p(x), z) = \Pi_f^C(\sigma(\xi), \zeta). \end{aligned} \quad (9)$$

Take any mixed strategy profile (ξ^A, ξ^B) of the game in question and suppose that $(\sigma(\xi^A), \sigma(\xi^B))$ is a MNE of the symmetrized variant of the game. Assume, to the contrary, that it is not a MNE of the original game. It means that there exists a player $C \in \{A, B\}$ and a mixed strategy $\zeta \in \Delta(S_C)$ of C such that $\Pi_f^C(\zeta, \sigma(\xi^{-C})) > \Pi_f^C(\sigma(\xi^A), \sigma(\xi^B))$. By (9), $\Pi_f^C(\zeta, \sigma(\xi^{-C})) = \Pi_f^C(\sigma(\zeta), \sigma(\xi^{-C}))$. Hence $\Pi_f^C(\sigma(\zeta), \sigma(\xi^{-C})) > \Pi_f^C(\sigma(\xi^A), \sigma(\xi^B))$. Since $\sigma(\zeta)$ is a symmetric strategy, this contradicts the assumption that the profile $(\sigma(\xi^A), \sigma(\xi^B))$ is a MNE of the symmetrized variant of the game. Thus the claim of the proposition must hold. \square

The advantage of considering the symmetrized models is that the strategy sets of the players are reduced by the exponential factor, which is presented in the following figure.

Figure 1a shows the comparison of the number of the pure strategies and the number of the symmetric strategies, when the number of resources is a linear function of the number of battlefields (here the number of battlefields increased by 5). The vertical axis shows the number of strategies using a logarithmic scale. By Eq. (8), when only the number of battlefields or only the number of resources grows, the size of strategy sets (both pure and symmetric) grows



(a) The number of pure and symmetric strategies for $n + 5$ resources. (b) The reduction factor.

Figure 1: Comparison of number of pure and pure symmetric strategies

polynomially. Only when both the parameters grow together at the same time, the size of strategy sets grow exponentially.

Figure 1b shows the ratio of the number of pure strategies to the number of symmetric strategies (the reduction factor) for data in Figure 1b. Notice that although the number of symmetric strategies is still exponential, the reduction is by an exponential factor with respect to the parameters of the model.

To realize that the number of symmetrized strategies is still exponential with respect to the model parameters when the number of resources of a player is a linear function of the number of the battlefields, note that, as shown in [8], the number $p(n)$ of different partitions of n into the sum of non-negative integers has asymptotic growth of:

$$p(n) \sim \frac{1}{4n\sqrt{3}} \cdot \exp\left(\pi\sqrt{\frac{n}{2}}\right). \quad (10)$$

Consider a model with n battlefields and a fixed parameter $d \in N$, such that the number of resources of both players is $(n + d)$. The number of symmetrized strategies of each player is a number of partitions of $(n + d)$ into at most n parts. This is larger than the number of partitions of n into at most n parts, which is exactly $p(n)$. Therefore, the number of symmetrized strategies grows exponentially with respect to the number of battlefields in such a setting.

5 COMPUTING PAYOFFS FROM SYMMETRIC STRATEGIES

To benefit from restricting to the symmetrized variant of the model, we need to be able to efficiently compute single payoffs from symmetric strategy profiles $(\sigma(s^A), \sigma(s^B))$.

Notice that payoff to player $C \in \{A, B\}$ from a strategy profile $(\sigma(s^A), \sigma(s^B)) \in \sigma(S_A) \times \sigma(S_B)$ is given by:

$$\begin{aligned} \Pi_f^C(\sigma(s^A), \sigma(s^B)) &= \frac{1}{(n!)^2} \sum_{q \in P_n} \sum_{p \in P_n} \pi_f^C(q(s^A), p(s^B)) = \\ &= \frac{1}{n!} \sum_{p \in P_n} \pi_f^C(p(s^A), p(s^B)). \end{aligned} \quad (11)$$

To calculate the payoff from a pair of symmetric strategies $(\sigma(s^A), \sigma(s^B))$ in a naïve way, we need to calculate the payoff of the general model for each permutation. This makes calculating the payoff matrices of the symmetrized games almost as costly as calculating the payoff matrices of unsymmetrized games. To address this issue, we provide an algorithm that computes a single payoff from a pair of symmetric strategies in polynomial time with respect to n .

5.1 Clash matrix

By a clash matrix of a pair of symmetric strategies $(\sigma(s^A), \sigma(s^B))$ we mean a matrix of dimension $n \times n$ defined as follows:

$$M_{i,j}^{s^A, s^B} = \text{sign}(s_i^A - s_j^B).$$

That is, the matrix cell at coordinates (i, j) stores information (difference sign) of the result of clashing the i -th resource in player A's vector s^A with the j -th resource in player B's vector s^B .

Example 5.1 (Clash matrix).

$$M^{(3,1,0),(2,2,0)} = \begin{matrix} & & 2 & 2 & 0 \\ & 3 & \begin{pmatrix} 1 & 1 & 1 \\ -1 & -1 & 1 \\ 0 & -1 & -1 & 0 \end{pmatrix} \end{matrix}$$

Note that each permutation in formula (5) can be used to select a subset of elements in matrix M^{s^A, s^B} in such a way that exactly one element is selected from each row and each column. This is illustrated by the following formula:

$$\begin{aligned} \Pi_f^A(\sigma(s^A), \sigma(s^B)) &= \frac{1}{n!} \sum_{p \in P_n} \pi_f^A(s^A, p(s^B)) = \\ &= \frac{1}{n!} \sum_{p \in P_n} f(M_{1,p(1)}^{s^A, s^B}, \dots, M_{n,p(n)}^{s^A, s^B}) \end{aligned} \quad (12)$$

We can think of the problem of computing a single payoff from a pair of symmetric strategies as follows. A matrix M^{s^A, s^B} is a chessboard of dimension $n \times n$, divided into three distinct areas:

$$W^{s^A, s^B} = \{(i, j) \in [n] \times [n] : M_{i,j}^{s^A, s^B} = 1\},$$

$$T^{s^A, s^B} = \{(i, j) \in [n] \times [n] : M_{i,j}^{s^A, s^B} = 0\},$$

$$L^{s^A, s^B} = \{(i, j) \in [n] \times [n] : M_{i,j}^{s^A, s^B} = -1\}.$$

Each permutation $p \in P_n$ corresponds to distributing n rooks on the mentioned chessboard of dimension $n \times n$ in such a way that no two rooks attack each other. Each row contains one rook and the rook in the i 'th row is located in $p(i)$ 'th column. For $D \in \{W, T, L\}$, let $K_D(p | M)$ denote the number of rooks distributed in area D of clash matrix M by permutation p . By Observation 1, it follows that:

$$\pi_f^A(s^A, p(s^B)) = f_n(K_W(p | M), K_L(p | M)). \quad (13)$$

Let $h(k_W, k_L | M)$ be the number of permutations $p \in P_n$ that place exactly k_W rooks in area W and k_L rooks in area L for a given clash matrix M , i.e.

$$h(k_W, k_L | M) = \sum_{p \in P_n} [K_W(p | M) = k_W \wedge K_L(p | M) = k_L].$$

Using function h and (13) we can redefine payoff (12):

$$\Pi_f^A(\sigma(s^A), \sigma(s^B)) = \frac{1}{n!} \sum_{k_W=0}^n \sum_{k_L=0}^n h(k_W, k_L) f_n(k_W, k_L). \quad (14)$$

For better clarity, we omit M^{s^A, s^B} parameter in h and f_n notation. In the following subsections, we show how to compute values of function h for a given clash matrix.

5.2 Properties of clash matrices

Note that due to the non-increasing ordering of the strategy vectors of both players, each column and row of a clash matrix describes a monotonic sequence of length n . Using this observation, we conclude that all clash matrices share a very particular structure. Area W occupies the upper part of the matrix, potentially reaching lower and lower in each successive column, and area T consists of rectangles, perhaps touching one another at the corners but never sharing a side. Example 5.2 illustrates such a matrix.

When looking for the recursive formula for function h we will be "cutting off" certain parts of the matrix (submatrix) with a certain number of distributed rooks and thus reducing the size of the considered matrix (submatrix).

Example 5.2 (Successive cut-offs of the clash matrix). An example of the clash matrix with the successive "cut-offs", marked by their numbers is shown in Figure 2. Areas are colored as follows: yellow (L), blue (T), and red (W). An example of a strategy pair that yields the considered clash matrix is $s_A = (8, 8, 6, 5, 4, 2, 1, 1, 0)$, $s_B = (8, 8, 8, 7, 5, 3, 3, 1, 0)$.

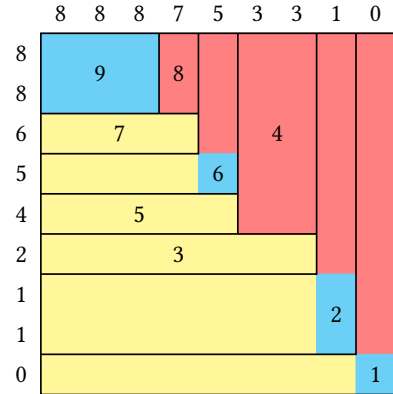


Figure 2: Clash matrix with successive cut-offs

5.3 Recursive formula

In this section, we derive a recursive formula describing the number of possible distributions of rooks that do not attack one another on a chessboard (clash matrix) with designated areas W , L , and T . Let:

$M_{|(i,j)}$ – the submatrix of M consisting of the intersection of the first i rows and first j columns of M .

$H(i, j, m, k_W, k_L | M)$ – the number of ways in which m rooks can be distributed in the intersection of the first i rows and j columns of the clash matrix M , such that there are

exactly k_W rooks in area W and exactly k_L rooks are in area L .

$R(i, j, t) = \binom{j}{i} \binom{i}{t} t!$ – the number of ways to distribute t rooks in a uniform area with i rows and j columns.

From the definition of H it follows that

$$h(k_W, k_L | \mathbf{M}) = H(n, n, n, k_W, k_L | \mathbf{M}).$$

To find the recursion, in each step, we choose the number of columns and rows so as to cover the entire coherent section of T lying in the right lower corner. If the corner lies in L , we say that the width of the considered section of T is 0. When the corner is in W , we say that the height of the considered section of T is 0. The recursive formula is expressed as follows:

$$H_1(i, j, m, k_W, k_L | \mathbf{M}) = \sum_{\substack{r_1, r_2, r_3 \geq 0 \\ r_1 + r_2 + r_3 \leq m}} H(i', j', m - r_s, k_W - r_3, k_L - r_1 | \mathbf{M}_{|(i', j')}) \cdot R(i - i', j' - (m - r_s), r_1) \cdot R(i - i' - r_1, j - j', r_2) \cdot R(i' - (m - r_s), j - j' - r_2, r_3),$$

where $r_s = r_1 + r_2 + r_3$. To show the recursion, we consider every possible division of m rooks into four groups, as indicated below in Figure 3. When the height or width of the considered section of T is 0, two of these groups have a size of 0.

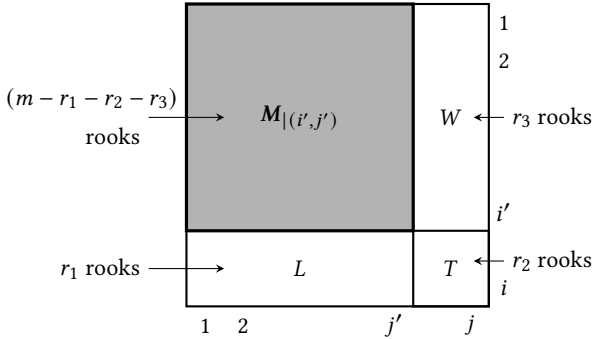


Figure 3: Cutting-off procedure

We can think of it as follows:

- (1) We place $(m - r_1 - r_2 - r_3)$ rooks in area $M_{|(i', j')}$.
- (2) In the rectangle lying in L (whose $(m - r_1 - r_2 - r_3)$ columns have already been excluded in (1)) we place r_1 rooks in $i - i'$ free rows and $j' - (m - r_1 - r_2 - r_3)$ free columns.
- (3) In the rectangle lying in T (whose r_1 rows have already been excluded in (2)), we place r_2 rooks in $i - i' - r_1$ free rows and $j - j'$ free columns.
- (4) In the rectangle lying in W (whose $(m - r_1 - r_2 - r_3)$ rows and r_2 columns have already been excluded in (1) and (3)) we place r_3 rooks in $i' - (m - r_1 - r_2 - r_3)$ free rows and $j - j' - r_2$ free columns.
- (5) As r_3 rooks were placed in W and r_1 rooks were placed in L , we subtract those values from k_W and k_L respectively in the recursive call of H in point (1).

Algorithm 1: The dynamic algorithm for calculating payoff from a pair of symmetric strategies

Input: Two non-increasing vectors s^A and s^B
Result: Payoff value from a given profile.

```

/* initialization */
Determine the clash matrix;
Determine the array knots[], containing pairs of
coordinates of consecutive points at which the “cut-off”
occurs, in ascending order;
Allocate a 4-dimensional array values[], of size
2n × (n + 1) × (n + 1) × (n + 1);
for knotIndex ← 0 to length(knots) - 1 do
    knot ← knots[knotIndex];
    i ← knot[0];
    j ← knot[1];
    for rooksNum ← 0 to min(i, j) do
        for k_W ← 0 to min(i, j) do
            for k_L ← 0 to min(i, j) do
                values[knotIndex, rooksNum, k_W, k_L] ←
                    H(i, j, rooksNum, k_W, k_L);
                /* function H uses value[] and clash
                    matrix */
lastKnot ← length(knots) - 1;
/* last knot is always (n, n) */
result = 0;
for k_W ← 0 to n do
    for k_L ← 0 to n do
        result += values[lastKnot, n, k_W, k_L] · f_n(k_W, k_L)
return result / factorial(n);

```

Using the formula described above, we will eventually arrive at a situation where submatrix M is entirely within one of the three areas under consideration. Then:

$$H_0(i, j, m, k_W, k_L | \mathbf{M}) = \begin{cases} R(i, j, m), & \text{if } k_W = m \text{ and } k_L = 0 \text{ and } \mathbf{M} \subseteq W, \\ R(i, j, m), & \text{if } k_L = m \text{ and } k_W = 0 \text{ and } \mathbf{M} \subseteq L, \\ R(i, j, m), & \text{if } k_L = k_W = 0 \text{ and } \mathbf{M} \subseteq T, \\ 0, & \text{otherwise.} \end{cases}$$

5.4 Dynamic algorithm

Values of the recursive formula described above can be computed using dynamic programming using Algorithm 1. Algorithm 1 runs in memory limited by $O(n^4) = 2n \cdot (n + 1) \cdot (n + 1) \cdot (n + 1)$ and requires no more than $O(n^7) = 4 \cdot 2n \cdot (n + 1) \cdot (n + 1) \cdot (n + 1) \cdot \binom{n+3}{3}$ arithmetic operations.

6 DOUBLE ORACLE ALGORITHM

In Section 5 we presented an algorithm for computing single payoffs from symmetric strategy profiles in polynomial time with respect to the model parameters. Sizes of the sets of symmetric strategies of both players, although much smaller, are still exponential with respect to the model parameters. Because of that, LP based methods

for solving zero-sum games, which require calculating the whole payoff matrix, are still inefficient for large parameters values. For that reason we use the Double Oracle Algorithm (DOA), described by Algorithm 2, which can be used for finding NE of zero-sum games without calculating the whole payoff matrix. Double Oracle Algorithm correctness was proven in [15].

6.1 Oracle

When using DOA, one has to use an *oracle* that for a given mixed strategy of a player C returns a best-response pure strategy of the opponent. Given a mixed strategy ξ^C of player C , we consider all the strategy profiles (ξ^C, s^{-C}) , where s^{-C} is a pure strategy of player $-C$, and find a pure strategy that maximizes payoff for player $-C$. To compute these payoffs we use Algorithm 1. Given two sets of pure strategies, X^A and X^B , of players A and B , respectively, by $\text{CoreLP}(X^A, X^B)$ we mean a strategy profile that is NE of the zero-sum game restricted to the pure strategies in X^A and X^B , found by an LP-solver.

Algorithm 2: The Double Oracle Algorithm for solving zero-sum games

```

Result: NE of the zero-sum game
/* initialization */
1. Initialize  $X^A$  with one pure strategy of player A;
2. Initialize  $X^B$  with one pure strategy of player B;
repeat
   $(\xi^A, \xi^B) \leftarrow \text{CoreLP}(X^A, X^B)$ ;
   $X^A \leftarrow X^A \cup \{\text{best\_response}(\xi^B)\}$ ;
   $X^B \leftarrow X^B \cup \{\text{best\_response}(\xi^A)\}$ ;
until convergence;
return  $(\xi^A, \xi^B)$ ;

```

As we can choose the initial strategies of both players arbitrarily, we decided to start with the most even assignment between all the battlefields for each player.

6.2 Proposed heuristic

In this section we propose a simple, but effective, heuristic that allows us to avoid unnecessary calculations for a subset of strategies. We first prove a simple observation regarding the pure strategies that are the best responses to a mixed strategy of the opponent. By a *best response* of player C to a mixed strategy ξ^{-C} of the opponent we mean a pure strategy s_{best}^C that maximizes the payoff of C against ξ^{-C} .

Definition 6.1 (Maximal assignment). We define $\max_assign(\xi^C) \in N$, of player C as

$$\max_{s \in S_C} \max_{i \in \{1, \dots, n\}} s_i, \quad (15)$$

$$\xi^C(s) > 0$$

i.e. the biggest number of resources that player C assigns to single battlefield with positive probability when playing the strategy ξ^C .

Let f be an aggregation function that is monotonic non-decreasing in its first argument and monotonic non-increasing in its second

argument, meaning that a player cannot decrease her payoff by increasing the number of battlefields won and cannot increase her payoff when the number of battlefields won by the opponent increases. When the disproportion of resources between the players, meaning $|D_A - D_B|$ is not greater than n , the following proposition always holds.

PROPOSITION 6.2. *For any mixed strategy ξ^C of player C and any aggregation function f as described above, there exists a best response pure strategy s_{best}^{-C} of player $-C$, that satisfies*

$$\max_assign(s_{best}^{-C}) \leq \max_assign(\xi^C) + 1. \quad (16)$$

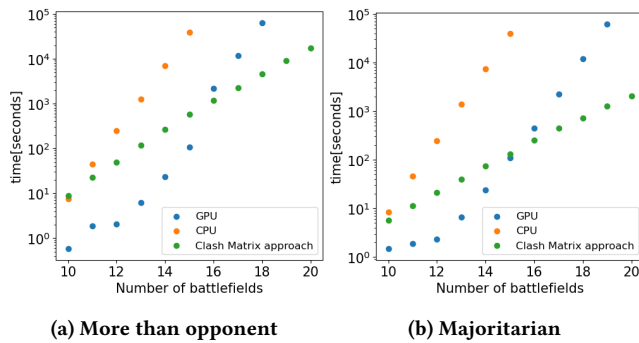
PROOF. Take any pure strategy s^{-C} that maximizes payoff of player $-C$ against a given mixed strategy ξ^C . If this strategy does not satisfy (16) then we can reduce the number of resources at battlefields where the number of resources exceeds $(\max_assign(\xi^C) + 1)$ to $(\max_assign(\xi^C) + 1)$, as those battlefields are still always won by $-C$, and increase the number of resources at the battlefields where the current assignment is below $(\max_assign(\xi^C) + 1)$. As the aggregation function is monotonic non-decreasing in its first argument and monotonic non-increasing in its second argument, the payoff of player $-C$ does not decrease after such a redistribution. \square

By Proposition 6.2, when computing a best response to a given mixed strategy ξ^C , we can restrict attention to the pure strategies of the opponent for which the \max_assign value exceeds the \max_assign value of the mixed strategy ξ^C by at most one. This significantly reduces the number of necessary calculations and therefore the time required to compute the NE of the game in question.

7 EXPERIMENTAL EVALUATION

We implemented two versions of Algorithm 1 for finding payoff matrices of the symmetrized games with the aggregation functions, one given by (7) and one given by (6), using C++'s *vector* class from C++'s standard library. We compared the acquired run times of the algorithm with CPU and GPU-based approaches that calculate payoffs based on (5) for finding payoff matrices of the symmetrized game. Experiments were run on a computational cluster XXXX (anonymization) with a central processing unit (CPU) Intel Xeon E5-2640 v4 (10 cores) and the GPU unit Titan V. Four cores of the processor were used while conducting each of the experiments. All the time results in the experiments are averages from 10 runs of the program. The program we used to obtain payoff matrices with CPU and GPU is proposed and described in [14]. Figure 4 shows the comparison of the times required by all three methods for finding the whole payoff matrices of both games. The numbers of battlefields are shown on the horizontal axis. The number of resources of each player is a linear function of the number of battlefields (the number of battlefields increased by 5). The vertical axis shows the run times of the programs using a logarithmic scale. The time limit for each experiment was set to 10^5 seconds.

When considering the *more than opponent* (Figure 4a) game, the maximal value of parameters that GPU based approach was able to calculate the payoff matrix for was 23 resources to distribute over 18 battlefields. The achieved speedup of the clash matrix for



(a) More than opponent

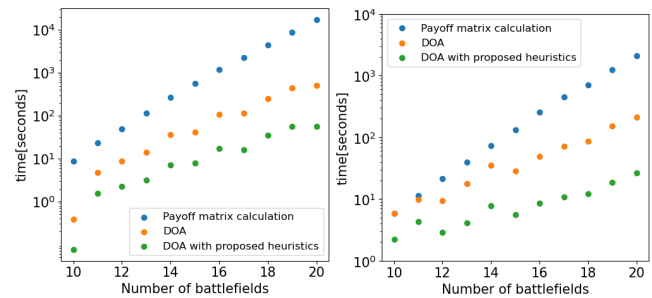
(b) Majoritarian

Figure 4: Run times to the number of battlefields

the same parameters was 18 times. For the *majoritarian* (Figure 4b) game, the maximal value of parameters that GPU based approach was able to calculate the payoff matrix for was 24 resources to distribute over 19 battlefields. The achieved speedup of the clash matrix for the same parameters was 48 times. The speedup for 23 resources to distribute over 18 battlefields was 16 times (similar for both aggregation functions).

Although our method is significantly faster, the time required to calculate the payoff matrix is still the bottleneck for solving the models in question. Therefore we include Figure 5, which shows a comparison of the time results of the calculation of the entire payoff matrix of the game, with the two versions of DOA (with and without the proposed heuristic) that yield a Nash Equilibrium of a game. We use the time results for calculating the entire payoff matrix as a time-bound on every approach that calculates the entire payoff matrix and then solves the game using this matrix (e.g. the standard LP-solver approach). All of the programs use a symmetrized model and calculate single payoffs using the clash matrix. The numbers of battlefields are shown on the horizontal axis. The number of resources of each player is once again a linear function of the number of battlefields (the number of battlefields increased by 5). We used the state-of-the-art *Gurobi* [7] LP-solver for solving the matrix games as well as finding the NE when using DOA (CoreLP function).

For the *more than opponent* game (Figure 5a), speed-up achieved for the game where both players have 25 resources to distribute over 20 battlefields by the DOA algorithm when compared to the LP-solver is 34 times. In contrast, when using the DOA algorithm with the heuristic that we propose, one gets a speed-up of more than 300 times. When considering the *more than opponent* game (Figure 5b), speed-up achieved for the game where both players have 25 resources to distribute over 20 battlefields by the DOA algorithm when compared to the LP-solver is 10 times. In contrast, when using the DOA algorithm with the heuristic that we propose, one gets a speed-up of more than 80 times. This shows that the proposed heuristic, although simple, is very effective in both models. When comparing the LP-solver-based approach, where payoff matrices are computed using a GPU, and the clash matrix-based DOA approach, using the proposed heuristic, the achieved speed-up for the game where both players have 25 resources to distribute over 20 battlefields would be more than 3000 times for both models.



(a) More than opponent

(b) Majoritarian

Figure 5: Run times of different algorithms

Unfortunately, the run time of the GPU-based method for finding the payoff matrix takes too long for us to know the exact speedups that could be achieved.

8 CONCLUSIONS

In this paper, we consider a class of conflicts with multiple battlefields with discrete resources and uniform battlefields values. We show that the NE of the models in question can be found by examining the symmetrized models, where the number of strategies, although still exponential with respect to the model parameters, is reduced by an exponential factor.

Our contribution has two main components. First, we propose a *clash matrix algorithm* to obtain a single payoff for a pair of symmetric strategies in polynomial time. We carry out a comparison of run times required to compute the payoff matrices using the clash matrix algorithm and naïve algorithms (CPU and GPU based). Our algorithm, which works for the whole class of the described models, is significantly faster in practice than both mentioned approaches. The symmetrization combined with the proposed clash matrix algorithm allows for reducing the number of strategies by an exponential factor at the expense of polynomial time for computing the payoffs.

Second, we apply the Double Oracle Algorithm to find an NE of the Chopstick Auctions. Using DOA with the heuristic that we propose, combined with the clash matrix approach for calculating single payoffs gives significant speed-up for the considered subclass of games.

ACKNOWLEDGMENTS

This work was supported by the Polish National Science Centre through grant 2018/29/B/ST6/00174.

REFERENCES

- [1] A. Ahmadinejad, S. Dehghani, M. Hajiaghayi, B. Lucier, H. Mahini, and S. Seddighin. 2019. From Duels to Battlefields: Computing Equilibria of Blotto and Other Games. *Mathematics of Operations Research* 44, 4 (2019), 1304–1325.
- [2] S. Behnezhad, A. Blum, M. Derakhshan, M. Hajiaghayi, M. Mahdian, C. Papadimitriou, R. Rivest, S. Seddighin, and P. Stark. 2018. From Battlefields to Elections: Winning Strategies of Blotto and Auditing Games. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*. 2291–2310.
- [3] S. Behnezhad, S. Dehghani, M. Derakhshan, M. T. Hajiaghayi, and S. Seddighin. 2017. Faster and Simpler Algorithm for Optimal Strategies of Blotto Game. In

- Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4–9, 2017, San Francisco, California, USA.* 369–375.
- [4] É. Borel. 1921. La Théorie du Jeu et les Équations Intégrales à Noyau Symétrique. *Comptes Rendus de l'Académie des Sciences* 173 (1921), 1304–1308. Translated by Savage L. J., The theory of play and integral equations with skew symmetric kernels. *Econometrica* 21 (1953) 97–100.
- [5] M. Carvalho, A. Lodi, and J. Pedroso. 2022. Computing equilibria for integer programming games. *European Journal of Operational Research* 303, 3 (2022), 1057–1070.
- [6] F. Englmaier, P. Guillén, L. Llorente, S. Onderstal, and R. Sausgruber. 2009. The chopstick auction: A study of the exposure problem in multi-unit auctions. *International Journal of Industrial Organization* 27, 2 (2009), 286–291.
- [7] Gurobi Optimization, LLC. 2022. Gurobi Optimizer Reference Manual.
- [8] G. H. Hardy and S. Ramanujan. 1918. Asymptotic formulae in combinatory analysis. *Proc. London Math. Soc* 17, 2 (1918), 75–115.
- [9] S. Hart. 2008. Discrete Colonel Blotto and General Lotto games. *International Journal of Game Theory* 36 (02 2008), 441–460.
- [10] Manish Jain, Dmytro Korzhuk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. 2011. A Double Oracle Algorithm for Zero-Sum Security Games on Graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1 (Taipei, Taiwan) (AAMAS '11)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 327–334.
- [11] M. Köppe, C. Ryan, and M. Queyranne. 2011. Rational Generating Functions and Integer Programming Games. *Operations Research* 59 (2011), 1445–1460.
- [12] D. Korzhuk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. 2011. Stackelberg vs. Nash in Security Games: An Extended Investigation of Interchangeability, Equivalence, and Uniqueness. *Journal of Artificial Intelligence Research* 41 (2011), 297–327.
- [13] D. Kovenock and B. Roberson. 2010. Conflicts with Multiple Battlefields. *Purdue University, Department of Economics, Purdue University Economics Working Papers* (01 2010).
- [14] J. Litwin. 2020. Equilibria of chopstick auctions with discrete resources.
- [15] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *ICML*, Tom Fawcett and Nina Mishra (Eds.). AAAI Press, 536–543.
- [16] J. Pita, M. Jain, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. 2009. Using Game Theory for Los Angeles Airport Security. *AI Magazine* 30, 1 (2009), 43–57.
- [17] R. Porter, E. Nudelman, and Y. Shoham. 2008. Simple search methods for finding a Nash equilibrium. *Games and Economic Behavior* 63, 2 (2008), 642–662.
- [18] T. Raghavan. 1994. Zero-sum two-person games. In *Handbook of Game Theory with Economic Applications* (1 ed.), R. Aumann and S. Hart (Eds.). Vol. 2. Elsevier, Chapter 20, 735–768.
- [19] B. Szentes and R. Rosenthal. 2003. Beyond chopsticks: Symmetric equilibria in majority auction games. *Games and Economic Behavior* 45, 2 (2003), 278–295. Special Issue in Honor of Robert Rosenthal.
- [20] B. Szentes and R. Rosenthal. 2003. Three-object two-bidder simultaneous auctions: Chopsticks and tetrahedra. *Games and Economic Behavior* 44 (07 2003), 114–133.
- [21] J. von Neumann. 1953. A certain zero-sum two-person game equivalent to the optimal assignment problem. In *Contributions to the Theory of Games (AM-28), Volume II*. Princeton University Press, 5–12.
- [22] Haifeng Xu. 2016. The Mysteries of Security Games: Equilibrium Computation Becomes Combinatorial Algorithm Design. *Proceedings of the 2016 ACM Conference on Economics and Computation* (2016). <https://api.semanticscholar.org/CorpusID:707610>