Approximating Bribery in Scoring Rules

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

The classic bribery problem is to find a minimal subset of voters who need to change their vote to make some preferred candidate win. We find an approximate solution for this problem for a broad family of scoring rules (which includes Borda and t-approval), in the following sense: if there is a strategy which requires bribing k voters, we efficiently find a strategy which requires bribing at most $k + \widetilde{O}(\sqrt{k})$ voters.

Our algorithm is based on a randomized reduction from bribery to coalitional manipulation (UCM). To solve the UCM problem, we apply the Birkhoff-von Neumann (BvN) decomposition to a fractional manipulation matrix. This allows us to limit the size of the possible ballot search space reducing it from exponential to polynomial, while still obtaining good approximation guarantees. Finding the optimal solution in the truncated search space yields a new algorithm for UCM, which is of independent interest.

Introduction

We study the popular preferential model for elections, where each of the agents (also: *voters*) *ranks* the candidates. Then, some function—known as a *voting rule* or a protocol—is applied on the voter rankings in order to decide on the winner(s). Ideally, we would like the voters to be truthful: that is, that the rankings submitted by each voter would correspond to his real preference over the alternatives. When the voters might have the incentive to do otherwise, we refer to the voting protocol as *manipulable*. Unfortunately, a celebrated result in social choice theory achieved independently by Gibbard and Satterthwaite (**?**; **?**) shows that when the number of candidates is at least 3, any reasonable voting rule is manipulable.

Manipulation can take several different forms. In *unweighted coalitional manipulation* (UCM), a set of voters (hereafter the *manipulators*) sharing a preference for a candidate p try to coordinate their voting so that p will win the election. This is assuming that all other voters are truthful and that their preferences are known. In *bribery*, an interested party is willing to pay some of the voters to change their vote into a given ballot, such that p will prevail. In both cases, the goal is to make p win using minimal resources, for example, the number of manipulators or bribed voters, respectively.

The importance of the aforementioned manipulation forms ranges beyond their immediate definition. First, various manipulation forms model various aspects of *campaign management*: which electorate or demographic should a candidate target during his campaign, and where should his or her campaign manager direct the campaign funds. In recent years, the significance of targeting specific electorates in election campaigns had skyrocketed, see (**?**) for one example out of many. In this context, bribery models the act of targeting prospective voters and convincing them to change their vote, and UCM models the act of urging people not planning on voting to exercise their right to vote (when it is done by an interested third party supporting one candidate).

Second, they can also be seen as means by which we measure *how well* did a candidate do in the election, or equivalently how far away was her from winning it. They achieve so by calculating the effort needed to promote her enough in order to win (**?**). In some way, such measures are more robust compared to intrinsic measures like the candidate score in the election.

In response to the Gibbard-Satterthwaite theorem, extensive research on the computational aspects of social choice focused on providing some hope against manipulation by showing that in many scenarios, finding a manipulative strategy is computationally hard. In such cases research naturally shifts its focus to devising approximations and heuristics.

In this paper, we shall focus on one of the most important families of voting rules, known as *positional scoring rules*, or scoring rules for short. For a given *score vector* $\boldsymbol{\alpha} = (\alpha_0, \ldots, \alpha_m)$, where $\alpha_0 \leq \cdots \leq \alpha_m$ and $m + 1$ is the number of candidates, a scoring rule $\mathcal{R}_{\boldsymbol{\alpha}}$ is a voting rule where each voter awards $\alpha_m, \ldots, \alpha_0$ points to the candidates he ranked in places $1, 2, \ldots, m+1$, respectively. The winners are the candidates with maximum aggregate score. Popular cases of scoring rules are the Plurality, Veto, and Borda voting rules.

A string of results researched both the hardness and approximability of UCM for various voting rules, and in particular scoring rules. However, is seems that the equivalent landscape for bribery is lacking: while several results had shown that Borda-Bribery is NP-hard, and that the same holds even for the relatively simple t-approval-bribery (**?**;

?; **?**; **?**; **?**), only little work was done on approximating bribery (**?**; **?**; **?**). We aim at filling this gap, by providing results which target the most classic bribery model, where the goal is to minimize the number of bribed voters. Specifically we show that for many scoring rules:

If there exists a strategy making a preferred candidate p win by bribing k voters, then we can efficiently *find a strategy making her win while bribing additional $\widetilde{O}(\sqrt{k})$ voters.*[1]

Our additive approximation can be seen as a $(1 + o(1))$-multiplicative approximation. It should be noted that the non-trivial feat was to provide a guarantee tighter than any constant-factor approximation. To provide an intuition as to why a constant-factor approximation is relatively easy, let us focus on one relatively simple rule, namely t-approval. If k is the optimal number of manipulators needed, this means that the gap between any candidate and p was at most $2k$, as bribing a voter can decrease the gap between some candidate and p by at most 2. Now imagine the following strategy: iteratively, pick a voter that did not vote for p (if no such voter exists, then p is already winning), and bribe him to transfer one point from any candidate he currently supports, to p. This decreases the aforementioned gap by at least 1, and therefore at most $2k$ bribed voters are required by this procedure. Thus, we have just described a 2-multiplicative approximation to t-approval-bribery.

We provide a non-trivial approximation to $\mathcal{R}_{\boldsymbol{\alpha}}$-Bribery for large families of scoring rules (encompassing well-known ones, like t-approval variants and Borda). Our methods are based on relaxed linear programs that are transformed to a valid solution (i.e., bribery strategy) using a seminal result from the interplay of combinatorics and geometry, namely the *Birkhoff-von Neumann (BvN) decomposition* (**?**; **?**; **?**), and specifically the constructive proof to its related theorem. We use them as a tool to reduce the size of the valid strategy search space from exponential to polynomial. It thus provides an important insight on the underlying combinatorial properties of manipulation under scoring rules, and is interesting in its own right.

En route to providing our approximation for $\mathcal{R}_{\boldsymbol{\alpha}}$-Bribery, we also supply new approximation results for $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM. Our results target both approximation objectives appearing in the literature, namely minimizing the number of required manipulators (see (**?**; **?**)), and minimizing the score margin between the highest non-preferred candidate and p (see (**?**)).

Our Results and Contributions

We focus on two families of scoring rules, namely *constant* and *non-concentrated* scoring rules, defined as follows. A scoring rule $\mathcal{R}_{\boldsymbol{\alpha}}$ is called *constant* if $\alpha_m = O(1)$. A scoring rule is called *non-concentrated* if $\bar{\alpha} \leq (1-\epsilon)\alpha_m$, for some constant $\epsilon > 0$, where $\bar{\alpha} = 1/(m+1)\sum_{j=0}^{m} \alpha_j$ is the average of the values in $\boldsymbol{\alpha}$. Our results can be summarized by the following theorems. For $\mathcal{R}_{\boldsymbol{\alpha}}$-Bribery:

[1] As common in the literature, the $\widetilde{O}$ notation discards poly-logarithmic factors.

Theorem 1. *In $\mathcal{R}_{\boldsymbol{\alpha}}$-Bribery under constant or non-concentrated $\mathcal{R}_{\boldsymbol{\alpha}}$, let k be the minimum number of voters needed to be bribed in order to make p win. Then there exists a polynomial-time randomized algorithm using at most $k + \widetilde{O}(\sqrt{k})$-voters, with exponentially-small failure probability.*

This theorem immediately yields the following:

Corollary 2. *In bribery under Borda, t-approval, plurality, or veto, let k be the minimum number of voters to be bribed in order to make p win. Then there exists a polynomial-time randomized algorithm using at most $k+\widetilde{O}(\sqrt{k})$-voters, with exponentially-small failure probability.*

The above theorem shall use the constructive proof of the following algorithm for $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM as a subprocedure:

Theorem 3. *For $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM under any score vector $\boldsymbol{\alpha}$, let k be the number of given manipulators. There exists a polynomial-time randomized algorithm yielding an $\widetilde{O}(\alpha_m\sqrt{k})$-additive-approximation to the margin-minimization objective, with exponentially-small failure probability.*

Moving to the objective of minimizing manipulators, the above theorem also enables the following result which—for the specific case of non-concentrated scoring rules—improves the $m-2$ additive approximation given in (**?**) when $k = o(m)$:

Theorem 4. *For $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM under non-concentrated $\mathcal{R}_{\boldsymbol{\alpha}}$, let k be the minimum number of manipulators required to make p win. Then there exists a polynomial-time randomized algorithm using at most $k+\widetilde{O}(\sqrt{k})$ manipulators, with exponentially-small failure probability.*

Aside for the above results, we provide tighter analysis for Borda-UCM, based on a result in (**?**):

Theorem 5. *The algorithm in (**?**) for Borda-UCM, yields an $(m-1)/2$-additive-approximation to the objective of margin-minimization.*

For brevity, we defer the proof of Theorem 5 to the full version of this paper. It should be noted that while we prove that their algorithm provides tighter analysis for Borda, an equivalent property for *all* scoring rules is not known to hold.

Related Work

Bribery. In the standard bribery model (every voter has a unit price), t-approval-bribery is NP-hard (**?**) except for some small values of t for which t-approval-bribery and t-veto-bribery become easy (**?**). Borda-bribery is NP-hard as well (**?**).

Little work was done on approximating bribery. Of interest is an FPTAS for plurality-weighted-$bribery (**?**) and other work targeting other models like shift-bribery (**?**; **?**; **?**). See survey in (**?**) for more details.

Coalitional Manipulation. When m is bounded, $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM is always easy (**?**). When it is not, it was shown that t-approval (and thus plurality and veto) are still easy (**?**;

?), and then recently, that every scoring rule where $\boldsymbol{\alpha}$ comprises of a constant number of coefficients is as well (**?**). Other cases are not necessarily easy; in particular Borda is NP-hard (**?**; **?**).

Its weighted counterpart $\mathcal{R}_{\boldsymbol{\alpha}}$-WCM (where every voter has an associated weight by which his ballot is multiplied) is always hard for $m \geq 3$, besides the single exception of the plurality scoring rule (**?**; **?**; **?**).

As for approximations, two main objectives for approximation appear in the literature. The first focuses on minimizing the *margin* between the highest non-preferred candidate and p, or some additive function thereof[2] which also attains its minimum when the margin is minimized (and thus is equivalent from the optimization standpoint). It was used in some older results, but also garnered recent interest: Brelsford et al. (**?**) provided approximation for $\mathcal{R}_{\boldsymbol{\alpha}}$-WCM for the limited case the number of candidates is constant. Keller et al. (**?**) have removed the requirement on the number of candidates, and showed a randomized algorithm providing $O(k \max_i |\alpha_{i+\beta} - \alpha_i|)$ additive approximation to the margin, where $\beta = \widetilde{O}(\sqrt{m})$ and k is the number of manipulators.

The second approximation objective revolves around minimizing the number of voters involved in the manipulation. For UCM it is the number of added manipulators; for bribery, it is the number of bribed voters. Zuckerman et al. (**?**) show that for Borda-UCM, their greedy algorithm, called REVERSE elsewhere (**?**), provides an additive $+1$ approximation for this objective. Xia et al. (**?**) provide an $m-2$ additive approximation for $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM.

Preliminaries

An election $E = (C, V)$ is defined by a candidate set $C = \{p, c_1, \ldots, c_m\}$ and a set of n voters V where each voter submits a ranking of the candidates according to its preference. Then, some *decision rule* $\mathcal{R}$ is applied in order to decide on the winner(s); formally $\mathcal{R}(E) \subseteq C$ is the set of winners of the elections. In the specific case of a *positional scoring rule* $\mathcal{R}_{\boldsymbol{\alpha}}$, the rule is described by a vector $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_m)$ for which $\alpha_0 \leq \alpha_1 \leq \cdots \leq \alpha_m$, and α_m is polynomial in m, used as follows: each voter awards α_i to the candidate ranked $(m+1-i)$-th. Finally, the winning candidate is the one with the highest aggregated score. In the specific case of Borda scoring rule, we have that $\boldsymbol{\alpha} = (0, 1, \ldots, m-1, m)$. In t-approval, $\boldsymbol{\alpha} = (\mathbf{0}^{m+1-t}; \mathbf{1}^t)$ where $\mathbf{0}^{t'}$ (resp. $\mathbf{1}^{t'}$) is 0 (resp. 1) concatenated t' times. plurality (resp. veto) is the specific case of 1-approval (resp. m-approval).

We assume the non-unique-winner/co-winner model where p is considered a winner even if she is not the only winner. Extending our work for the unique-winner model is rather straightforward, but omitted for brevity.

Problem Definitions

Bribery. Given an election E under a rule $\mathcal{R}$ and a preferred candidate p, the goal is to bribe the minimum amount of voters, such that p will win, where bribing a voter is the act of replacing its ballot by a ranking to our choosing. The output is thus the identity of the bribed voters along with their new ballots.

Minimum-manipulator-UCM. Given an election E under a rule $\mathcal{R}$ and a preferred candidate p. The goal is to add the least amount of additional voters (the manipulators), and to determine their strategies, such that p will win.

Minimum-margin-UCM. Given an election E under a scoring rule $\mathcal{R}_{\boldsymbol{\alpha}}$, a preferred candidate p, and the number of allowed manipulators k, the goal is to determine the manipulator strategies, such that the margin $\max_{c \in C \setminus \{p\}} s(c) - s(p)$ is minimized, where $s(c')$ is c''s final score. Notice that as the number of manipulators is limited, p will not necessarily win.

As we focus on scoring rules $\mathcal{R}_{\boldsymbol{\alpha}}$, we will define the *scoring profile* $\boldsymbol{\sigma}$ such that $\sigma(c)$ is the initial score of c. Notice that for UCM, having $\boldsymbol{\sigma}$ in the input makes V redundant. Also notice that for minimum-margin-$\mathcal{R}_{\boldsymbol{\alpha}}$-UCM, minimizing the margin boils down to to minimizing $\max_{c \in C \setminus \{p\}} s(c)$, as $s(p)$ is determined in advance (every manipulator will award her the maximum score possible and thus $s(p) = \sigma(p) + k\alpha_m$). Therefore we can effectively discard p when solving the problem, and use $\boldsymbol{\alpha}' = (\alpha_0, \alpha_1, \ldots, \alpha_{m-1})$ instead of $\boldsymbol{\alpha}$ (i.e., α_m is removed).

For ease of notation, let $\mathcal{N} = nm$ denote the natural size of the input and let $[m] = \{1, \ldots, m\}$ and $[m]_0 = \{0, \ldots, m-1\}$. For two parameters a, b, we denote the continuous set $[a-b, a+b]$ as $[a \pm b]$.

Probabilities. Throughout the lemmas in this paper, a constant $\lambda > 1$ will be used, where its value will be determined in the main theorems. Many of the lemmas contain mathematical expressions that are said to hold with probability at least $1 - c\mathcal{N}^{-\lambda+d}$ for some constants c, d. That is, their *failure* probability is arbitrarily-chosen polynomially-small (in m), based on our selection of λ, where 'failure' refers to the event that the discussed expression does not hold, and more generally, to the event that algorithm does not provide the desired approximation guarantee. We would sometimes write "with failure probability at most $c\mathcal{N}^{-\lambda+d}$" when presenting such expressions, and sometimes informally use the term 'with high probability' when the exact probability is clear from context.

As the aforementioned λ is constant and always appears in both the runtime and approximation factors in polynomial form, it would not have an effect on the asymptotic behavior of both.

We use the following corollary of the well-known Chernoff bounds, focusing on their behavior for arbitrarily-chosen polynomially-small error-probabilities:

Lemma 6. *Let $X_1, \ldots, X_n$ be independent random variables where $X_i \in [0, U]$ for all i, λ be some constant, and $\mathcal{N}$ be some large enough value. Let $X = \sum_{i=1}^{n} X_i$. Then*

$$\Pr[X \notin [\mathbb{E}[X] \pm R_1(\lambda, U, \mathbb{E}[X])]] \leq \mathcal{N}^{-\lambda}$$

[2] For example, the score of the highest non-preferred candidate, or the difference in margin with and without the manipulation.

where $R_1(\lambda, U, \mathbb{E}[X]) \leq 6\lambda \max\{\sqrt{U\mathbb{E}[X]}, U\} \ln \mathcal{N} = \widetilde{O}(\sqrt{U\mathbb{E}[X]} + U)$ *is the "approximation error" we allow w.r.t. the expected value* $\mathbb{E}[X]$.

For brevity, we defer the proof to the full version of this paper.

Algorithm For UCM

We begin by presenting an approximation for minimum-margin UCM, which is outlined as Algorithm 1. It will be used as a subprocedure in our Bribery algorithm. Consider the relaxed *linear program* (LP) for minimizing the margin for $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM, described as follows.

Given an instance of UCM, as described by a score profile $\boldsymbol{\sigma} = (\sigma(c_i))_{i\in[m]}$ and the number of manipulators k, we start by defining the variables $x_{i,j}$ for $(i,j) \in [m] \times [m]_0$, and the variable θ, with the intent that $x_{i,j}$ will equal the number of times candidate c_i received a score of type α_j, and that θ will serve as the upper-bound on each candidate's final aggregate score. Notice that since the voters are unweighted (i.e., they are identical; all have the same voting power) we do not care which manipulators awarded her those scores; this is true due to (**?**, Theorem 7) where they show that every score assignment such that each candidate has received k scores, and each *score-type* α_j is repeated exactly k times, can be modified in polynomial time into a valid strategy without affecting each candidate's final score.

The LP is defined as:

$$\min_{\mathbf{x}} \theta \tag{1}$$

subject to:

$$\sum_{i=1}^{m} x_{i,j} = k \qquad \forall j \in [m]_0 \,, \tag{2}$$

$$\sum_{j=0}^{m-1} x_{i,j} = k \qquad \forall i \in [m] \,, \tag{3}$$

$$\sigma(c_i) + \sum_{j=0}^{m-1} \alpha_j x_{i,j} \leq \theta \qquad \forall i \in [m] \,, \tag{4}$$

$$x_{i,j} \in [0,k] \qquad \forall i \in [m], j \in [m]_0 \,, \tag{5}$$

where (2) guarantees that every score was awarded k times, (3) guarantees that every candidate was given k scores, and (4) guarantees that every candidate gets at most θ points. The constraint $x_{i,j} \in [0,k]$ is a relaxation of the constraint $x_{i,j} \in \{0,\ldots,k\}$, however, the latter would have caused the LP to become an *integer program* (IP), and solving such is NP-hard. We denote the relaxed LP as $\mathsf{LP}_{\mathrm{CM}}(\boldsymbol{\alpha}, \boldsymbol{\sigma}, k)$.

Assume we solve the above LP, and let $\mathbf{x}^\star$ be the resulting solution w.r.t. the objective value $\theta^\star$. As an optimum of the LP, $\mathbf{x}^\star$ denotes some allocation of scores such that the margin in minimized, however, as this allocation is fractional, it does not translate into a valid strategy.

Many algorithms solve such scenarios using some form of *randomized rounding* over the fractional variables. This seems quite problematic as the variables are highly interdependent, where their dependency is given by the LP constraints. These constraints should still hold even after the rounding.

To work around this, notice that each possible ballot corresponds to some permutation π (with the meaning that c_i receives a score of $\alpha_{\pi(i)}$), and therefore randomized rounding should be done on the ballot level. However, to do so we have to define some distribution over the ballots (permutations), and there are $m!$ of them. This is where the BvN decomposition comes to our rescue.

Observe the matrix $\mathbf{Y} = [y_{i,j}]$ where $y_{i,j} = x^\star_{i,j}/k$ and notice that it is *doubly-stochastic*, that is $\sum_i Y_{i,j} = \sum_j Y_{i,j} = 1$. Roughly speaking, the Birkhoff-von Neumann theorem states that each doubly-stochastic matrix is a point in the Birkhoff polytope, whose vertices are all the $m!$ permutation matrices. In other words, every doubly-stochastic matrix can be obtained by a convex combination (a weighted sum with coefficients in $[0,1]$ which sum to 1) of all permutation matrices. However, the surprising fact—as shown by various constructive proofs to the theorem—is that such a convex sum can be found in which the number of nonzero coefficients is at most m^2.

Let us now state a suitable variant of the Birkhoff-von Neumann theorem:

Theorem 7 (BvN Theorem)**.** *Let* $\mathbf{Y}$ *be a doubly-stochastic matrix. Then we can decompose* $\mathbf{Y}$ *to a convex combination of at most* m^2 *permutation matrices, that is, we decompose* $\mathbf{Y} = \lambda_1 \mathbf{P}_{\pi_1} + \cdots + \lambda_q \mathbf{P}_{\pi_q}$ *where each* π_t *is a permutation with* $\mathbf{P}_{\pi_t}$ *being its corresponding permutation matrix, each* $\lambda_t \in [0,1]$ *and* $\sum_t \lambda_t = 1$*, and* $q \leq m^2$*. This decomposition can be found in polynomial time.*

For the sake of completeness, we supply a constructive proof of this theorem in the full version of this paper.

The remarkable thing about this form of the BvN decomposition, is that it implies that when choosing ballots for each of the manipulators, we only have to consider at most m^2 ballots—a polynomial number—out of the the $m!$ possible ballots (that is, permutations of order m).

Let $\Pi = \{\pi_1, \ldots, \pi_q\}$ (resp. $\lambda_1, \ldots, \lambda_q$) be the set of permutations (resp. weights) used in the above decomposition, and let $\hat{p}\colon \Pi \to [0,1]$ be a distribution over Π the such that $\hat{p}(\pi_t) = \lambda_t$.

We proceed as follows. For each manipulator ℓ, we randomly draw a random permutation $\pi_\ell \sim \hat{p}$, and assign it to ℓ as his ballot, that is ℓ awards c_i a score of $\alpha_{\pi_\ell(i)}$.

Let $R_2(\lambda, \alpha_{m-1}, k) = 6\lambda\alpha_{m-1}\sqrt{(k+1)}\ln\mathcal{N}$. The following lemma shows that this bounds the additional points received by a candidate by the rounding process:

Lemma 8. *With failure probability at most* $\mathcal{N}^{-\lambda+1}$*, the above algorithm adds at most* $R_2(\lambda, \alpha_{m-1}, k)$ *points to the score each candidate had received by the LP.*

Proof. Let $\tilde{x}_{i,j}$ be the number of α_j scores awarded to c_i by the manipulators. Now focus on some i, and let Q be all points awarded to c_i by the manipulators, according to our algorithm. On one hand, $Q = \sum_{j=0}^{m-1} \alpha_j \tilde{x}_{i,j}$. On the other hand, $Q = \sum_{\ell=1}^{k} \alpha_{\pi_\ell(i)}$, and thus

Algorithm 1: $\mathcal{R}_{\boldsymbol{\alpha}}$-UCM Algorithm.

1 Solve $\mathsf{LP}_{\mathrm{CM}}(\boldsymbol{\alpha}, \boldsymbol{\sigma}, k)$, and let $\mathbf{x}^\star$ be the resulting solution.
2 Apply the BvN decomposition on $\mathbf{Y} = \mathbf{x}^\star/k$, and let $\Pi = \{\pi_1, \ldots, \pi_q\}$ be the resulting permutations with respective weights $\lambda_1, \ldots, \lambda_q$. /* The divisiion in $\mathbf{x}^\star/k$ is element-wise */
3 Define a distribution $\hat{p}\colon \Pi \to [0,1]$ over Π such that $\hat{p}(\pi_t) = \lambda_t$.
4 For each manipulator ℓ, randomly draw a random permutation $\pi_\ell \sim \hat{p}$, and assign it to ℓ as his ballot. /* meaning ℓ awards c_i a score of $\alpha_{\pi(i)}$ */

$$\mathbb{E}[Q] = \sum_{\ell=1}^{k} \mathbb{E}[\alpha_{\pi_\ell(i)}] = k \sum_{\pi \in \Pi} \hat{p}(\pi) \alpha_{\pi(i)} \ ,$$

where the last equality follows by the π_ℓ's being i.i.d. By further splitting the summation in the r.h.s.:

$$\mathbb{E}[Q] = k \sum_{j=0}^{m-1} \alpha_j \sum_{\substack{\pi \in \Pi \\ \pi(i)=j}} \hat{p}(\pi) = k \sum_j \alpha_j y_{i,j} = \sum_{j=0}^{m-1} \alpha_j x^\star_{i,j}$$

In addition, notice that

$$\sum_{j=0}^{m-1} \alpha_j x^\star_{i,j} \leq \alpha_{m-1} \sum_{j=0}^{m-1} x^\star_{i,j} = \alpha_{m-1} k \ . \tag{6}$$

Applying Corollary 6 to $Q = \sum_{\ell=1}^{k} \alpha_{\pi_\ell(i)}$ and recalling that $\alpha_{\pi_\ell(i)} \in [0, \alpha_{m-1}]$, we get that with failure probability at most $\mathcal{N}^{-\lambda}$:

$$\begin{aligned} |Q - \mathbb{E}[Q]| &\leq R_1(\lambda, \alpha_{m-1}, \mathbb{E}[Q]) \\ &\leq 6\lambda \max\{\alpha_{m-1}\sqrt{k}, \alpha_{m-1}\} \ln \mathcal{N} \\ &= 6\lambda \alpha_{m-1} \sqrt{k} \ln \mathcal{N} \\ &\leq R_2(\lambda, \alpha_{m-1}, k) \ . \end{aligned}$$

where the first inequality follows by an application of Corollary 6, the second by Eq. (6), and the first equality by naturally assuming that $k \geq 1$ (otherwise there are no manipulators and the problem instance is degenerate).

In words, when we created valid ballots for each of the voters, the score of each candidate increased by at most $R_2(\lambda, \alpha_{m-1}, k)$ with probability failure probability at most $\mathcal{N}^{-\lambda}$. By applying the union-bound over all $m \leq \mathcal{N}$ candidates, it can be made to hold for all candidates simultaneously with failure probability at most $\mathcal{N}^{-\lambda+1}$. □

Let $\tilde{\theta} = \max_{i \in [m]} \sigma(c_i) + \sum_{j=0}^{m-1} \alpha_j \tilde{x}_{i,j}$ be the objective value obtained by our algorithm. We are now ready to complete the proof of Theorem 3:

Proof of Theorem 3. This is true by the fact that the algorithm adds at most $R_2(\lambda, \alpha_{m-1}, k)$ points to each candidate, and thus $\tilde{\theta} \leq \theta^\star + R_2(\lambda, \alpha_{m-1}, k) \leq \bar{\theta} + R_2(\lambda, \alpha_{m-1}, k)$, where $\bar{\theta}$ is the UCM (integral) optimum. The last inequality holds since the LP is a relaxation of the original IP. The overall running time is polynomial, as it is comprised of solving an LP (**?**), followed by the polynomial-time BvN decomposition. The above algorithm has a polynomially-small failure probability $\mathcal{N}^{-\lambda+1}$. By choosing e.g. $\lambda = 2$ and running it a linear number of times, and choosing the run yielding minimal $\tilde{\theta}$, the failure probability becomes exponentially-small, while the runtime stays polynomial. □

The proof of Theorem 4 continues from here, by showing that for non-concentrated scoring rules, when the margin is at most $R_2(\lambda, \alpha_{m-1}, k)$, then with high probability $O(R_2(\lambda, \alpha_{m-1}, k)/\alpha_{m-1}) = \widetilde{O}(\sqrt{k})$ additional manipulators are needed in order to close the gap. We defer the proof to the full version of the paper, and note that it is similar in nature to the proof of Lemma 14.

Algorithm For Bribery

We can now move to describe the actual bribery LP, denoted. Let y_v be an indicator variable for each voter v indicating whether he should be bribed. As bribing voters boils down to their deletion, followed by re-adding them with a new ballot, we also need to describe how to allocate the points of the new ballots. Therefore we define—as in the UCM case—the variables $x_{i,j}$ with the intent that $x_{i,j}$ will equal the number scores of type j awarded to c_i. The relaxed LP is defined as follows:

$$\min_{\mathbf{x}, \mathbf{y}, \theta, k} k$$

subject to:

$$\sum_{v \in V} y_v = k \tag{7}$$

$$\sigma(c_i) - \sum_{v \in V} \alpha_{j(v,c_i)} y_v + \sum_{j=0}^{m-1} \alpha_j x_{i,j} \leq \theta \quad \forall i \in [m] \ , \tag{8}$$

$$\sigma(p) - \sum_{v \in V} \alpha_{j(v,p)} y_v + \alpha_m k \geq \theta \quad , \tag{9}$$

$$\sum_{i=1}^{m} x_{i,j} = k \qquad \forall j \in [m]_0 \ , \tag{10}$$

$$\sum_{j=0}^{m-1} x_{i,j} = k \qquad \forall i \in [m] \ , \tag{11}$$

$$y_v \in [0,1] \qquad \forall v \ , \tag{12}$$

$$x_{i,j} \in [0,k] \qquad \forall i,j \ , \tag{13}$$

where $\alpha_{j(v,c)}$ is the score currently given by a voter v to a candidate c. Constraints (10,11) are identical to their corresponding constraints in the coalitional manipulation LP. Constraint (7) makes sure that the number of bribed voters will be k, the variable we seek to minimize. (8,9) together make sure that p has a final score greater than or equal to any

Algorithm 2: $\mathcal{R}_{\boldsymbol{\alpha}}$-bribery Algorithm.

Solve the Bribery LP as described, and let $(\mathbf{x}^\star; \mathbf{y}^\star, \theta^\star)$ and $k^\star$ be the resulting solution
foreach v **do**
  $\tilde{y}_v \leftarrow \begin{cases} 1 & \text{with probability } y_v^\star \\ 0 & \text{otherwise} \end{cases}$
Set $f \leftarrow \sum_{v \in V} \tilde{y}_v - \sum_{v \in V} y_v^\star$ and let $\tilde{k} = \sum_{v \in V} \tilde{y}_v$
Define $\hat{\sigma}(c_i) = \sigma(c_i) - \sum_{v \in V} j(v, c_i)\tilde{y}_v$ for every $i \in [m]$
Apply our UCM algorithm on the input $((\alpha_0, \ldots, \alpha_{m-1}), \hat{\boldsymbol{\sigma}}, \tilde{k})$
while $\hat{\sigma}(p) < \max_{c \in C'} \hat{\sigma}(c)$ **do**
  `/* while p loses */`
  Pick at most $\epsilon^{-1}N$ voters and bribe them according to Lemma 13/Lemma 14

other candidate. Notice that the use of $\alpha_m k$ in (9) stems from the fact that the score awarded to p by the k bribed voters is known; each will give her the maximum score available.

Our algorithm is outlined as Algorithm 2, and is comprised of four stages. In the first stage, we shall solve the Bribery LP, as previously defined. In the second, we will choose an initial set of voters to bribe using a simple form of randomized rounding. This choice of voters will reduce the problem to an instance of the UCM problem, in which the number of manipulators is known (as we have already determined them). This is the point—the third stage—where we shall use our UCM algorithm, to determine their strategy. Our main claim here is that by bribing not too many voters, and assigning them ballots, we have reduced this bribery instance to another bribery instance, in which the margin is small—at most $\widetilde{O}(\alpha_m\sqrt{k})$. Then, at the fourth stage, we will show that this margin is relatively easy to close, by using at most $\widetilde{O}(\sqrt{k})$ additional bribed voters.

Stage 1: Solving the Bribery LP. We solve, obtaining the solution $(\mathbf{x}^\star, \mathbf{y}^\star, \theta^\star)$ w.r.t. the optimal objective $k^\star$. While $(\mathbf{x}^\star, \mathbf{y}^\star, \theta^\star)$ is a fractional solution (since it solves a relaxed LP) it will enable us to obtain an *integral* solution without too much compromise on the increase in the number of bribed voters.

Stage 2: Rounding $\mathbf{y}^\star$. We round the vector $\mathbf{y}^\star$ without touching $\mathbf{x}^\star$ for now. This is done as follows:

$$\tilde{y}_v = \begin{cases} 1 & \text{with probability } y_v^\star\ ; \\ 0 & \text{otherwise.} \end{cases}$$

Now let $\tilde{k} = \sum_{v \in V} \tilde{y}_v$.

Lemma 9. *Recall that* $R_2(\lambda, \alpha_m, k) = 6\lambda\alpha_m(k+1)^{1/2}\ln\mathcal{N}$. *Then:*

- *With probability at least* $1 - \mathcal{N}^{-\lambda}$,

$$\tilde{k} \in [k^\star \pm R_2(\lambda, 1, k^\star)]\ ,$$

- *For every* $c \in C \setminus \{p\}$, *with probability at least* $1 - \mathcal{N}^{-\lambda}$,

$$\sigma(c) - \sum_{v \in V} \alpha_{j(v,c)}\tilde{y}_v + \sum_{j=0}^{m-1} \alpha_j x_{i,j}^\star \leq \theta^\star + R_2(\lambda, \alpha_m, k^\star)\ ,$$

- *With probability at least* $1 - \mathcal{N}^{-\lambda}$,

$$\sigma(p) - \sum_{v \in V} \alpha_{j(v,p)}\tilde{y}_v + \alpha_m k^\star \geq \theta^\star - R_2(\lambda, \alpha_m, k^\star)\ .$$

Proof sketch. Full proof is deferred to the full version of this paper. As a sketch, the first part follows by Corollary 6. The second and third by the LP definition, and Corollary 6 with Eqs. (8) and (9), respectively. □

Stage 3: Running UCM. Let $f = \tilde{k} - k^\star$, $\boldsymbol{\alpha}' = (\alpha_0, \ldots, \alpha_{m-1})$, and let $\hat{\sigma}(c_i) = \sigma(c_i) - \sum_{v \in V} \alpha_{j(v,c_i)}\tilde{y}_v$. In words, $\hat{\sigma}(c_i)$ is $\sigma(c_i)$ when it is adjusted for the loss of the voters who were deleted as described by the vector $\tilde{\mathbf{y}}$. Notice that $\hat{\boldsymbol{\sigma}}$ is only defined for the non preferred candidates. We have now reduced the problem to the UCM problem: $\hat{\boldsymbol{\sigma}}$ is the new score profile, $\tilde{k}$ is the number of manipulators we have at our disposal (one for each of the deleted voters), and $\boldsymbol{\alpha}'$ is $\boldsymbol{\alpha}$ without the score α_m which is always awarded to p by the manipulators, and thus is irrelevant to the input. We call our UCM algorithm on $(\boldsymbol{\alpha}', \hat{\boldsymbol{\sigma}}, \tilde{k})$. We do not rely here on its approximation guarantee provided by Theorem 3, but on the stronger claim, hinted by Lemma 8, that the $R_2(\lambda, \alpha_{m-1}, k)$ factor is an additive term not just w.r.t. the optimum, but also w.r.t. the fractional solution of $\mathsf{LP}_{\mathrm{CM}}(\boldsymbol{\alpha}', \hat{\boldsymbol{\sigma}}, \tilde{k})$.

In this spirit, let $\mathcal{L}$ be a shorthand to $\mathsf{LP}_{\mathrm{CM}}(\boldsymbol{\alpha}', \hat{\boldsymbol{\sigma}}, \tilde{k})$, and let $\theta_{\mathcal{L}}$ be the optimal objective value of $\mathcal{L}$. We provide the two following lemmas, where the first will compare $\theta_{\mathcal{L}}$ to the value $\theta^\star$. The second will then directly compare the result of our UCM algorithm to $\theta_{\mathcal{L}}$.

Lemma 10. *With failure probability at most* $2\mathcal{N}^{-\lambda+1}$, *it holds that* $\theta_{\mathcal{L}} \leq \theta^\star + 2R_2(\lambda, \alpha_m, k^\star)$.

Proof. We will manually define a (not necessarily optimal) solution to $\mathcal{L}$ where the objective is at most $\theta^\star + 2R_2(\lambda, \alpha_m, k^\star)$. As the LP solution cannot be worse, the lemma will follow.

Let $x_{i,j}^{\star\star} = (\tilde{k}/k^\star)x_{i,j}^\star$ for all i, j. Notice that now Equations (2,3) hold w.r.t. $\mathbf{x}^{\star\star}$ and $\tilde{k}$, as required. When we also plug $\mathbf{x}^{\star\star}$ into (8) instead of $\mathbf{x}^\star$, the l.h.s. of (8) increases by at most

$$\begin{aligned} \sum_{j=0}^{m-1} \alpha_j x_{i,j}^{\star\star} - \sum_{j=0}^{m-1} \alpha_j x_{i,j}^\star &= (\tilde{k} - k^\star)/k^\star \sum_{j=0}^{m-1} \alpha_j x_{i,j}^\star \\ &\leq f\alpha_m/k^\star \sum_{j=0}^{m-1} x_{i,j}^\star \\ &= \alpha_m f \end{aligned}$$

We conclude that for all i with failure probability at most

$2\mathcal{N}^{-\lambda}$, it holds that:

$$\begin{aligned}
&\hat{\sigma}(c_i) + \sum_{j=0}^{m-1} \alpha_j x_{i,j}^{\star\star} \\
&\quad = \sigma(c_i) - \sum_{v \in V} \alpha_{j(v,c_i)} \tilde{y}_v + \sum_{j=0}^{m-1} \alpha_j x_{i,j}^{\star\star} \\
&\quad \leq \sigma(c_i) - \sum_{v \in V} \alpha_{j(v,c_i)} \tilde{y}_v + \sum_{j=0}^{m-1} \alpha_j x_{i,j}^{\star} + \alpha_m f \\
&\quad \leq \theta^\star + R_2(\lambda, \alpha_{m-1}, k^\star) + \alpha_m f \\
&\quad = \theta^\star + R_2(\lambda, \alpha_{m-1}, k^\star) + \alpha_m R_2(\lambda, 1, k^\star) \\
&\quad \leq \theta^\star + 2R_2(\lambda, \alpha_m, k^\star) \ ,
\end{aligned}$$

where the first equality is by definition, the second follows from the above argument, and the third and fourth follow by Lemma 9 (where each also contributes $\mathcal{N}^{-\lambda}$ to the failure probability). Notice that we have just showed here that Equation (4) holds w.r.t. $\mathbf{x}^{\star\star}$ and $\tilde{k}$, as required.

Since we want the above to hold for all i simultaneously, the failure probability becomes at most $2\mathcal{N}^{-\lambda+1}$ by the union bound. We have just defined a valid solution with objective value at most $\theta^\star + 2R_2(\lambda, \alpha_m, k^\star)$ with high probability; as $\theta_{\mathcal{L}}$ cannot be worse, we are done. □

Let $\tilde{\theta}$ be the maximum candidate score as a result of our UCM algorithm on the input $(\boldsymbol{\alpha}', \hat{\boldsymbol{\sigma}}, \tilde{k})$. Define:

$$\begin{aligned}
R_3 = 2R_2(\lambda, \alpha_m, k^\star) \\
+ R_2(\lambda, \alpha_{m-1}, k^\star + R_2(\lambda, 1, k^\star)) \ .
\end{aligned}$$

We claim the following:

Lemma 11. *With probability at least $1 - 4\mathcal{N}^{-\lambda+1}$, it holds that $\tilde{\theta} \leq \theta^\star + R_3$.*

Proof. By combining Lemma 8 w.r.t. $\tilde{\theta}$ and $\theta_{\mathcal{L}}$ and Lemma 10, we obtain that, with failure probability at most $4\mathcal{N}^{-\lambda+1}$:

$$\begin{aligned}
\tilde{\theta} &\leq \theta_{\mathcal{L}} + R_2(\lambda, \alpha_{m-1}, \tilde{k}) \\
&\leq \theta_{\mathcal{L}} + R_2(\lambda, \alpha_{m-1}, k^\star + R_2(\lambda, 1, k^\star)) \\
&\leq \theta^\star + 2R_2(\lambda, \alpha_m, k^\star) \\
&\quad + R_2(\lambda, \alpha_{m-1}, k^\star + R_2(\lambda, 1, k^\star)) \\
&= \theta^\star + R_3 \ ,
\end{aligned}$$

where the first inequality is by Lemma 8, the second by Lemma 9, and the third by Lemma 10. The aforementioned failure probability is obtained by a union-bound over the failure probability of each of the lemmas used. □

Stage 4: Bribing More Voters. Let us return to, and let $\tilde{\mathbf{x}}$ be the allocation of the scores to the candidates according to our UCM algorithm, w.r.t. the optimum value $\tilde{\theta}$. Does plugging $(\tilde{\mathbf{x}}; \tilde{\mathbf{y}}; \tilde{\theta}, \tilde{k})$ into their respective places in the LP constitute a valid solution? The answer is unfortunately no; while we showed that all other constraints hold, Eq. (9) does not necessarily hold. $\sigma(p) - \sum_{v \in V} j(v,p)\tilde{y}_v + \alpha_m \tilde{k}$ might be less then $\tilde{\theta}$. However, we can prove that the margin needed for Eq. (9) to hold is not too large; let $R_4 = R_2(\lambda, \alpha_m, k^\star) + \alpha_m R_2(\lambda, 1, k^\star) + R_3$. Then:

Lemma 12. *With probability at least $1 - 6\mathcal{N}^{-\lambda+1}$, it holds that $\sigma(p) - \sum_{v \in V} \alpha_{j(v,p)} \tilde{y}_v + \alpha_m \tilde{k} \geq \tilde{\theta} - R_4$.*

Proof. Let $Q = \sigma(p) - \sum_{v \in V} \alpha_{j(v,p)} \tilde{y}_v + \alpha_m \tilde{k}$. Then assuming none of the previous lemmas fail:

$$\begin{aligned}
Q &\geq \sigma(p) - \sum_{v \in V} \alpha_{j(v,p)} \tilde{y}_v + \alpha_m k^\star - \alpha_m R_2(\lambda, 1, k^\star) \\
&\geq \theta^\star - (R_2(\lambda, \alpha_m, k^\star) + \alpha_m R_2(\lambda, 1, k^\star)) \\
&\geq \tilde{\theta} - (R_2(\lambda, \alpha_m, k^\star) \\
&\qquad + \alpha_m R_2(\lambda, 1, k^\star) + R_3) \\
&= \tilde{\theta} - R_4 \ ,
\end{aligned}$$

where the first inequality follows by the first part of Lemma 9, the second follows by the third part of Lemma 9, the third by Lemma 11. The failure probability is bounded by $6\mathcal{N}^{-\lambda+1}$, by a union-bound on the failure probabilities of the lemmas used. □

Summing up, Lemma 12 shows that currently p might be still losing by a margin of at most $R_4 = \widetilde{O}(\alpha_m \sqrt{k^\star})$.

The next two lemmas will show the relation between this margin, and the number of manipulators needed in order to close it and make p win. The proofs will be constructive and will supply a polynomial-time algorithm. Let $C' = C \setminus \{p\}$ and let $N = \ln^{1+\delta} \mathcal{N}$ for some constant $\delta > 0$. Let $s'(c)$ be the current score of c at some point in time and let $g = \max_{c \in C'} s'(c) - s'(p)$ be the margin at this point in time.

Lemma 13. *Let $\mathcal{R}_{\boldsymbol{\alpha}}$ be a constant voting rule. By bribing at most N voters we can change the margin to be $g' \leq \max\{0, g - \epsilon\alpha_m N\}$ for some constant $\epsilon > 0$. That is, we can either make p win, or at least reduce the margin by which she loses by $\epsilon\alpha_m N$.*

Proof. By repeating the following procedure at most N times: pick a voter who gave p at most $\alpha_m - 1$ points (if no such voter exists, p is already winning and we can stop repeating the procedure), and change his ballot such that p and the candidate ranked first are swapped. Notice that by this the gap between any c and p has decreased by at least 1.

If the above procedure stopped short of N iterations, then p is winning and we are done. Otherwise the gap is decreased by at least N. Setting $\epsilon = 1/\alpha_m$ we are done. □

Non-concentrated rules have a much more involved proof:

Lemma 14. *Let $\boldsymbol{\alpha}$ be such that $\bar{\alpha} \leq (1 - 6\epsilon)\alpha_m$ for some constant $0 < \epsilon < 1/6$. Then by bribing at most $\epsilon^{-1}N$ voters we can—with failure probability at most $2\mathcal{N}^{-\lambda+1}$—change the margin to be $g' \leq \max\{0, g - \epsilon\alpha_m N\}$. That is, we can either make p win, or at least reduce the margin by which she loses by $\epsilon\alpha_m N$.*

Proof. We split to cases; we will first discuss two easy ones:

- Let A be the set of un-bribed voters who awarded p at most $\alpha_m - 1$ points. If $|A| \leq \epsilon^{-1}N$, simply bribe all voters in A and let them move p to the top position. p is now ranked top by all voters and received the maximum score obtainable and by the co-winner assumption she wins.
- Otherwise, let $B \subseteq A$ be the set un-bribed voters which gave p at most $(1-\epsilon)\alpha_m$ points. If $|B| \geq N$ we are done: we can just bribe N of them and let them move p to the top position, thus decreasing g by $\epsilon\alpha_m N$ points.

If the two above cases do not hold, then it holds that $|A| > \epsilon^{-1}N$, but $|B| < N$. In words, there are at most $|B| < N$ voters who gave p at most $(1-\epsilon)\alpha_m$ points, and $|A \setminus B| > 5N$ voters who gave p more than $(1-\epsilon)\alpha_m$ points. For the time being, we bribe the voter-set B using the method described in the latter item; we will shortly bribe another N voters as well.

Assume we have bribed B; then by now all voters have given p more than $(1-\epsilon)\alpha_m$ points. In other words, p's current score is at least $(1-\epsilon)\alpha_m n$. Now randomly pick N voters from $A \setminus B$. Let them all put p in the top position, and rank all other candidates randomly, that is, the ranking of all other candidates will be determined by a random permutation. Now let c be some candidate and define r such that $s'(c) = r\alpha_m n$. In words, c has received $r\alpha_m$ points from each voter *on average*. Now assume for a moment we first *delete* the N voters we bribe, and only then *re-add* the voters with their new ballots.

When we delete N voters, c loses $r\alpha_m N$ points in expectation. Formally, let X_c be the number of points c had actually lost. Then $\mathbb{E}[X_c] = r\alpha_m N$. We want to make sure that c will lose *approximately* $r\alpha_m N$ points. However—as it is many times the case—we are afraid that X_c will diverge too much from $\mathbb{E}[X_c]$. To analyze that, note that we can treat X_c as a sum of independent random variables $X_{v,c}$, where

$$X_{v,c} = \begin{cases} \alpha_{j(v,c)} & \text{if } v \text{ is chosen to be bribed;} \\ 0 & \text{otherwise.} \end{cases}$$

By Corollary 6, we get that $X_c \in [r\alpha_m N \pm R_1(\lambda, \alpha_m, r\alpha_m N)] \subseteq [r\alpha_m N \pm R_1(\lambda, \alpha_m, \alpha_m N)]$ with failure probability at most $\mathcal{N}^{-\lambda}$.

When we re-add the bribed voters according to our scheme, c receives a score in $[\bar{\alpha}N \pm R_1(\lambda, \alpha_m, \alpha_m N)]$ points with failure probability at most $\mathcal{N}^{-\lambda}$—again, by a similar application of Corollary 6.

Summing up, after the entire bribery process, c had lost at least $(r\alpha_m - \bar{\alpha})N - 2R_1(\lambda, \alpha_m, \alpha_m N)$ points with failure probability at most $2\mathcal{N}^{-\lambda}$. Using the union-bound, the same can be made to hold for all m candidates simultaneously with failure probability at most $2\mathcal{N}^{-\lambda+1}$.

We can now split to cases; candidates with $r \geq 1-4\epsilon$ lost at least $\epsilon\alpha_m N$ points, assuming that $2R_1(\lambda, \alpha_m, \alpha_m N) \leq \epsilon\alpha_m N$ (as it is asymptotically; otherwise the entire input is constant-sized). Candidates with $r < 1-4\epsilon$ might have gained points in the process, however the number of points gained in the process is bounded by the number of points awarded in the voter re-addition stage. Since the number of these awarded points is at most $\bar{\alpha}N + R_1(\lambda, \alpha_m, \alpha_m N)$, each such candidate c now have score of at most $s''(c) \leq (1-4\epsilon)\alpha_m n + \bar{\alpha}N + R_1(\lambda, \alpha_m, \alpha_m N)$. However, since $N \leq \epsilon n$ (follows by the fact that $n \geq |A| > \epsilon^{-1}N$), and $R_1(\lambda, \alpha_m, \alpha_m N) \leq \epsilon\alpha_m N < \epsilon\alpha_m n$,

$$\begin{aligned} s''(c) &\leq (1-4\epsilon)\alpha_m n + \bar{\alpha}N + R_1(\lambda, \alpha_m, \alpha_m N) \\ &\leq (1-4\epsilon)\alpha_m n + \bar{\alpha}\epsilon n + \epsilon\alpha_m n \\ &< (1-4\epsilon)\alpha_m n + \alpha_m \epsilon n + \epsilon\alpha_m n \\ &= (1-2\epsilon)\alpha_m n \leq s'(p) - \epsilon\alpha_m n \,. \end{aligned}$$

We conclude that after this process, every candidate either lost $\epsilon\alpha_m N$ points, or gained points, but in that case never surpassed $s'(p) - \epsilon\alpha_m n \leq s''(p) - \epsilon\alpha_m n$. The amount of voters we have bribed is $|B| + N \leq 2N < \epsilon^{-1}N$. The lemma thus follows. □

With Lemmas 13 and 14, we have just shown that for many types of $\boldsymbol{\alpha}$, the ratio between a margin to the number of bribed voters needed in order to close the margin is $O(\alpha_m)$. This leads to the following:

Lemma 15. *Assuming that Lemma 12 did not fail, then besides the $\tilde{k} = k^\star + f$ voters we have already bribed, with failure probability at most $\lceil R_4/(\epsilon\alpha_m \ln^{1+\delta}\mathcal{N})\rceil \cdot 2\mathcal{N}^{-\lambda+1}$, it holds that at most $f' = \epsilon^{-2}R_4/\alpha_m + \epsilon^{-1}\ln^{1+\delta}\mathcal{N}$ additional voters are needed to be bribed in order for p to win, for some constant $\epsilon > 0$.*

Proof. By repeatedly applying the algorithm in the constructive proof of either Lemma 13 or Lemma 14, until p wins. For constant scoring rules the analysis is straightforward. For non-concentrated scoring rules, since every batch of $\epsilon^{-1}N = \epsilon^{-1}\ln^{1+\delta}\mathcal{N}$ bribed voters decrease the margin by at least $\epsilon\alpha_m N$ points, at most $f' = \lceil R_4/(\epsilon\alpha_m N)\rceil \cdot \epsilon^{-1}N$ bribed voters are needed.

As for the failure probability, we can be conservative and require that each of the $\lceil R_4/(\epsilon\alpha_m N)\rceil$ iterations will succeed; using the union-bound, the probability any of the iterations will fail is at most $\lceil R_4/(\epsilon\alpha_m N)\rceil \cdot 2\mathcal{N}^{-\lambda+1}$. □

We are now ready to complete the proof for Theorem 1.

Proof of Theorem 1. Let $\bar{k}$ be the number of voters bribed by an optimal strategy, and notice that $k^\star \leq \bar{k}$, since the LP is a relaxation of the original problem. Following the above discussion, we had bribed overall $k^\star + f + f' \leq \bar{k} + f + f'$ voters. For the sake of brevity, and since our concern is order of magnitude analysis, we will only loosely bound both the approximation factor $f + f'$ and the failure probability.

Since R_4 can be loosely bounded by $41\lambda^2\alpha_m(\bar{k}+1)^{1/2}\ln^2\mathcal{N}$, then $f + f'$ is bounded by $43\lambda^2\epsilon^{-2}(\bar{k}+1)^{1/2}\ln^2\mathcal{N} = \widetilde{O}(\sqrt{\bar{k}})$. As for the failure probability, we require both Lemmas 12 and 15 to succeed; the probability any of them would fail is at most $6\mathcal{N}^{-\lambda+1} + \lceil R_4/(\epsilon\alpha_m N)\rceil \cdot 2\mathcal{N}^{-\lambda+1} \leq (48\lambda^2\epsilon^{-1}\ln\mathcal{N}) \cdot (\bar{k}+1)^{1/2}2\mathcal{N}^{-\lambda+1} = \widetilde{O}(\bar{k})/\mathcal{N}^{\lambda-1}$. Setting $\lambda = 3$ will thus provide at most $1/\Omega(\mathcal{N})$ failure probability, since $\bar{k} \leq n$.

By running the algorithm a linear number of times, and choosing the run yielding minimal number of bribed voters, the failure probability becomes exponentially-small, while the runtime stays polynomial. □

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