New Approximation for Borda Coalitional Manipulation

Orgad Keller
Department of Computer Science
Bar-Ilan University
Israel
orgad.keller@gmail.com

Avinatan Hassidim
Department of Computer Science
Bar-Ilan University
Israel
avinatan@cs.biu.ac.il

Noam Hazon
Department of Computer Science
Ariel University
Israel
noamh@ariel.ac.il

ABSTRACT

We study the problem of Borda Unweighted Coalitional Manipulation, where k manipulators try to manipulate an election on m candidates under the Borda protocol. This problem is known to be NP-hard. While most recent approaches to approximation tried to minimize k, the number of manipulators needed to make the preferred candidate win (thus assuming that the number of manipulators is not limited in advance), we focus instead on minimizing the maximum score obtainable by a non-preferred candidate. We provide a randomized, additive $O(k\sqrt{m \log m})$ approximation to this value; in other words, if there exists a strategy enabling the preferred candidate to win by an $\Omega(k\sqrt{m \log m})$ margin, our method, with high probability, will find a strategy enabling her to win (albeit with a possibly smaller margin). It thus provides a somewhat stronger guarantee compared to the previous methods, where the addition of an extra manipulator implied (with respect to the original k) a strategy that provides an $\Omega(m)$-additive approximation to a runner-up's score: when k is $o(\sqrt{m/\log m})$, our strategy provides a stronger approximation. Our algorithm can also be viewed as a $(1 + o(1))$-multiplicative approximation since the value we approximate has a natural $\Omega(km)$ lower bound.

Our methods are novel and adapt techniques from multiprocessor scheduling by carefully rounding an exponentially-large configuration linear program that is solved by using the ellipsoid method with an efficient separation oracle. We believe that such methods could be beneficial in approximating coalitional manipulation in other election protocols as well.

CCS Concepts

•**Computing methodologies** → **Multi-agent systems;**

Keywords

Elections, Borda Voting Protocol, Coalitional Manipulation

1. INTRODUCTION

Elections are one of the pillars of democratic societies, and are an important part of social choice theory. In addition they have played a major role in multiagent systems, where a group of intelligent agents would like to reach a joint decision [9]. In its essence, an election consists of n agents (also called voters) who need to decide on a winning candidate among m candidates. In order to do so, each voter reveals a ranking of the candidates according to his preference and the winner is then decided according to some protocol.

Ideally in voting, we would like the voters to be truthful, that is, that their reported ranking of the candidates will be their true one. However, almost all voting rules are prone to manipulation: Gibbard and Satterthwaite [11, 18] show that for any reasonable preference-based voting system with at least 3 candidates, voters might benefit from reporting a ranking different than their true one in order to make sure that the candidate they prefer the most wins. Furthermore, several voters might decide to collude, to form a coalition and then to coordinate their votes in such a way that a specific candidate p (hereafter the *preferred candidate*) will prevail. Such a setting is reasonable especially when the voters are agents that are operated by one party of interest.

For some time, the hope for making voting protocols immune to manipulations at least *in practice* relied on computational assumptions: for several common voting protocols, it was shown that computing a successful voting strategy for the manipulators is NP-hard [3, 6, 22, 10]. However, as it is many times the case, approximation algorithms and heuristics were devised in order to overcome the NP-hardness albeit with some compromises on the quality of the resulting strategy. This paper fits within this scheme.

In this paper we focus on the Borda voting rule and study the problem of Borda (constructive) unweighted coalitional manipulation (*Borda-UCM*)[1]: assume that k additional voters (hereafter the manipulators), all of them preferring a specific candidate p, can be added to the voting system, thus forming a coalition. Also assume that all n original voters (hereafter the non-manipulators) voted first (or equivalently, that the non-manipulators are truthful and that their preferences are known). Find a strategy for the manipulators telling each one of them how to vote so that p wins, if such strategy exists.

Borda-UCM was conjectured and then proven to be NP-hard [7, 4]. As a way of overcoming this, recent research [23] focused on an approximation to the minimum number of manipulators needed to be added to the system in order to guarantee that the preferred candidate p would win. They showed that if there exists some strategy for k manipulators

[1]The problem was called Borda-CCUM, for "constructive coalitional unweighted manipulation", in [23].

Appears in: *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

making p win, then they will find a strategy making her win with at most one additional manipulator.

This kind of approximation might seem a bit problematic: while in some cases it might be reasonable that the party behind the manipulators can add another manipulator to the system, in many cases we do not expect this to be true. Instead, it is interesting to ask what can we assert—assuming that the number of manipulators cannot be changed—on the ability to promote a specific candidate p given the non-manipulator scores of all candidates and the value k.

We provide a positive result of the following type:

> **Main Result:** *If there exists a manipulation strategy enabling p to win by a large-enough margin, we efficiently find a successful manipulation strategy making p win.*

Specifically, assume that we can provide, for some function $f(k, m)$, an $f(k, m)$-additive approximation to the maximum difference obtainable between p's final score and the final score of the highest-scoring non-preferred candidate. Then, if there exists a strategy such that this difference is at least $f(k, m)$, we can be rest-assured that the algorithm will find a strategy enabling p to win.

This, in turn, boils down to approximating an upper-bound to the score of the highest-ranked candidate who is not p. Earlier research of this flavor only focused on the case where the number of candidates is bounded: for the weighted case, Brelsford et al. [5] provide an FPTAS to that upper bound (to be exact, they provide an FPTAS to the same exact value we defined, and then use it to provide another FPTAS to another value, which is their value-of-interest.[2]) For the ordinary unweighted case, if the number of candidates is bounded, the entire problem becomes easy and polynomial-time solvable [6, Proposition 1].

1.1 Our Results and Contributions

Let T^* be the minimum possible score (ranging over all possible manipulation strategies S) of the highest scoring candidate who is not p. Formally, it is defined as $T^* = \min_S \max_{c' \in C \setminus \{p\}} s(c')$, where $s(c')$ is the final score of a candidate c', and C is the candidate set. Our main contribution is a constructive proof to the following theorem:

THEOREM 1. *There exists a randomized Monte Carlo algorithm for Borda-UCM which provides an $O(k\sqrt{m \log m})$-additive approximation to T^* with an exponentially-small failure probability.*

In other words, if there exists a strategy enabling p to win by a margin of $\Omega(k\sqrt{m \log m})$ compared to the score of the highest-scoring non-preferred candidate, our method will find a strategy enabling p to win (albeit with a possibly smaller margin). Notice that such an approximation can also be seen as a $(1 + o(1))$-multiplicative approximation on the score of the highest-scoring non-preferred candidate, and thus is superior to an FPTAS; to see that, notice that the overall 'voting mass' given by the manipulators is $\Omega(km^2)$, and so the highest scoring candidate has score of at least $\Omega(km)$. Therefore $\tilde{O}(k\sqrt{m})$ [3] is a lower order term. This is an advantage over the previous methods:

- Opposed to the heuristics in [8], we provide *provable* guarantees.
- Consider the REVERSE algorithm of [23], and assume that adding extra manipulators is not allowed. We will show that their method implies no better than an $\Omega(m)$-additive approximation to the score of the highest-scoring non-preferred candidate. Our approximation is thus superior when k is $o(\sqrt{m/\log m})$.

The following claim analyzes REVERSE according to our metric. It is proven in Section 3.

CLAIM 2. *For any m, when the addition of more than k extra manipulators is not allowed, there are cases in which the optimal strategy enables p to win by a margin of at least $m/3$, but* REVERSE *fails to find a strategy enabling p to win.*

Our techniques are novel: they employ the use of *configuration linear programs* (C-LP), a method that is also used in the scheduling literature, namely for two well-studied problems, the problem of *scheduling on unrelated machines* [21], and the so-called *Santa Claus problem* [2]. See Section 1.2 for a discussion of these problems. It is important to note that the solutions to the two above problems and to ours all differ from one another with respect to how the algorithm proceeds once the C-LP result is computed.

C-LPs are used for the generation of an initial, invalid strategy, which is later modified to become valid. They are unique in the sense that they are linear programs that have an exponential number of variables, an issue which we solve by referring to the LP dual and using the ellipsoid method with a polynomially-computable separation oracle [16, 12]. We also implemented our algorithm: as a result of not finding a library which enables solving an LP this way, we simulated this by an iterative use of a general LP-solving library, each time adding a violated constraint based on running the separation oracle externally.

1.2 Related Work

The Borda voting mechanism was introduced by Jean-Charles de Borda in 1770. It is used, sometimes with some modifications, by parliaments in countries such as Slovenia, and competitions such as the Eurovision song contest, selecting the MVP in a major league baseball, Robocup robot soccer competitions and others. The Borda voting mechanism is described as follows: every agent ranks the candidates from 1 to m, and awards the candidate ranked i-th a score of $m - i$. Notice that this makes the scores given by each single voter a permutation of $0, \ldots, m-1$. Finally, the winning candidate is the one with the highest aggregated score.

The computational complexity of Borda coalitional manipulation was studied extensively. For the more general case in which each voter can have an associated weight, the problem was shown to be NP-hard [6, 13]. However, the natural, unweighted case still remained open for quite some time, until shown to be NP-hard as well [7, 4] in 2011, even for the case of $n = 3$ and adding 2 manipulators.

[2] They are interested in the difference between the score of the preferred candidate and the highest-scoring non-preferred candidate when including the manipulator votes, minus the same difference when not including the manipulator votes. Notice that the upper-bound we defined is the only non-trivial value in this computation.

[3] The $\tilde{O}$ notation suppresses poly-logarithmic factors.

Several approximation algorithms and heuristics [20] were devised. Zuckerman et al. [23] present REVERSE[4], a greedy method which works as follows: after the non-manipulators had finished voting, we go over the manipulators one by one, and each manipulator will rank the candidates (besides p) by the *reversed* order of their aggregated score so far (candidate with the highest score so far gets the lowest ranking). As mentioned, REVERSE can be seen as an additive $+1$ approximation for the objective of finding the minimum number of manipulators needed for Borda manipulation; in the weighted case, the added manipulator should have weight $\max_{\ell=1,\ldots,k} w_\ell$, where w_ℓ is the weight of manipulator ℓ. Davies et al. [8] present two additional heuristics: iteratively, assign the largest un-allocated score to the candidate with the largest gap (LARGEST FIT), or to the candidate with the largest ratio of gap divided by the number of scores yet-to-be-allocated to this candidate (AVERAGE FIT).

As discussed, configuration linear programs were also used in scheduling literature, for example for the following two problems which were extensively studied before:

- In the so-called *Santa Claus problem* [2], Santa Claus has t presents that he wishes to distribute between m kids, and $p_{i,j}$ is the value that kid i has to present j. The goal is to *maximize* the happiness of the least happy kid: $\min_i \sum_{j \in S_i} p_{i,j}$, where S_i is the presents allocated to kid i.
- In the problem of *scheduling on unrelated machines* [21]. We need to assign t jobs between m machines, and $p_{i,j}$ is the time required for machine i to execute job j. The goal is to *minimize* the makespan $\max_i \sum_{j \in S_i} p_{i,j}$, where S_i is the jobs assigned to machine i.

Both papers researched a natural and well-researched 'restricted assignment' variant of the two problems where $p_{i,j} \in \{p_j, 0\}$. In [2], they obtained an $O(\log\log m/\log\log\log m)$-multiplicative approximation to the first problem and in [21], they obtained a $(33/17+\epsilon)$ -multiplicative approximation to the second.

2. PRELIMINARIES

2.1 Problem Definition

With a slight change of notation, let $C = \{c_0, c_1, \ldots, c_m\}$ be a candidate set consisting of the preferred candidate $p = c_0$ and the other m candidates $c_1, \ldots, c_m$. Note that we changed the notation so that the overall number of candidates is $m+1$; this will help streamline the writing.

An election is defined by a candidate set C and a set of voters V, where each voter submits a ranking of the candidates. Then, some *decision rule* $\mathcal{R}$ is applied in order to decide on the winner. In the specific case of a *positional scoring rule* $\mathcal{R}_{\vec{\alpha}}$, the rule is described by a vector $\vec{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_m)$, used as follows: each voter awards α_i to the candidate ranked i-th. Finally, the winning candidate is the one with the highest aggregated score. In this paper we will focus on the Borda scoring rule, in which $\vec{\alpha} = (m, m-1, \ldots, 0)$.

In the Borda (constructive) unweighted coalitional manipulation (Borda-UCM) problem, we are also given as input an integer k representing the number of manipulators and a score profile vector $(\sigma_0, \sigma_1, \ldots, \sigma_m)$ representing the aggregated scores given so far to each candidate in C by the non-manipulators. It should then be determined if either (a) no strategy exists in which p wins, or that (b) there exists a voting strategy under Borda such that p can win. In this case, the algorithm should find it.

Note that in case (b), the output is a voting strategy S which can be represented as a $k \times (m+1)$ matrix in which the entry $S_{t,i}$ describes the score given by voter t to candidate c_i, and where each row of S is a permutation of $\{0, \ldots, m\}$. Such a representation is also called a *manipulation matrix*. We can relax the requirement that each row of S is a permutation, and replace it by the requirement that each *score-type* j, is repeated exactly k times in S. Such a matrix is called a *relaxed manipulation matrix*. We can perform this relaxation as Davies et al. [8, Theorem 7] show that each relaxed manipulation matrix can be rearranged to become a valid manipulation matrix while preserving each candidate's final score.

Throughout the paper, when we use the term 'with high probability', we mean an arbitrarily-chosen polynomially-small failure probability, i.e., success probability of the form $1-m^{-d}$ where $d \geq 1$ is a constant that can be chosen without affecting the asymptotic running time. 'Failure' refers to the event that the algorithm does not provide the desired approximation guarantee.

2.2 Reduction to a Pure Min-Max Problem

Since we know what will be the final score of p (non-manipulator votes are known and each manipulator will give p the maximum score possible), we can effectively discard p and treat the problem as a minimization problem on the final scores of $c_1, \ldots, c_m$ only. In other words, we focus on finding $\min_S \max_{c' \in C \setminus \{p\}} s(c') = \min_S \max_{i=1,\ldots,m} s(c_i)$, where $s(c')$ is c''s final score. Thus the output S is actually a $k \times m$ relaxed manipulation matrix.

Another thing to note is that we can not assume anything about the values in the initial score profile $(\sigma_0, \sigma_1, \ldots, \sigma_m)$; this follows from [8, Lemma 1], where it is shown that in the context of Borda, for any given non-negative integer vector $(\sigma_0, \sigma_1, \ldots, \sigma_m)$, we can define a set of non-manipulators (along with their preferences) and an additional candidate c_{m+1} that will induce an initial score profile $(T+\sigma_1, \ldots, T+\sigma_m, y)$ for some values T and $y < T$. Since such an additive translation and the addition of a candidate that will be awarded less than any other candidate have no influence on the winner (and on the difference between each two candidates' final scores), and since our results are concerned with an *additive* approximation, we should assume no prior limitation on the nature of values in $(\sigma_0, \sigma_1, \ldots, \sigma_m)$.

3. LOWER BOUND FOR REVERSE

We start by showing that there are cases in which the REVERSE algorithm for Borda only gives an $\Omega(m)$-additive approximation to minimum final score of the highest-scoring non-preferred candidate.

PROOF OF CLAIM 2. We provide an infinite family of cases where the claim holds.

Let $k = 3$ and let $m = 3t$ for some integer t. Consider the case where after the non-manipulators voted, all candidates (but p) have the same score $\sigma_i = s$ for all i. Effectively this can be normalized to $(\sigma_1, \ldots, \sigma_m) = \vec{0}$.

[4]This name was given in [7].

By the REVERSE algorithm, the first manipulator can award $c_1, \ldots, c_m$ with $0, \ldots, m-1$ respectively, after which the second manipulator will be obliged to award $c_1, \ldots, c_m$ with $m-1, \ldots, 0$ respectively. Repeat this process with the rest of the manipulators, until the final one. It can be verified that c_m will end up with the maximal score of $\lceil k/2 \rceil (m-1) = 2(m-1)$.

Conversely, as an upper bound for an optimal solution, consider the following strategy: place all scores to be given in a descending sequence, that is the sequence $\langle m-1, m-1, m-1, m-2, m-2, m-2, \ldots, 0, 0, 0 \rangle$. Give the first m scores in the sequence to $c_1, \ldots, c_m$ respectively, the next m to $c_m, \ldots, c_1$ respectively, and the last m to $c_1, \ldots, c_m$ respectively. Since every score-type has 3 copies, we have just described a relaxed manipulation matrix and therefore by Davies et al. [8, Theorem 7] it can be rearranged to become a valid manipulation matrix without changing the final score of each candidate. Now notice that the score given to any candidate is of the form $(m-r)+(m/3+r-1)+(m/3-r)$ for some $r \in \{1, \ldots, m/3\}$. As this is at most $5m/3-2$ (when $r=1$), the difference is thus $m/3 = \Omega(m)$. □

4. LINEAR PROGRAMMING

We will begin by providing a "natural" way to formulate the min-max version of the problem as an Integer Program (IP). As solving IPs is NP-hard, we will relax it to the equivalent Linear Program (LP). However, such a natural LP will not be useful in our setting, and we will thus introduce a totally different LP formulation, called *Configuration Linear Programming* (C-LP). The number of variables in the C-LP is exponential in the size of the input. Nevertheless, we show that our C-LP can be solved in polynomial time.

Let $[m] = \{1, \ldots, m\}$ and $[m]_0 = \{0, \ldots, m-1\}$. We define the variables $x_{i,j}$ for $(i,j) \in [m] \times [m]_0$, and the variable T, with the intent that $x_{i,j}$ will equal the number of times candidate c_i received a score of j, and T will serve as the upper-bound on each candidate final aggregate score. The IP can then be stated as follows:

$$\min_{\vec{x}} T$$

subject to:

$$\sum_{i=1}^{m} x_{i,j} = k \qquad \forall j \in [m]_0 \ , \quad (1)$$

$$\sum_{j=0}^{m-1} x_{i,j} = k \qquad \forall i \in [m] \ , \quad (2)$$

$$\sum_{j=0}^{m-1} j \cdot x_{i,j} \leq T - \sigma_i \qquad \forall i \in [m] \ , \quad (3)$$

$$x_{i,j} \in \{0, \ldots, k\} \qquad \forall i \in [m], j \in [m]_0 \ , \quad (4)$$

where (1) guarantees that every score was awarded k times, (2) guarantees that every candidate was given k scores, and (3) guarantees that every candidate gets at most T points.

It should be noted that when treating the problem as a min-max problem, we need to take T as a variable that we wish to minimize (this is done by the objective function). However, if we consider the original definition in which our aim is to make the preferred candidate p win, T can be set to $\sigma_0 + km$ (the final score of the preferred candidate), and the IP will not have an objective function.

While we can relax this IP into an LP by replacing the set in the last constraint to be the continuous interval $[0, k]$, it will not be helpful: for an intuition notice that the system is somewhat under-determined with $\Theta(m)$ equations and $\Theta(m^2)$ variables; it can assign each candidate a similar mixture of score-types, where the only difference will be in the average of this mixture for each candidate. Therefore, a solution would not give us any guidance on how manipulators should vote.

In order to alleviate this we will have to resort to a totally different approach, in which variables no longer represent score types, and instead represent the set of scores (*configuration*) that can be awarded to a candidate.

Formally, a *configuration* C for some candidate c_i is a vector of dimension m in which C_j represents a number of scores of type j that i has received, and for which $\sum_{j=0}^{m-1} C_j = k$, that is, the overall number of scores awarded is k. For a candidate c_i and a bound T, let $\mathcal{C}_i(T)$ be the set of configurations that do not cause the candidate overall score to surpass T, i.e., the set of configurations C for which $\sum_{j=0}^{m-1} C_j \cdot j \leq T - \sigma_i$.

We formulate the configuration LP is as follows:

$$\sum_{C \in \mathcal{C}_i(T)} x_{i,C} \leq 1 \qquad \forall i \in [m] \ , \quad (5)$$

$$\sum_{\substack{i,C \\ C \in \mathcal{C}_i(T)}} C_j x_{i,C} \geq k \qquad \forall j \in [m]_0 \ , \quad (6)$$

$$x_{i,C} \geq 0 \qquad \forall i \in [m], C \in \mathcal{C}_i(T) \ . \quad (7)$$

where we wish that the $x_{i,C}$'s would serve as indicator variables indicating whether or not c_i was awarded with configuration C, (5) guarantees that every candidate was given at most 1 configuration and (6) guarantees that every score was awarded at least k times. The choice of inequalities over equalities will be explained soon.

EXAMPLE 1. *Consider the case where $k=2$, $m=5$ and $(\sigma_1, \ldots, \sigma_5) = (5,6,6,6,7)$. We are omitting the non-manipulator votes that provided $\vec{\sigma}$, however recall that there is a possible non-manipulator voting yielding any $\vec{\sigma}$ up to an additive factor and an addition of a candidate. Now assume $T = 10$ (this is indeed the optimum). Let us focus on the last candidate c_5. $\mathcal{C}_5(T)$ should therefore contain all configurations which award c_5 at most $T - \sigma_5 = 3$ points. Those configurations are $(2,0,0,0,0)$ (0 points), $(1,1,0,0,0)$ (1 point), $(0,2,0,0,0)$, $(1,0,1,0,0)$ (2 points), and $(1,0,0,1,0)$ (3 points).*

When solving the $C-LP$, only two of her configurations will get non-zero value: $x_{5,(1,0,0,1,0)} \approxeq 0.7$ and $x_{5,(0,1,1,0,0)} \approxeq 0.3$. We omit the variables corresponding to the rest of the candidates.

After solving the LP, we will execute a rounding procedure that will transform the fractional LP solution into a valid solution for the original problem. This procedure can increase the score of some of the candidates, and hence we wish to start with the smallest possible T (so that even after the increase the final score will hopefully be bounded by $\sigma_0 + km$).

To find the smallest possible T, we perform a one-sided binary search on the value of T. For this purpose, for each possible value of T that we come across during the binary search, we redefine the LP and then solve the new LP from

scratch, and see if it has a valid solution. The reason we do not add T as a variable in an objective function (instead of the binary search) is that the number of summands in Equations (5,6) depends on T.

This formulation has the obvious drawback that the number of variables is exponential in k. However, following the approach of [2], if we find a polynomially-computable separation oracle we can solve the LP by referring to the LP dual and using the ellipsoid method. Such an oracle will require a solution to the following seemingly unrelated problem as a subroutine: a variant of the classic Knapsack problem.

4.1 k-Multiset Knapsack

Let $\{1,\ldots,m\}$ be a set of distinct items, where each item has an associated value v_j and a weight w_j. We also obtain a weight upper-bound W and a value lower-bound V. As opposed to ordinary knapsack, we also obtain an integer k. We are required to find a multiset S of exactly k items (i.e., we can repeat items from the item-set), such that S's overall weight is at most W and S's overall value is greater than V (or to return that no such multiset exists).

LEMMA 3. *The k-multiset knapsack can be solved in time polynomial in k and W (which is pseudo-polynomial due to the dependence on W).*

PROOF. We fill out a table $Q[w,\ell]$, for $w=0,\ldots,W$ and $\ell=0,\ldots,k$, in which $Q[w,\ell]$ is the highest value obtainable with a size-ℓ multiset of items of aggregate-weight at most w. Notice that Q can be filled using the following recursion:

$$Q[w,\ell]=\begin{cases}0 & \text{if } \ell=0,\\ \max_j v_j + Q'(w-w_j,\ell-1) & \text{otherwise,}\end{cases} \quad (8)$$

where $Q'(w,\ell)=Q[w,\ell]$ if it is defined, i.e., $w\geq 0$ and $\ell\geq 0$, and otherwise is $-\infty$.

Therefore Q can be filled-out using dynamic programming. Finally, the entry $Q[W,k]$ contains the highest value obtainable with overall weight at most W. Therefore, if $Q[W,k]>V$, we have found a required multiset; otherwise such does not exists. The resulting multiset itself can be recovered using backtracking on the table Q. □

4.2 Solving the C-LP

We return to our problem. The choice of inequalities over equalities is motivated by our use of the LP dual in Theorem 5. However, they have the same effect as equalities, as shown by the following lemma:

LEMMA 4. *In a solution to the above C-LP, Equations (5,6) will actually be equalities.*

PROOF. Notice that by Equation (6):

$$km \leq \sum_{j=0}^{m-1}\sum_{i=1}^{m}\sum_{C\in\mathcal{C}_i(T)} C_j x_{i,C} \quad (9)$$

$$= \sum_{i=1}^{m}\sum_{C\in\mathcal{C}_i(T)} x_{i,C}\sum_{j=0}^{m-1} C_j \quad (10)$$

$$= k\sum_{i=1}^{m}\sum_{C\in\mathcal{C}_i(T)} x_{i,C} \quad (11)$$

$$\leq km \quad (12)$$

where (12) holds by plugging (5) into (11). We therefore get that $\sum_{j=0}^{m-1}\sum_{i=1}^{m}\sum_{C\in\mathcal{C}_i(T)} C_j x_{i,C}=km$ which forces both above non-trivial LP inequalities to be equalities. □

THEOREM 5. *Given a fixed value T, the C-LP can be solved in polynomial time.*

PROOF. We need to refer to the LP dual for our C-LP in order to solve it; we briefly repeat some LP duality concepts here, refer to [19] for complete definitions and discussion.

The dual of a maximization problem is a minimization problem. In order to define it we can treat our primal program as a maximization problem having all coefficients 0 in its objective function. In the dual there is a variable for every constraint of the primal, and a constraint for every variable of the primal. Therefore, we define a variable y_i for each candidate c_i and a variable z_j for each score-type j (since the primal has a constraint for each candidate c_i and score-type j). However, since our primal has an exponential number of variables, the dual will have an exponential number of constraints. We will show how to address this.

In short, the non-trivial constraints are then obtained by transposing the constraint-coefficient matrix of the primal, using the primal objective function coefficients as the right-hand side of the dual constraints, and using right-hand side of the primal constraints as the coefficients of the dual objective function.

The process yields the following dual:

$$\min_{\vec{y},\vec{z}} \sum_{i=1}^{m} y_i - k\sum_{j=0}^{m-1} z_j$$

subject to:

$$\sum_{j=0}^{m-1} C_j z_j \leq y_i \qquad \forall i\in[m], C\in\mathcal{C}_i(T)$$

$$y_i \geq 0 \qquad \forall i=1,\ldots,m$$

$$z_j \geq 0 \qquad \forall j=1,\ldots,m$$

As mentioned, the dual has an exponential number of constraints. However it is solvable; the *ellipsoid method* [16] is a method for solving an LP which iteratively tries to find a point inside the feasible region described by the constraints. However, we do not need to provide all the constraints in advance. Instead, the algorithm can be provided with a subroutine, called a *separation oracle*, to which it calls with a proposed point, and the subroutine then either confirms that the point is inside the feasible region or that it returns a violated constraint [12]. The ellipsoid method algorithm performs a polynomial number of iterations, therefore if the separation oracle runs in polynomial time as well, the LP is solved in overall polynomial time. Notice that the polynomial number of iterations performed by the ellipsoid method implies that the number of constraints that played a part in finding the optimum (known as *active* constraints) was polynomial as well. In other words, we could effectively discard all the constraints but a polynomial number of them.

As discussed, a separation oracle for the dual, given a proposed solution $(\vec{y};\vec{z})$, needs to find in polynomial time a violated constraint, if exists. It remains to show that such a separation oracle is polynomial-time computable.

Observe that a violated constraint to this program is a pair i,C for which $C\in\mathcal{C}_i(T)$ (and therefore $\sum_{j=0}^{m-1} C_j\cdot j\leq T-\sigma_i$) and at the same time $\sum_{j=0}^{m-1} C_j z_j > y_i$. Fortunately, for a

specified i, finding a configuration C that induces a violated constraint can be seen as finding a k-multiset (since $\sum_{j=0}^{m-1} C_j = k$) given by a solution to our knapsack variant: $[m]_0$ is the item set (over which j ranges), the value for the item j is z_j, while its weight is j. The given value lower bound is y_i, and $T - \sigma_i$ is the given upper bound on the weight. Effectively, we use the possibly-tighter weight bound $\min\{km, T - \sigma_i\}$ instead, as km bounds the overall weight obtainable with a size-k multiset. As now the weight bound is polynomial in m and k, the solution to our knapsack variant becomes polynomial.

We repeat this knapsack-solving step for each i until we find a violated constraint, or conclude that no constraint is violated. Once we have solved the dual using the ellipsoid method with the separation oracle, we can discard all variables in the primal that do not correspond to violated constraints of the dual, since the inclusion of those constraints (resp. their corresponding variables) did not have any effect on the dual optimum (resp. the primal optimum).[5] The primal now contains only a polynomial number of variables and can be solved directly using the ellipsoid method or any other known polynomial solvers for LP, such as [15]. □

5. ALGORITHM

Solve the above mentioned configuration-LP formulation as described in Section 4. As mentioned, while both constraints are inequalities, in any solution they will actually be equalities. For each candidate c_i, observe the variables $x_{i,C}$, $C \in \mathcal{C}_i(T)$. Since $\sum_{C \in \mathcal{C}_i(T)} x_{i,C} = 1$, treat the $x_{i,C}$'s as a distribution over the configurations for i and randomly choose one according to that distribution. For the time being, give i this configuration.

While every candidate now has a valid configuration (and her score does not exceed T), it is possible that the number of scores of a certain type is above or below k. Formally, if the candidate c_i received a configuration C^i, let the array H such that $H[j] = \sum_{i=1}^{m} C_j^i$ be the *histogram* of the scores. It is then possible that $H[j] \neq k$. If we would translate the configuration given to each candidate to the list of the scores awarded within it, and would write this list as the column of a matrix, this matrix might not be a relaxed manipulation matrix. In order to solve this, we need to replace some of scores in this matrix with others such that the number of scores of each type will be k. On the other hand, we need to make sure this process does not add much to the score of each candidate.

Let $t = (i, j)$ be a tuple representing the event that candidate c_i received a score of j in its configuration. Place all such tuples in a single multiset (if j is awarded to i more than once, repeat (i, j) as needed). Now sort this multiset according to the j value (break ties between candidates arbitrarily) thus creating the event-sequence $t_0, \ldots, t_{km-1}$, i.e., the tuples are now indexed by their rank in this sequence. For each tuple $t_\ell = (i, j)$ having rank ℓ in the list, change its score from j to $\lfloor \ell/k \rfloor$. Notice that now every score is repeated k times. Finally, for each tuple return its score to its candidate, such that every candidate now obtained a new configuration. The process is summarized as Algorithm 1.

[5] In other words, the dual of the dual without the discarded constraints is the primal without their corresponding variables. Another way to explain this is that this is exactly the complementary slackness condition of the Karush-Kuhn-Tucker conditions [17], a necessary condition for obtaining the optimum.

Algorithm 1: Approximation algorithm.

```
Solve the C-LP as described in Section 4
foreach i do define distribution q s.t. q(C) = x_{i,C} for
    all C ∈ C_i(T) and randomly choose C^i ~ q.
L ← ⟨⟩                          /* L is the empty list */
foreach i ∈ [m], j ∈ [m]_0 do
    | Append C_j^i copies of (i, j) to L
Sort L in an ascending order by score(·) /* score(t) = j
    if t = (i, j) */
Re-index L such that L = ⟨t_0, ..., t_{km-1}⟩
for ℓ = 0, ..., km − 1 do
    | Let t_ℓ = (i, j)
    | t_ℓ ← (i, ⌊ℓ/k⌋)
    | Assign the score ⌊ℓ/k⌋ to c_i
return the resulting relaxed manipulation matrix
```

LEMMA 6. *Let $C \in \{C^1, \ldots, C^m\}$ be a configuration obtained for some candidate by the rounding process, and let C' be its corrected version given by the process described above. Then there exists a constant d such that with arbitrary-chosen polynomially-small failure probability,*

$$\sum_{j=0}^{m-1} jC'_j \leq \sum_{j=0}^{m-1} jC_j + d \cdot k\sqrt{m \ln m} \ .$$

.

PROOF. Let H be the histogram of the original configurations $C^1, \ldots, C^m$, and let the array G be the array of histogram partial sums, i.e., $G[j] = \sum_{j'=0}^{j} H[j']$. In a similar manner, define $D^i[j] = \sum_{j'=0}^{j} C^i_{j'}$ to be the partial sums array w.r.t. each candidate c_i. We will show that with high probability, $G[j] \leq (j+1)k + dk\sqrt{m \ln m}$.

Fix a specific j. Notice that

$$\mathbb{E}[G[j]] = \sum_{j'=0}^{j} \mathbb{E}[H[j']] = \sum_{j'=0}^{j} \sum_{\substack{i,C \\ C \in \mathcal{C}_i(T)}} C_{j'} x_{i,C} = (j+1)k$$

according to the LP constraints, and that $G[j] = \sum_{i=1}^{m} D^i[j]$, that is, $G[j]$ is a random variable which is the sum of the independent random variables $D^i[j]$ for $i = 1, \ldots, m$. In addition, for every candidate c_i, it holds that $D^i[j] \in [0, k]$, as a configuration contains at most k scores. Therefore, using the generalized Hoeffding inequality [14, Theorem 2]:

$$\begin{aligned} \Pr\left[G[j] - \mathbb{E}[G[j]] \geq \epsilon m\right] &\leq \exp\left(-\frac{2\epsilon^2 m^2}{\sum_{i=1}^{m} k^2}\right) \\ &= \exp\left(-\frac{2\epsilon^2 m}{k^2}\right) \ . \end{aligned}$$

Setting $\epsilon = d'k\sqrt{\ln m}/\sqrt{m}$, for some arbitrary constant d', we get that $\Pr[G[j] - \mathbb{E}[G[j]] \geq d'k\sqrt{m \ln m}] \leq 1/m^{d'}$, that is, the probability that we deviate from $\mathbb{E}[G[j]]$ by more than $\tilde{O}(k\sqrt{m})$ can be made arbitrarily polynomially small. Using the union bound, the same can be made to hold for all $j = 0, \ldots, m-1$ simultaneously.

Now observe a tuple $t_\ell = (i, j')$ before being possibly corrected by the algorithm. Since its rank ℓ in the sorted sequence is at most the number of scores whose type is at

most j', which is by definition $G[j']$, we get that $\ell \leq G[j'] \leq (j'+1)k + d'k\sqrt{m \ln m}$, where the second inequality holds with high probability. Therefore by the algorithm changing the score j' to $\lfloor \ell/k \rfloor$, the score increases by at most $\lfloor \ell/k \rfloor - j' \leq (j'+1) + d'\sqrt{m \ln m} - j' \leq d'\sqrt{m \ln m} + 1$.

Now observe some candidate c_i with a given configuration C^i corrected to become a configuration C' by the algorithm. Since at worst case, all of c_i's k scores where affected as such, her overall score has increased by at most $O(k\sqrt{m \log m})$. □

COROLLARY 7. *The above algorithm provides an additive $\tilde{O}(k\sqrt{m})$ approximation with high probability. By repeating the randomized rounding procedure a linear number of times, the failure probability becomes exponentially-small. The overall running time is polynomial.*

PROOF. Let T^* be the optimal value for the original problem, and let T_{CLP} be the best bound obtainable via the above CLP combined with the binary search on T. Notice that $T_{\text{CLP}} \leq T^*$, as the optimal solution is also a valid solution for the CLP. Now observe the highest scoring candidate in the CLP. When the algorithm terminates, we get that with high-probability her score is $T_{\text{CLP}} + \tilde{O}(k\sqrt{m}) \leq T^* + \tilde{O}(k\sqrt{m})$. If we repeat the randomized rounding procedure a linear number of times and pick the iteration yielding the minimum addition to T_{CLP}, the probability of not getting a $\tilde{O}(k\sqrt{m})$-approximation becomes exponentially-small.

As the additional score given by the algorithm to any other candidate is also $\tilde{O}(k\sqrt{m})$, the bound $T^* + \tilde{O}(k\sqrt{m})$ holds for all candidates. We conclude that this is indeed an $\tilde{O}(k\sqrt{m})$ additive approximation.

As discussed, solving the C-LP is done in polynomial time (by the polynomial number of iterations of the ellipsoid method and the polynomial runtime of the k-multiset knapsack separation oracle). The rounding is dominated by going over a polynomial number of non-zero variables of the C-LP and is therefore polynomial as well. It is repeated a linear number fo times in order to provide an exponentially-small failure probability. □

5.1 Implementation

We implemented our algorithm and uploaded the code to a public repository[6]. The main subroutine is solving the LP dual we have defined. However, the dependence on an LP solver using the ellipsoid method with a separation oracle proved difficult as to the best of our knowledge there is no library which enables solving an LP this way. Instead we simulated this by using a general LP-solving library [1] and running the separation oracle externally as described in Algorithm 2.

Our practical running time is dominated by solving the C-LP. Therefore, we can perform the randomized part of rounding process several times, and pick the best one, while incurring only a negligible overhead to the runtime.

6. EXPERIMENTS

We have run experiments on sets of values for n, k and m. As mentioned, our algorithm is theoretically competitive when $k = o(\sqrt{m/\log m})$. We have wished to verify this also in an empirical setting, and we have thus chosen $k \approx \sqrt{m}$ or smaller, as these are the values for which our algorithm is known to be theoretically superior to REVERSE. Lower k values are also the cases which are more difficult to AVERAGE FIT heuristic of [8]. We have chosen $n = 2k$; for each n, k, m combination, we have run 8 experiments in which we have drawn the non-manipulator votes from a uniform distribution. We have compared our results to those obtained by AVERAGE FIT, that was shown empirically to outperform REVERSE [8], and to the fractional solution, T_{CLP}. The results are summarized in Figure 1.

[6] `https://github.com/okeller/BordaManipulation`

Algorithm 2: Simulating the ellipsoid method with a separation oracle.

Input: A linear program $P = (f, S)$ with an objective function f and a separation oracle S (instead of an explicit list of constraits)

```
1 Let R ← ∅          /* R will be a list of effective
  constraints */
2 Let P' ← (f, R)    /* P' is P without any
  constraints */
3 x⃗ ← LP-SOLVE(P')
4 while S(x⃗) returns a violated constraint r do
5 |   R ← R ∪ {r}
6 |   P' ← (f, R)
7 |   x⃗ ← LP-SOLVE(P')
8 return x⃗
```

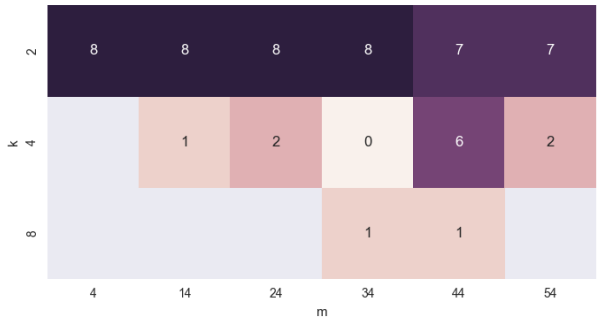
As can be seen, our algorithm performs well in practice. In the vast majority of the cases, both the C-LP and AVERAGE FIT were known to be identical to the optimal solution, as depicted in Figure 1a: these are the instances in which the C-LP solution did not increase when performing the LP rounding. Therefore, in those cases we know what is the real optimal solution, as T_{CLP} (resp. our final algorithm result) provides a lower (resp. an upper) bound for it, and both are equal. Thus our formulation in many times provides a very efficient method for verifying whether our algorithm (or any other algorithm) finds an optimal solution.

In all other cases, i.e., where the C-LP solution increased when performing the LP rounding, C-LP either beats or equals AVERAGE FIT, as depicted in Figure 1b. For instance, consider our example from before, this time with p: $k = 2$, $m = 5$ and $(\sigma_0, \ldots, \sigma_5) = (0, 5, 6, 6, 6, 7)$. Obviously both methods will award p with $5 + 5 = 10$. However in ours the top score of a candidate who is not p will be 10, and in AVERAGE FIT it is 12. Therefore, the choice of algorithm determines if p wins or not by a difference of 2.

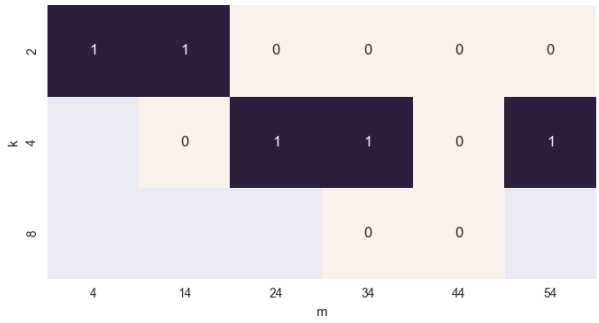
7. CONCLUSIONS

We have presented an $\tilde{O}(k\sqrt{m})$-additive approximation to the score of the highest-scoring non-preferred candidate. It enables us to find a winning strategy for p in cases where other methods would not necessarily have found one. Our method employs new techniques based on configuration linear programs. There are several interesting directions for future work, which vary from the concrete to the general:

1. Understand in which instances different manipulation strategies outperform others.
2. Find algorithms that can guarantee victory even if the margin is smaller, or prove NP-hardness even if there is a solution with margin $\sqrt{m}$.



(a) Number of cases for various value of m and k (out of 8 experiments carried out for each m, k), where our algorithm provided results equal to the fractional solution T_{CLP} (and therefore, to the optimum T^*).



(b) Number of cases for various value of m and k (out of 8 experiments carried out for each m, k), where the C-LP provided strictly better results compared to Average Fit.

Figure 1: Experimental results.

3. Apply a C-LP to other voting methods. The application to some scoring rules seems like an easy first step.

4. Apply a C-LP to other problems in computational social choice, such as fair division of multiple indivisible goods.

Acknowledgments

We thank Sarit Kraus and Ariel Procaccia for insightful discussions.

This work was supported by the Israel Science Foundation, under Grant No. 1488/14 and Grant No. 1394/16.

REFERENCES

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. CVXOPT: A Python package for convex optimization, version 1.1.9. cvxopt.org, 2016.

[2] N. Bansal and M. Sviridenko. The Santa Claus problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 31–40, 2006.

[3] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6(3):227–241, 1989.

[4] N. Betzler, R. Niedermeier, and G. J. Woeginger. Unweighted coalitional manipulation under the Borda rule is NP-hard. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 55–60, 2011.

[5] E. Brelsford, P. Faliszewski, E. Hemaspaandra, H. Schnoor, and I. Schnoor. Approximability of manipulating elections. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 44–49, 2008.

[6] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):14, 2007.

[7] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. Complexity of and algorithms for Borda manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, 2011.

[8] J. Davies, G. Katsirelos, N. Narodytska, T. Walsh, and L. Xia. Complexity of and algorithms for the manipulation of Borda, Nanson's and Baldwin's voting rules. *Artificial Intelligence*, 217:20–42, 2014.

[9] E. Ephrati and J. S. Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 423–431, 1993.

[10] P. Faliszewski and A. D. Procaccia. Ai's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[11] A. Gibbard. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society*, pages 587–601, 1973.

[12] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[13] E. Hemaspaandra and L. A. Hemaspaandra. Dichotomy for voting systems. *J. Comput. Syst. Sci.*, 73(1):73–83, 2007.

[14] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[15] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.

[16] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

[17] H. Kuhn and A. Tucker. Nonlinear programming. In *Proceedings of 2nd Berkeley Symposium. Berkeley: University of California Press*, pages 481–492, 1951.

[18] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of economic theory*, 10(2):187–217, 1975.

[19] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[20] S. Sina, N. Hazon, A. Hassidim, and S. Kraus. Adapting the social network to affect elections. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 705–713. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[21] O. Svensson. Santa Claus schedules jobs on unrelated machines. *SIAM J. Comput.*, 41(5):1318–1341, 2012.

[22] L. Xia, M. Zuckerman, A. D. Procaccia, V. Conitzer, and J. S. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 348–353, 2009.

[23] M. Zuckerman, A. D. Procaccia, and J. S. Rosenschein. Algorithms for the coalitional manipulation problem. *Artificial Intelligence*, 173(2):392–412, 2009.