How Hard is Bribery in Elections with Randomly Selected Voters

Liangde Tao
Zhejiang University
Hangzhou, China
tld@zju.edu.cn

Lin Chen
Texas Tech University
Lubbock, Texas, United States
chenlin198662@gmail.com

Lei Xu
University of Texas Rio Grande Valley
Edinburg, Texas, United States
xuleimath@gmail.com

Weidong Shi
University of Houston
Houston, Texas, United States
larryshi@ymail.com

Ahmed Sunny
Texas Tech University
Lubbock, Texas, United States
ahmed.sunny@ttu.edu

Md Mahabub Uz Zaman
Texas Tech University
Lubbock, Texas, United States
m.zaman@ttu.edu

ABSTRACT

Many research works in computational social choice assume a fixed set of voters in an election, and study the resistance of different voting rules against electoral manipulation. In recent years, however, a new technique known as *random sample voting* has been adopted in many multi-agent systems. One of the most prominent examples is blockchain. Many proof-of-stake based blockchain systems like Algorand will randomly select a subset of participants of the system to form a committee, and only the committee members will be involved in the decision of some important system parameters. This can be viewed as running an election where the voter committee (i.e., the voters whose votes will be counted) is randomly selected. It is generally expected that the introduction of such randomness should make the election more resistant to electoral manipulation, despite the lack of theoretical analysis. In this paper, we present a systematic study on the resistance of an election with a randomly selected voter committee against bribery. Since the committee is randomly generated, by bribing any fixed subset of voters, the designated candidate may or may not win. Consequently, we consider the problem of finding a feasible solution that maximizes the winning probability of the designated candidate. We show that for most voting rules, this problem becomes extremely difficult for the briber as even finding any non-trivial solution with non-zero objective value becomes NP-hard. However, for plurality and veto, there exists a polynomial time approximation scheme which computes a near-optimal solution efficiently. The algorithm builds upon a novel integer programming formulation together with techniques from n-fold integer programming, which may be of a separate interest.

KEYWORDS

Voting; Computational Social Choice; Approximation Algorithms

ACM Reference Format:

Liangde Tao, Lin Chen, Lei Xu, Weidong Shi, Ahmed Sunny, and Md Mahabub Uz Zaman. 2022. How Hard is Bribery in Elections with Randomly Selected Voters. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Auckland, New Zealand, May 9–13, 2022*, IFAAMAS, 11 pages.

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Auckland, New Zealand.

1 INTRODUCTION

We study the computational resistance/vulnerability of random sample voting schemes for elections under bribery attacks. Our problem is motivated by the extensive research in computational social choice that studies the computational resistance/vulnerability of various voting rules in a deterministic setting with a fixed set of voters (see, e.g. [7] for a comprehensive survey), as well as the growing popularity of the adoption of random sample voting schemes in multi-agent systems. Briefly speaking, a random sample voting scheme will poll a small number of randomly selected voters into a *committee*, and the election is eventually conducted within the committee. That is, the set of voters (who really votes) is no longer deterministic but rather a random subset.

Remarkably, random sample voting schemes have already been widely implemented in many blockchain systems, including Algorand [26], Bitshares [37], Ethereum [10], etc. For example, shareholders of the Bitshares system randomly select a committee (called BitShares Committee) to control over blockchain parameters such as block size, block time, witness pay, etc. One of the key techniques used by Algorand [26] is Byzantine agreement protocol that uses verifiable random functions which randomly select users in a private and non-interactive way. Another example is Ethereum [10]. Ethereum 2.0 adopts a sharding mechanism where validators can be uniformly selected to form a shard/committee.

Consequently, we are interested in figuring out whether the random sample voting schemes can improve the resilience of a voting system. Towards this, we consider the following problem where we assume that there exists a set of voters, among whom a committee will be formed to run an election. Every voter is selected independently with a probability of p into the committee. There is a briber/attacker who aims at making a designated candidate win the election by bribing a subset of voters within a given budget, however, the briber cannot control whether a bribed voter is selected into the committee or not. Consequently, depending on the randomly generated committee, the briber may or may not succeed in manipulating the electoral result. The goal of the briber is to bribe a subset of voters such that it maximizes the probability of winning for the designated candidate.

It is worth mentioning that there are two common ways of generating a random committee. The first method is to select each voter independently with a uniform probability of p into the committee, which has been used in, e.g, Algorand [26]. In this case, the size of the committee (i.e., the total number of voters in the committee) is not fixed, rather a random variable (though with an extremely high probability that it lies within a small region). The second method is to select a committee of a fixed size uniformly at random. In this paper, we restrict our attention explicitly to the first method, as such a random scheme guarantees some important features in cryptography [26]. Nevertheless, due to the close relationship between the two methods, our technique may be extended to handle the second method, or even a broader class of randomness.

Remarkably there are only a few number of prior researches involving a random voter set (see Section 1.2). Among them, our model is most relevant to those studied in [14, 40, 41]. However, there is a fundamental difference between our model and all of these prior models.

- Walsh and Xia [40] studied a very similar election setting, where a random committee is first generated and then the winner is selected within the committee. However, they study a different manipulation model. They assume that voters are divided into manipulator(s) and non-manipulators, and they study whether the manipulator(s) can change preference in such a way that the winning probability of the designated candidate can increase. This is substantially different to our model as we assume the briber is able to bribe (and hence manipulate the preference of) any voter, with the objective of maximizing the winning probability of the designated candidate.
- Wojtas and Faliszewski [41] studied the problem where voters have no-show (i.e., absent in voting) probabilities. It is not difficult to observe that if a voter can be selected into the committee with probability p, then it is equivalent to that the voter has a no-show probability of $1 - p$. Therefore, [41] also studied a similar election setting. However, they are concerned with the prediction of possible winner(s), and showed ♯P-completeness of computing the probability that a certain candidate wins. Note that, this does not necessarily mean that it is computationally prohibitive to manipulate the result. Finding an optimal or near-optimal solution that is arbitrarily close to the optimal one may still be tractable.
- Chen et al. [14] considered the complexity of electoral manipulation when bribed voters have a probability of no-show. Note that while they considered the complexity of bribery, the randomness in their paper is only associated with bribed voters (i.e., without being bribed a voter votes deterministically), which is substantially different from the random sample voting schemes considered in this paper where the randomness of the voter set is independent of manipulation.

Despite the difference in modeling, our paper is greatly motivated by the above research works.

1.1 Our Contributions

The major contribution of this paper is to provide a systematic study on the computational vulnerability/resistance of the random sample voting scheme with several scoring rules under bribery attack.

Despite the common intuition that the introduction of random sample voting should always make the system more robust, we show that its effect is quite sophisticated and dependent on the election setting, or more precisely, on the number of candidates and voting rules. If the number of candidates is a fixed constant, then bribery in random sample voting can be solved in polynomial time for any scoring rule. This coincides with the fact that bribery problems in the classical deterministic setting are usually easy to solve when there are few candidates [12]. Consequently, random sample voting schemes do not help much when there are few candidates. On the other hand, if the number of candidates is part of the input, then there is no $O(1)$-approximation algorithm for the bribery problem with random sample voting schemes under k-approval for $k \geq 3$ as well as Borda (see Section 1.3 for a rigorous definition of different voting rules).

Our main technical contribution is a polynomial time approximation scheme (PTAS) for the bribery problem with random sample voting schemes under plurality and veto, when the number of candidates is part of the input. This is a surprising result, particularly as the winning probability of the designated candidate has a very convoluted mathematical expression and is difficult to compute even if a solution is given. We emphasize that our approximation algorithm is substantially different from many existing algorithms for stochastic optimization that utilizes the central limit theorem to bypass the obstacle in optimizing the tail probability (see, e.g. [13]). Indeed, if the central limit is used, then it will inevitably introduce an additive ϵ-error in the objective value, which can be significant when the optimal objective value is small. In contrast, our approximation scheme only incurs a multiplicative factor of $1 + \epsilon$.

In terms of techniques, our algorithmic result utilizes a non-standard integer programming (IP) formulation of the problem, followed by a sequence of modifications that accommodate the application of n-fold integer programming. The novel IP-formulation is not only crucial to the theoretical analysis but also proves to be very efficient when we adopt common IP-solvers to solve them in experiments. Our techniques for dealing with optimizing winning probabilities in random sample voting, particularly the adoption of n-fold integer programming in optimizing a sophisticated probability, may be of the separate interest for other stochastic optimization problems.

1.2 Related Work

The computational complexity of the bribery problems has been systematically studied in [24] and followed by a series of research works. We refer the reader to the book [7] for a comprehensive survey.

While most of the prior research works focus on deterministic electoral manipulation problems, uncertainty in these problems has received an increasing attention in recent years. In particular, uncertainty in elections has been investigated from the various aspects: voter's preference list is incomplete [3–5, 11, 33, 42]; bona fide incomplete voter's preference list [22, 39]; voter's preference list is under the probabilistic model [27, 31]; missing voters [16, 18]; additional candidates may be added [2, 15, 43]; incomplete knowledge about the voting rule [23, 25, 34, 41]. Bribery problem in multiple rounds of election/tournaments with the uncertain winning relationship between each candidate is investigated in [1, 36] . For the lobby problem, which is related but slightly different, uncertainty information is considered in [6].

We utilize n-fold integer programming. Extensive research has been conducted on efficient algorithms for n-fold integer programming [19, 20, 28, 29, 32].

1.3 Problem Statement

We give the formal definition of the bribery problem in random sample voting (BRSV).

There are a set of $m+1$ candidates and a set of n voters. The election is conducted on the $m+1$ candidates and a subset of voters through a random sample scheme, which is specified in the following paragraph. Each voter has a preference list (e.g., a permutation of all candidates) over all candidates. There is a voting rule $\mathcal{R}$. In this paper, we focus on the scoring rule that maps a preference list to an $(m+1)$-vector $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{m+1})$, where $\alpha_i \in \mathbb{Z}_{\geq 0}$ is the score assigned to the candidate on the i-th position of the preference list of voter v_j and $\alpha_1 \geq \alpha_2 \geq \ldots \geq \alpha_m$. The *total score* of a candidate is the summation of the scores it received from the voters. Popular scoring rules include:

- Plurality: $\alpha = (1, 0, 0, \cdots, 0, 0)$;
- Borda: $\alpha = (m, m-1, \cdots, 1, 0)$;
- k-approval: $\alpha = (\underbrace{1, 1, \ldots, 1}_{k}, \underbrace{0, 0, \ldots, 0}_{m+1-k})$;
- k-veto: $\alpha = (\underbrace{1, 1, \ldots, 1}_{m+1-k}, \underbrace{0, 0, \ldots, 0}_{k})$;

For convenience, we denote the bribery problem in random sample voting under voting rule $\mathcal{R}$ as BRSV-$\mathcal{R}$. The winner is the candidate who receives the highest score. Co-winners (e.g., if there are more than one candidates receive the highest score simultaneously, then all of them are winners) are allowed in our model.

The election runs a random sample voting scheme. In such a scheme, while there are n voters, only a subset of them will eventually vote. More precisely, each voter is selected into a committee independently with a probability of $p \in (0, 1]$. The election will eventually be conducted over the voter committee and the $m+1$ candidates, the winner(s) is determined solely by the preferences of voters within the committee.

We consider the bribery problem in elections with a random sample scheme. There is a briber/attacker who wants to make a designated candidate win. Without loss of generality, we assume the $(m+1)$-th candidate is the designated candidate. The briber can pay a voter-dependent cost c_j to voter j to change the preference of this voter arbitrarily. There is a total budget B for the briber.

We assume that the briber cannot manipulate the random sample scheme. Hence, given a fixed set of bribed voters, the briber may or may not succeed in making the designated candidate win, depending on which bribed voters are in the committee. The goal of the briber is to bribe a subset of voters within the budget such that the winning probability of the designated candidate is maximized.

Formally, the bribery problem in random sample voting under voting rule $\mathcal{R}$ is formulated as follows.

Bribery in Random Sample Voting (BRSV-$\mathcal{R}$)
Input: A set of $m+1$ candidates $D = \{d_1, \ldots, d_{m+1}\}$ where d_{m+1} is the designated candidate; a set of n voters $V = \{v_1, \ldots, v_n\}$, together with the preference of each voter; the voting rule $\mathcal{R}$; a bribe cost $c_j \in \mathbb{Q}^+$ for each voter $v_j \in V$; the probability $p \in (0, 1]$ of being selected into the random committee for each voter; total bribing budget $B \in \mathbb{Q}^+$.
Output: Bribe a subset of voters $V^* \subseteq V$ which maximizes the winning probability of the designated candidate d_{m+1} under voting rule $\mathcal{R}$ that satisfies $\sum_{v_j \in V^*} c_j \leq B$.

2 PRELIMINARY

In general, approximation algorithms are defined with respect to the multiplicative ratio.

Definition (α-approximation algorithm). *For a maximization problem, ALG is an α-approximation algorithm if for any instance I of the problem it holds that $\alpha \cdot ALG(I) \geq OPT(I)$.*

Our work relies on the recent breakthrough in integer linear programming with n-fold structure. To help understand, we give the definition and the current known result (Lemma 1) for n-fold integer linear programming in the beginning.

Definition (n-fold integer linear programming). *An integer linear programming $\max\{c^T x : \mathcal{A}x \leq b, \ell \leq x \leq u, x \in \mathbb{Z}^{nt}\}$ is called n-fold integer linear programming if the coefficient matrix $\mathcal{A}$ has the following structure*

$$\mathcal{A} = \begin{bmatrix} A^1 & A^2 & \cdots & A^n \\ B^1 & 0 & \cdots & 0 \\ 0 & B^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B^n \end{bmatrix}$$

where $A^1, \ldots, A^n \in \mathbb{Z}^{r \times t}$ are $r \times t$ matrices and $B^1, \ldots, B^n \in \mathbb{Z}^{s \times t}$ are $s \times t$ matrices.

Lemma 1. *[17] The optimal solution of n-fold integer linear programming can be solved by $2^{O(rs^2)}(rs\Delta)^{O(r^2 s + s^2)}(nt)^{1+o(1)}$ arithmetic operations where Δ denotes the upper bound on the absolute value of each entry of $\mathcal{A}$.*

3 RANDOM SAMPLE VOTING WITH ARBITRARY NUMBER OF CANDIDATES

3.1 Hardness

We observe that, BRSV-$\mathcal{R}$ problem incorporates the classical bribery problem introduced in [24] as a special case. More precisely, if $p = 1$, then every voter is deterministically selected into the committee, in this case, BRSV-$\mathcal{R}$ reduces to the classical bribery problem under voting rule $\mathcal{R}$. Consequently, if it is NP-hard to determine whether the designated candidate can win in the classical bribery problem under $\mathcal{R}$, then it becomes NP-hard to determine whether the winning probability of the designated candidate is 1 or 0 in BRSV-$\mathcal{R}$, implying that there is no $O(1)$-approximation algorithm. The statement remains true even if we restrict that $p \in (0, 1)$ instead $p \in (0, 1]$. This is because if we choose p arbitrarily close to 1, say, $p = 1 - 1/n^2$, then with sufficiently high probability (which is at least

$1-1/n$) all voters are selected into the committee, consequently it becomes NP-hard to distinguish between an instance with winning probability at least $1-1/n$ and an instance with winning probability at most $1/n$.

Notice that the classical bribery problem has been shown to be NP-hard for common voting rules including k-approval for $k \geq 3$ [35], k-veto for $k \geq 2$ [8], Borda [9], the following theorem follows directly according to our argument above.

Theorem 1. *Assuming P≠NP, there does not exist $O(1)$-approximation algorithm for BRSV-$\mathcal{R}$ if $\mathcal{R}$ is k-approval for $k \geq 3$ or k-veto for $k \geq 2$ or Borda.*

3.2 Algorithms for BRSV-Plurality

Given the strong inapproximability of BRSV under most of the natural voting rules, we now consider plurality, which is generally expected to be easier to solve (and thus vulnerable to bribery). Our goal in this section is to show that the introduction of random sample voting does not help. In particular, the (near-) optimal solution for the briber can be computed efficiently for BRSV-Plurality, as implied by the following two theorems.

Theorem 2. *There exists a greedy algorithm within $O(n+m\log m)$ time that returns an optimal solution for the BRSV-plurality problem when all bribery costs are unit, i.e., $c_j = 1$ for all $v_j \in V$.*

It is remarkable that the basic idea of greedy in Theorem 2 follows from that in the deterministic bribery problems considered in [24]. The major challenge is due to that the formula for computing the objective function, which is the winning probability, is very complicated. It does not admit a simple closed form but has to be expressed as a lengthy summation over binary probabilities, which makes the comparison of two feasible solutions very intricate. Towards this, we employ a non-trivial exchange argument and utilize Vandermonde's identity.

Theorem 3. *For any $\epsilon > 0$, there exists an approximation scheme (Algorithm 1) that runs in $(nmL/\epsilon)^{O(1/\epsilon^2)}$ time and outputs an $(1+\epsilon)$-approximation solution for BRSV-plurality, where L denotes the input length of problem.*

Unfortunately, the greedy algorithm fails once the bribery costs are no longer unit. Towards this, we leverage recent advances in integer programming to handle the general problem. There are two critical challenges. One is that the winning probability is too convoluted to serve directly as an objective function in an integer program. A common approach in stochastic optimization is to approximate it by the central limit theorem, however, this will inevitably create an additive error which violates our target of a multiplicative approximation. To handle this, we introduce the notion of "τ-segment" which provides a "staircase" approximation. Another challenge is that to model BRSV-plurality we have to introduce a lot of integer variables, while integer programming in high dimension is typically hard to solve. To handle this, we formulate a novel integer program NIP(ℓ), and provide a series of modification on NIP(ℓ) such that its constraint matrix has a specific structure that allows us to apply the algorithm for n-fold integer programming, which is a recent breakthrough achieved in the community of integer programming (see Lemma 1).

3.2.1 A Natural Integer Programming Formulation for BRSV-Plurality. We first provide a natural integer programming formulation of the BRSV-plurality problem. This will be useful for our greedy algorithm and will also serve as a starting point towards our novel integer programming formulation in the following subsection.

Under plurality rule e.g., $\alpha = (1, 0, \ldots, 0)$, for each voter only the candidate who is on the 1-st position of the preference list could get one score. Hence, for ease of notations, we say a voter votes for a candidate d_i if d_i is on the 1-st position of the voter's preference list. And denote V_i as the set of voters who vote for the candidate d_i in the absence of bribery.

Towards the integer programming formulation, we need the following functions. Suppose there remains y_i voters who vote for candidate d_i after bribery. We are mainly interested in the number of votes eventually received by candidate d_i, which is equal to the number of voters, among the y_i voters, who are selected into the committee. Let it be X_i, using the fact that each voter is selected independently with a probability of p, we have that

$$\Pr[X_i = t] = \phi_{y_i}(t) := \begin{cases} \binom{y_i}{t} p^t (1-p)^{y_i - t} & 0 \leq t \leq y_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

and

$$\Pr[X_i \leq t] = \Phi_{y_i}(t) := \begin{cases} 0 & t < 0 \\ \sum_{h=0}^{t} \phi_{y_i}(h) & 0 \leq t \leq y_i \\ 1 & t > y_i \end{cases} \quad (2)$$

Now we are ready to give the natural integer programming NIP, which has a non-linear objective function:

$$\begin{aligned} \max \quad & \sum_{t=0}^{n} \left(\phi_{y_{m+1}}(t) \cdot \prod_{i=1}^{m} \Phi_{y_i}(t) \right) \\ \text{s.t.} \quad & \sum_{j=1}^{n} c_j x_j \leq B && \text{(3a)} \\ & |V_{m+1}| + \sum_{j=1}^{n} x_j = y_{m+1} && \text{(3b)} \\ & \sum_{j \in V_i} (1 - x_j) = y_i \qquad \forall i \in [1, m] && \text{(3c)} \end{aligned}$$

Here we omit the constraints $x_j \in \{0, 1\}$ and $y_i \in \mathbb{N}$. The binary decision variable $x_j = 1$ denotes that voter j is bribed. The integral variable y_i represents the number of voters voting for candidate d_i after bribery.

We explain constraints: Eq (3a) represents that the total bribing cost cannot be larger than B. Note that under plurality rule, if a voter is bribed, then the voter will vote for the designated candidate d_{m+1}, and thus we have Eq (3b) and Eq (3c).

We explain the objective function of NIP. Recall that for any fixed t, $\phi_{y_{m+1}}(t)$ is the probability that exactly t voters voting for d_{m+1} are selected into the committee, and $\Phi_{y_i}(t)$ is the probability that at most t voters voting for d_i are selected into the committee. So $\phi_{y_{m+1}}(t) \cdot \prod_{i=1}^{m} \Phi_{y_i}(t)$ denotes the probability of the event that $X_{m+1} = t$ and $X_i \leq t, \forall 1 \leq i \leq m$ happens simultaneously. Taking the summation over t, this is the probability of the event that $X_i \leq X_{m+1}, \forall 1 \leq i \leq m$, i.e., when the designated candidate d_{m+1} becomes one of the co-winners.

3.2.2 *Proof of Theorem 2 – Optimal Algorithms for BRSV-Plurality with Unit Cost.*

In this subsection, we consider the BRSV problem when voters have the same bribing cost, i.e., $c_j = 1$ for all j.

Since the bribing cost is the same, we may well assume that the total number of voters we need to bribe is equal to $\min\{B, \sum_{i=1}^m |V_i|\}$. Without loss of generality, we assume $B \le \sum_{i=1}^m |V_i|$. This problem can be solved through the following simple greedy strategy: ignore the random sampling process and we simply focus on each V_i. Find the candidate who receives the most votes (i.e., with the largest $|V_i|$), and bribe one voter who votes for this candidate. We repeat the above process until the budget runs out. Obviously, this is the optimal algorithm in the deterministic election problem. It turns out that this remains an optimal algorithm under random sample voting. Towards showing its optimality, we need the following lemma that justifies the exchange argument we will adopt in proving Theorem 2.

Lemma 2. *Suppose* $\mathbf{y}, \mathbf{y}'$ *are two feasible solution of NIP which differ on exactly two coordinates* u, v *and satisfying*

$$y'_i = \begin{cases} y_i & i \neq u \text{ or } v, \\ y_i + 1 & i = u, \\ y_i - 1 & i = v. \end{cases}$$

If $y_u \le y_v - 2$, *then it holds that* $obj(\mathbf{y}) < obj(\mathbf{y}')$, *where* $obj(\mathbf{y}) = \sum_{t=0}^n (\phi_{y_{m+1}}(t) \cdot \prod_{i=1}^m \Phi_{y_i}(t))$ *is the objective function of NIP with respect to solution* $\mathbf{y}$.

Proof. We know $obj(\mathbf{y}) - obj(\mathbf{y}')$ equals to

$$\sum_{t=0}^n [\phi_{y_{m+1}}(t) \prod_{i \neq u,v} \Phi_{y_i}(t)] [\Phi_{y_u}(t)\Phi_{y_v}(t) - \Phi_{y_u+1}(t)\Phi_{y_v-1}(t)].$$

It suffices to compare $\Phi_{y_u}(t)\Phi_{y_v}(t)$ and $\Phi_{y_u+1}(t)\Phi_{y_v-1}(t)$. Note that $y'_u = y_u + 1 \le y_v - 1 = y'_v$, and observe that by definition $\Phi_y(t) = 1$ if $t \ge y$, we have the followings:

- For $t \ge y_v$: $\Phi_{y_u+1}(t)\Phi_{y_v-1}(t) = \Phi_{y_u}(t)\Phi_{y_v}(t) = 1$
- For $y_u + 1 \le t \le y_v - 1$: $\Phi_{y_u+1}(t)\Phi_{y_v-1}(t) = \Phi_{y_v-1}(t) > \Phi_{y_v}(t) = \Phi_{y_u}(t)\Phi_{y_v}(t)$.
- For $t \le y_u$: we observe that $\Phi_{y_u}(t)\Phi_{y_v}(t)$ equals to the the following:

$$\begin{aligned} &[\sum_{j=0}^t \binom{y_u}{j} p^j (1-p)^{y_u - j}][\sum_{k=0}^t \binom{y_v}{k} p^k (1-p)^{y_v-k}] \\ =& \sum_{k=0}^t \sum_{j=0}^t \binom{y_u}{j}\binom{y_v}{k} p^{j+k}(1-p)^{y_u+y_v-j-k} \\ =& \sum_{h=0}^t \sum_{j=0}^h p^h (1-p)^{y_u+y_v-h} \binom{y_u}{j}\binom{y_v}{h-j} \\ =& \sum_{h=0}^t p^h (1-p)^{y_u+y_v-h} \binom{y_u+y_v}{h} \end{aligned}$$

  Here the last equality follows from Vandermonde's identity. Consequently, $\Phi_{y_u}(t)\Phi_{y_v}(t) = \Phi_{y_u+1}(t)\Phi_{y_v-1}(t)$.

Therefore, we have $obj(\mathbf{y}) - obj(\mathbf{y}') < 0$ and the lemma is proved. □

Now we are ready to prove Theorem 2 utilizing Lemma 2.

Proof of Theorem 2. Let $\mathbf{y}^*$ be the optimal solution to NIP and $\hat{\mathbf{y}}$ be the solution returned by our greedy algorithm. We prove $obj(\mathbf{y}^*) = obj(\hat{\mathbf{y}})$, thus Theorem 2 is true.

Consider an arbitrary solution $\mathbf{y}$. Note that after bribery V_i may lose some voters as they get bribed. we define by $I(\mathbf{y}) = \{i \in [1,m] : y_i < |V_i|\}$ the indices of V_i's who lose voters, and $\bar{I}(\mathbf{y}) = \{i \in [1,m] : i \notin I(\mathbf{y})\}$ the indices of remaining ones.

Now consider the optimal solution $\mathbf{y}^*$. Define

$$u = \arg\min_{i \in I(\mathbf{y}^*)} y_i^*, \quad v = \arg\max_{i \in I(\mathbf{y}^*)} y_i^*.$$

That is, among those V_i's who lose voters, V_u is the one with the least remaining voters, and V_v is the one with the most remaining voters.

We claim that $y_v^* \le y_u^* + 1$. If the claim is not true, then we let $\mathbf{y}' = (y_1^*, \dots, y_u^* + 1, \dots, y_v^* - 1, \dots, y_{m+1}^*)$, i.e., we bribe one less voter from V_v and one more voter from V_u. According to Lemma 2, the objective value increases, contradicting the fact that $\mathbf{y}^*$ is optimal.

Next, we claim that for any $h \in \bar{I}(\mathbf{y})$, $y_h^* \le y_u^* + 1$. If the claim is not true, then we bribe one voter less from V_u, and meanwhile bribe one voter from V_h, by Lemma 2 the objective value will increase, contradicting that $\mathbf{y}^*$ is optimal.

The above claim implies that if $|V_i| < y_u^*$, then candidate d_i does not lose voters. Hence, all bribed voters are from V_h's where $h \in I_{\ge y_u^*} := \{i \in [1,m] : |V_i| \ge y_u^*\}$. Note that after bribery those V_i's have either y_u^* or $y_u^* + 1$ remaining voters. Denote by $\xi(\mathbf{y}^*)$ the number of V_i's with y_u^* remaining voters, then $|I_{\ge y_u^*}| - \xi(\mathbf{y}^*)$ is the number of V_i's with $y_u^* + 1$ remaining voters. Based on the fact that exactly B voters are bribed, we have

$$\xi(\mathbf{y}^*)y_u^* + \left(|I_{\ge y_u^*}| - \xi(\mathbf{y}^*)\right)(y_u^* + 1) + B = \sum_{i \in I_{\ge y_u^*}} |V_i|. \tag{4}$$

Now we are ready to compare $\mathbf{y}^*$ and $\hat{\mathbf{y}}$. Similarly we can define $\hat{u} = \arg\min_{i \in I(\hat{\mathbf{y}})} \hat{y}_i$, $\hat{v} = \arg\max_{i \in I(\hat{\mathbf{y}})} \hat{y}_i$. According to the greedy algorithm, we have $\hat{y}_{\hat{v}} \le \hat{y}_{\hat{u}} + 1$, and for any $\hat{h} \in \bar{I}(\hat{\mathbf{y}})$, we have $\hat{y}_{\hat{h}} \le \hat{y}_{\hat{u}} + 1$. Let $I_{\ge \hat{y}_{\hat{u}}} := \{i \in [1,m] : |V_i| \ge \hat{y}_{\hat{u}}\}$. After bribery, define $\xi(\hat{\mathbf{y}})$ the number of V_i's with $\hat{y}_{\hat{u}}$ remaining voters, then similarly $|I_{\ge \hat{y}_{\hat{u}}}| - \xi(\hat{\mathbf{y}})$ is the number of V_i's with $\hat{y}_{\hat{u}} + 1$ remaining voters, then

$$\xi(\hat{\mathbf{y}})\hat{y}_{\hat{u}} + \left(|I_{\ge \hat{y}_{\hat{u}}}| - \xi(\hat{\mathbf{y}})\right)(\hat{y}_{\hat{u}} + 1) + B = \sum_{i \in I_{\ge \hat{y}_{\hat{u}}}} |V_i|. \tag{5}$$

Compare Eq (5) with Eq (4), we claim that $\hat{y}_{\hat{u}} = y_u^*$. Suppose on the contrary that it does not hold, then either $\hat{y}_{\hat{u}} > y_u^*$ or $\hat{y}_{\hat{u}} < y_u^*$. We show a contradiction for $\hat{y}_{\hat{u}} > y_u^*$ and the other case can be proved in a similar way. If $\hat{y}_{\hat{u}} \ge y_u^* + 1$, then

$$\begin{aligned} &\xi(\hat{\mathbf{y}})\hat{y}_{\hat{u}} + \left(|I_{\ge \hat{y}_{\hat{u}}}| - \xi(\hat{\mathbf{y}})\right)(\hat{y}_{\hat{u}} + 1) \\ \ge& |I_{\ge \hat{y}_{\hat{u}}}|(y_u^* + 1) \\ \ge& \xi(\mathbf{y}^*)y_u^* + \left(|I_{\ge y_u^*}| - \xi(\mathbf{y}^*)\right)(y_u^* + 1), \end{aligned}$$

that is, the left side of Eq (5) is larger than or equal to the left side of Eq (4). By definition $I_{\ge \hat{y}_{\hat{u}}} \subseteq I_{\ge y_u^*}$, whereas $\sum_{i \in I_{\ge \hat{y}_{\hat{u}}}} |V_i| \le \sum_{i \in I_{\ge y_u^*}} |V_i|$, i.e., the right side of Eq (5) is smaller than or equal to the right side of Eq (4). Hence, for Eq (4) and Eq (5) to both hold, all inequalities above shall be equalities simultaneously, which is impossible through simple calculations. Therefore, $\hat{y}_{\hat{u}} = y^*_u$, which further implies that $\xi(\mathbf{y}^*) = \xi(\hat{\mathbf{y}})$. Thus $obj(\mathbf{y}^*) = obj(\hat{\mathbf{y}})$ follows directly by simply comparing all the terms in their expressions. □

3.2.3 Proof of Theorem 3 – Approximation Scheme for Arbitrary Cost. A crucial problem of the natural integer programming formulation we presented in the above subsection is that its objective function is too complex to approach theoretically as well as to conduct experimental studies via existing optimization toolboxes such as Gurobi or CPLEX. To handle this, we leverage the idea of "configuration linear programming" and introduce new variables for each V_i instead of each voter.

A New Integer Programming Formulation. It is easy to see that if we know the number of voters within V_i that are bribed, then we will always bribe the cheapest voters. Therefore, we define λ_{ik} as the total bribing cost of the cheapest $|V_i| - k$ voters within V_i. Suppose the total number of bribed voters is $\ell - |V_{m+1}| \in [0, n]$, that is, ℓ is the total number of voters preferring the designated candidate d_{m+1} after bribery. We propose the following integer programming with a non-linear objective, NIP(ℓ), for BRSV-plurality:

$$\max \quad \sum_{t=0}^{n} \phi_\ell(t) e^{z_t}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} \sum_{k=0}^{|V_i|} \lambda_{ik} x_{ik} \le B \tag{6a}$$

$$\sum_{i=1}^{m} \sum_{k=0}^{|V_i|} k x_{ik} = n - \ell \tag{6b}$$

$$\sum_{k=0}^{|V_i|} x_{ik} = 1 \qquad \forall i \in [1, m] \tag{6c}$$

$$\sum_{i=1}^{m} \sum_{k=0}^{|V_i|} \ln(\Phi_k(t)) x_{ik} = z_t \qquad \forall t \in [0, n] \tag{6d}$$

Here we omit the constraints $x_{ik} \in \{0, 1\}$ and $z_t \in \mathbb{R}$. The binary decision variable x_{ik} indicates that if $x_{ik} = 1$, then there are exactly k voters vote for candidate d_i after bribery (e.g., $|V_i| - k$ voters among V_i are bribed).

We explain the constraints. First notice that Eq (6c) enforces that for candidate i there exists one and only one k such that $x_{ik} = 1$, which implies that we bribe $|V_i| - k$ voters in V_i. As bribing the cheapest $|V_i| - k$ voters in V_i costs λ_{ik}, adding up the bribery cost for each V_i shall not exceed the budget B, as is implied by Eq (6a). Further, adding up the number of bribed voters, which is $|V_i| - k$ for each V_i, shall be exactly ℓ. Using the fact that $\sum_i |V_i| = n$, Eq (6b) follows. Finally, by Eq (6d) we use the supplementary variable z_t to denote the logarithm of the probability of the event that $X_j \le t$ happens simultaneously for $1 \le j \le m$. Consequently, e^{z_t} is the probability of this event. Recall that $\phi_\ell(t)$ is the probability that $X_{m+1} = t$, $\sum_{t=0}^{n} \phi_\ell(t) e^{z_t}$ is thus exactly the probability of the event that $X_j \le X_{m+1}$ for all $1 \le j \le m$.

Remark. It is worth mentioning that using z_t is merely to simplify the objective function. The reader may simply substitute z_t in the objective with Eq (6d) to obtain an objective function in x_{ik}'s. Notice that while the current objective function $\sum_{t=0}^{n} \phi_\ell(t) e^{z_t}$ is separable convex (in z_t's), constraint Eq (6d) contains $n+1$ inequalities where each of them involves all x_{ik}'s, which is far from the structure of n-fold IP. On the other hand, if we remove Eq (6d) and rewrite the objective function in x_{ik}'s, then the objective function is no longer separable convex. Hence, we cannot directly apply the algorithm for n-fold IP (see e.g., [20]) to solve NIP(ℓ). New techniques are needed to further modify the structure of NIP(ℓ).

Notice that we do not know how many voters are bribed in the optimal solution, however, we may simply solve NIP(ℓ) for each integer $\ell \in [0, n]$ and pick the best solution. The remaining part of this section is devoted to the solving of NIP(ℓ) for each fixed ℓ.

Lemma 3. *There exists an algorithm which runs in $(nmL/\epsilon)^{O(1/\epsilon^2)}$ time and outputs an $(1+\epsilon)$-approximation solution for NIP(ℓ) where L is the encode length of NIP(ℓ).*

To solve NIP(ℓ), the high-level idea is to utilize a recent breakthrough in integer programming – the algorithm for n-fold integer programming. Recall Lemma 1 and the structure of n-fold integer programming. There are two major issues with NIP(ℓ) that prevent us from applying the algorithm: (i) the objective function is non-linear; (ii) the constraints involve huge coefficients λ_{ik}, while noting that the running of n-fold integer programming depends on Δ, the largest absolute value of coefficients in the constraints. In the following, we handle these two issues.

Dealing with the nonlinear objective function. To handle the nonlinearity, we introduce the following notion.

Definition. *Let $(\mathbf{x}, \mathbf{z})$, $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ be two feasible solutions to NIP(ℓ). We say $(\mathbf{x}, \mathbf{z})$ is δ-better than $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$, if for any t it holds that $z_t \ge \hat{z}_t - \delta$.*

Based on the above definition, simple calculations lead to the following observation.

Observation 1. *If $(\mathbf{x}, \mathbf{z})$ is $k \cdot \log(1+\delta)$-better than $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$, then*

$$\sum_{t=0}^{n} \phi_\ell(t) e^{\hat{z}_t} \le (1 + 2k\delta) \sum_{t=0}^{n} \phi_\ell(t) e^{z_t}.$$

That is, the objective value of the two solutions differ by at most a multiplicative factor of $1 + 2k\delta$.

The above property provides a way to bypass the nonlinearity of the objective function. In particular, denote by $(\mathbf{x}^*, \mathbf{z}^*)$ the optimal solution to NIP(ℓ), then a feasible solution $(\mathbf{x}, \mathbf{z})$ has an objective value at least $1+\epsilon$ fraction of the optimal value it is $k \cdot \log(1+\epsilon/2k)$-better than $(\mathbf{x}^*, \mathbf{z}^*)$. Unfortunately, the optimal solution is unknown. To handle this issue, we need to further introduce the following concept called τ-segment vector.

Definition. *For any feasible solution $(\mathbf{x}, \mathbf{z})$ to NIP(ℓ), a τ-dimensional vector $S(\mathbf{x}) = (S_1(\mathbf{x}), S_2(\mathbf{x}), \cdots, S_\tau(\mathbf{x})) \in \mathbb{N}^\tau$ is called the τ-segment vector of $(\mathbf{x}, \mathbf{z})$ if it satisfies the following property:*

- *for any $t \le S_j(\mathbf{x})$, it holds that*

$$z_t = \sum_{i=1}^{m} \sum_{k=0}^{|V_i|} \ln(\Phi_k(t)) x_{ik} \le \ln(t/\tau)$$

- *for any $t > S_j(\mathbf{x})$, it holds that*

$$z_t = \sum_{i=1}^{m} \sum_{k=0}^{|V_i|} \ln(\Phi_k(t)) x_{ik} > \ln(t/\tau)$$

Here τ can be viewed as a control over the precision. In the final part of this section we will show that setting $\tau = 1/\log(1 + \frac{\epsilon}{4}) \leq O(1/\epsilon)$ suffices.

Figure 1 depicts the value of e^{z_j}'s for a feasible solution $(\mathbf{x}, \mathbf{z})$ in a toy example consisting of 10 voters (i.e., $n = 10$). We can see that for $\tau = 5$, the 5-segment vector of $(\mathbf{x}, \mathbf{z})$ is $(2, 3, 5, 6, 8)$.

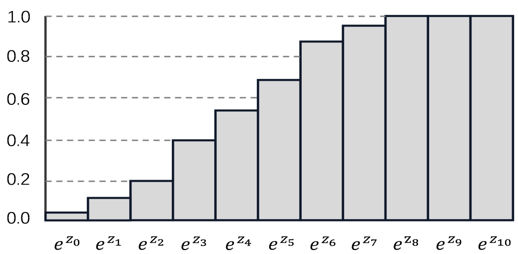

Figure 1: Illustration of k-segment vector

Based on the above definition, simple calculations lead to the following observation.

Observation 2. *Let $(\mathbf{x}, \mathbf{z})$, $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$ be two feasible solutions to NIP(ℓ). Let $S(\mathbf{x})$ and $S(\hat{\mathbf{x}})$ be the τ-segment of these two solutions, respectively. If $S_j(\mathbf{x}) \geq S_{j-k}(\hat{\mathbf{x}})$ for all $j \geq k$, then $(\mathbf{x}, \mathbf{z})$ is $\frac{k}{\tau}$-better than $(\hat{\mathbf{x}}, \hat{\mathbf{z}})$.*

While the optimal solution to NIP(ℓ) is unknown, we can guess its τ-segment vector through at most n^τ enumerations. Denoted by S^* the τ-segment vector of the optimal solution. We can further transform NIP(ℓ) into a feasibility test problem of the following integer linear programming denoted as ILP(ℓ, S^*):

$$\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \lambda_{ik} x_{ik} \leq B \tag{7a}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} k x_{ik} = n - \ell \tag{7b}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \ln(\Phi_k(S_t^*)) x_{ik} \leq \ln(t/\tau) \qquad \forall t \in [1, \tau] \tag{7c}$$

$$\sum_{k=0}^{|V_i|} x_{ik} = 1 \qquad \forall i \in [1, m] \tag{7d}$$

Here again we omit the constraint $x_{ik} \in \{0, 1\}$. The binary decision variable $x_{ik} = 1$ denotes that $|V_i| - k$ voters in V_i are bribed.

We explain constraints: Eq (7a), Eq (7b), Eq (7d) are same to the Eq (6a), Eq (6b), Eq (6d) of NIP(ℓ); Eq (7c) requires that each coordinate of the solution's τ-segment vector is no less than the corresponding coordinate of S^*. Of course, the optimal solution will satisfy all the constraints.

According to our previous analysis, by setting the parameter τ to be $\tau = 1/\log(1 + \frac{\epsilon}{2})$, we know that if S^* is guessed correctly, then any feasible solution to ILP(ℓ, S^*) is a $(1 + \epsilon)$-approximation solution to NIP(ℓ).

Note that ILP(ℓ, S^*) is a feasibility test problem and it only involves linear constraints whose structure follows that of n-fold integer programming. However, recall that by Lemma 1 the running time of the algorithm for n-fold integer programming depends on Δ, the largest absolute value among all entries. It remains to deal with the large coefficients in ILP(ℓ, S^*).

Dealing with huge coefficients in constraints. First note that ILP(ℓ, S^*) involves non-integral coefficients i.e., $\ln(\Phi_k(S_t^*))$ in Eq (7c). Fortunately, these non-integral coefficients can be scaled up and rounded to the nearest integer. Through the following calculation, we can show it only introduces an additive error of $1/\tau$ error via rounding up to a multiple of $1/mn\tau$. Notice that while the error here is additive, Eq (7c) is the logarithm the probability used in the objective function, whereas an additive error of $1/\tau$ eventually leads to a multiplicative factor of $e^{1/\tau}$, which is bounded by $1 + O(\epsilon)$ once we set $\tau = 1/\log(1 + \frac{\epsilon}{4})$.

$$\begin{aligned}&\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \frac{\lfloor mn\tau \ln(\Phi_k(S_t^*))\rfloor}{mn\tau} x_{ik}\\ \geq &\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} (\ln(\Phi_k(S_t^*)) - \frac{1}{mn\tau}) x_{ik}\\ \geq &\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \ln(\Phi_k(S_t^*)) x_{ik} - \frac{1}{\tau}\end{aligned}$$

Finally, we observe that λ_{ik} could be some value that is very large as it denotes the least bribing cost for $|V_i| - k$ voters among V_i. Notice that except λ_{ik}'s, the coefficients in all the other constraints of ILP(ℓ, S^*) have an absolute value which is bounded by a polynomial in n. Given that ILP(ℓ, S^*) is a feasibility test problem, we can shift Eq (7a) to the objective, and derive the following n-fold integer linear programming RIP(ℓ, S^*):

$$\min \quad \sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \lambda_{ik} x_{ik}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{k=0}^{|V_i|} k x_{ik} = n - \ell \tag{8a}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} f_{kt} x_{ik} \leq mn\tau \ln(t/\tau) \qquad \forall t \in [1, \tau] \tag{8b}$$

$$\sum_{k=0}^{|V_i|} x_{ik} = 1 \qquad \forall i \in [1, m] \tag{8c}$$

Here again we omit the constraint $x_{ik} \in \{0, 1\}$. The constraint Eq (8b) is a rounded version of Eq (7c) where $f_{kt} = \lfloor mn\tau \ln(\Phi_k(S_t^*))\rfloor$.

It is easy to see the coefficient matrix of RIP(ℓ, S^*) has n-fold structure with $r = \tau{+}1, s = 1, t = n{+}1$ and $\Delta = mn^2\tau \max\{|\log(p)|, |\log(1-p)|\}$. As we know, $|\log(p)|$ and $|\log(1-p)|$ are both bounded by the input length of the problem.

As stated in Lemma 1, solving the optimal solution of RIP(ℓ, S^*) costs $(nm\tau L)^{O(\tau^2)}$ time where L denotes the input length of problem. In total, we solve RIP(ℓ, S^*) for $n^{O(\tau)}$ times and get an feasible solution which is $\frac{2}{\tau}$-better than the optimal solution of the BRSV

Algorithm 1 Approximation Scheme for BRSV-plurality

Input: $m, n, B, \epsilon, \{V_1, \ldots, V_{m+1}\}$
Output: $\hat{x}$

$\mathcal{F} = \emptyset$; $\tau = 1/\log(1 + \frac{\epsilon}{4})$;
for all $\ell \in [|V_d|, n]$ and $S^* \in \mathbb{N}^\tau$ **do**
  solve the optimal solution x of RIP(ℓ, S^*)
  if $\sum_{i=1}^{m} \sum_{k=0}^{|V_i|} \lambda_{ik} x_{ik} \le B$ **then**
    $\mathcal{F} \leftarrow \mathcal{F} \cup (x, \ell, S^*)$
  end if
end for
$\hat{x} \leftarrow \arg\max_{(x,\ell,S^*)\in\mathcal{F}} \sum_{t=0}^{n} \phi_\ell(t) e^{\sum_{i=1}^{m}\sum_{k=0}^{|V_i|} \ln(\Phi_k(t)) x_{ik}}$

problem. Hence, setting $\tau = 1/\log(1 + \frac{\epsilon}{4}) \le O(1/\epsilon)$, Theorem 3 is proved. We summarize our algorithm as Algorithm 1.

3.3 Algorithms for BRSV-Veto

Our goal in this section is to show that random sample voting under the veto (i.e., 1-veto) rule is computationally vulnerable to bribery in the sense that the (near-)optimal solution for the attacker/briber can be computed in a very efficient way. More precisely, we prove the following theorem.

THEOREM 4. *For any $\epsilon > 0$, there exists an approximation scheme within $(nmL/\epsilon)^{O(1/\epsilon^2)}$ time that outputs an $(1 + \epsilon)$-approximation solution for the BRSV-Veto problem, where L denotes the input length of problem.*

Recall that the score vector for veto is $\alpha = (1, \ldots, 1, 0)$, all candidates could get one score except the one who is on the last position of the preference list. In fact, we observe that the BRSV-veto problem can be viewed as the following equivalent form: each voter votes for one candidate i.e., the one who is on the last position of his/her preference list and the winner is the candidate who receives the *least* number of votes.

For simplicity, under veto, we say a voter votes for the candidate d_i if d_i is on the last position of the voter's preference list and denote V_i as the set of voters who votes the candidate d_i in the absence of bribery.

Recall the nonlinear integer programming NIP(ℓ) introduced for the BRSV-plurality problem. Under the plurality rule, we can see that there is no need to bribe voters between two undesignated candidates. When we bribe an voter from an undesignated candidate, it is always better to let he/her vote for the designated candidate rather than other candidates.

When considering the veto rule, we need to consider two kinds of bribery: from the designated candidate to one of the undesignated candidate; from one of the undesignated candidate to another undesignated candidate. Similarly, replacing Eq (6a) (6d) with Eq (9a) (9d), we can still formulate the BRSV-veto problem as the following nonlinear integer programming NIP2(ℓ):

$$\max \quad \sum_{t=0}^{n} \phi_\ell(t) e^{z_t}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{k=0}^{n} \hat{\lambda}_{ik} x_{ik} \le B - \lambda_{m+1,\ell} \tag{9a}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{n} k x_{ik} = n - \ell \tag{9b}$$

$$\sum_{k=0}^{n} x_{ik} = 1 \qquad \forall i \in [1, m] \tag{9c}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{n} \ln(1 - \Phi_k(t-1)) x_{ik} = z_t \qquad \forall t \in [0, n] \tag{9d}$$

where $\ell \le |V_{m+1}|$ denotes the total number of voters preferring the designated candidate d_{m+1} after bribery.

The definition of the decision variable x_{ik} and z_t are same to NIP(ℓ). We omit the constraints $x_{ik} \in \{0, 1\}$ and $z_t \in \mathbb{R}$. The binary decision variable x_{ik} indicates that if $x_{ik} = 1$, then there are exactly k voters vote for candidate d_i after bribery. It has to be noticed that for each candidate i we need to take n decision variables (e.g., $x_{i1}, x_{i2}, \ldots, x_{in}$) into consideration rather than $|V_i|$ decision variables in NIP(ℓ).

The coefficient $\hat{\lambda}_{ik} = \lambda_{ik}$ for $k \le |V_i|$, and $\hat{\lambda}_{ik} = 0$ for $k > |V_i|$. And λ_{ik} means the total bribing cost of the cheapest $|V_i| - k$ voters within V_i which is initially introduced in NIP(ℓ).

We explain the objective: recall the definition of $\phi_{y_i}(t)$ and $\Phi_{y_i}(t)$ see e.g., Eq (1) (2). For any fixed t, $\phi_{y_{m+1}}(t)$ is the probability that exactly t voters voting for d_{m+1} are selected into the committee, and $\Phi_{y_i}(t)$ is the probability that at most t voters voting for d_i are selected into the committee. So $\phi_{y_{m+1}}(t) \cdot \prod_{i=1}^{m} [1 - \Phi_{y_i}(t-1)]$ denotes the probability of the event that $X_{m+1} = t$ and $X_i \ge t, \forall 1 \le i \le m$ happens simultaneously. Taking the summation over t, this is the probability of the event that $X_i \ge X_{m+1}, \forall 1 \le i \le m$, i.e., when the designated candidate d_{m+1} becomes one of the co-winners[1].

Through simple calculation, we can found that the technique introduced in Section 3.3 - "Dealing with the nonlinear objective function" and "Dealing with huge coefficients in constraints" can still be utilized for dealing with NIP2(ℓ). By replacing the coefficient λ_{ik} appearing in the objective of NIP2(ℓ) with $\hat{\lambda}_{ik}$ and the coefficient f_{kt} appearing in Eq (9d) with $\hat{f}_{kt} = \lfloor mn\tau \ln(1 - \Phi_k(S_t^* - 1)) \rfloor$.

[1] If co-winners are not allowed, we can simply replace the objective function with the probability of $X_i > X_{m+1}, \forall 1 \le i \le m$.

We can derive a n-fold integer linear programming RIP2(ℓ, S^*):

$$\min \quad \sum_{i=1}^{m}\sum_{k=0}^{|V_i|}\hat{\lambda}_{ik}x_{ik}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{k=0}^{|V_i|}kx_{ik} = n-\ell \tag{10a}$$

$$\sum_{i=1}^{m}\sum_{k=0}^{|V_i|}\hat{f}_{kt}x_{ik} \leq mn\tau\ln(t/\tau) \qquad \forall t\in[1,\tau] \tag{10b}$$

$$\sum_{k=0}^{|V_i|}x_{ik} = 1 \qquad \forall i\in[1,m] \tag{10c}$$

Similarly, we have the following observation.

Corollary 1. *There exists an algorithm which runs in $(nmL/\epsilon)^{O(1/\epsilon^2)}$ time and outputs an $(1+\epsilon)$-approximation solution for NIP2(ℓ) where L is the encode length of NIP2(ℓ).*

4 RANDOM SAMPLE VOTING WITH A CONSTANT NUMBER OF CANDIDATES

We consider BRSV when the number of candidates, $m+1$, is a fixed constant. We show that under an arbitrary scoring rule, BRSV admits a polynomial time algorithm with a constant number of candidates, more precisely,

Theorem 5. *For any scoring rule $\mathcal{R}$, there exists an algorithm that outputs an optimal solution for the BRSV-$\mathcal{R}$ problem within $n^{(m+1)!}$ time.*

We know the total different kinds of preference order is $M := (m+1)!$. Define $S_{i,k}$ as the set of preference orders where candidate i is on the k-th position of the preference order. For each preference order σ_j, we define N_j as the set of voters whose preference is σ_i after bribing.

Lemma 4. *The winning probability of the designated candidate after bribing is only dependent on $|N_j|$'s, where $1\leq j\leq M=(m+1)!$.*

Proof. Suppose the random committee is selected to be $V'\subseteq V$. Then, whether the designated candidate d_{m+1} has no less score than candidate d_i can be characterized by the following $(0,1)$-indicate function $g_i(V')$ where $g_i(V')=1$ if and only if the following holds

$$\sum_k\sum_{j:\sigma_j\in S_{i,k}}\alpha_k\cdot|V'\cap N_j| \leq \sum_k\sum_{j\in S_{m+1,k}}\alpha_k\cdot|V'\cap N_j|.$$

Furthermore, whether the designated candidate wins or not if the random committee is selected to be $V'\subseteq V$ can be characterized by the $(0,1)$-indicate function $g(V')$ which is defined as below

$$g(V') = \prod_i g_i(V').$$

We know that, in the absence of bribery, the winning probability of the designated candidate equals

$$\sum_{V'\subseteq V} g(V')Pr[\text{RC}=V'],$$

where the random variable RC denotes the set of voters be selected into the random committee.

We can expand $g(V')$ as following

$$g(V') = \sum_{n_1,\ldots,n_M}\Big(\prod_{i=1}^{M}\mathbb{1}_{|V'\cap N_j|=n_j}\Big)\cdot$$
$$g(V' \mid |V'\cap N_1|=n_1\&\ldots\&|V'\cap N_M|=n_M)$$

where $\prod_j \mathbb{1}_{|V'\cap N_j|=n_j}$ is the $(0,1)$-indicate function which equals 1 if and only if it holds that $|V'\cap N_1|=n_1$ and ... and $|V'\cap N_M|=n_M$.

Observing the definition of the $g(V')$, it is no hard to see when $|V'\cap N_j|=n_j$ the function $g(V')$ can be expressed in terms of $n_1,\ldots,n_M$. Abuse the notation, we denote it as $g(n_1,\ldots,n_M)$.

Hence, the winning probability of the designated candidate can be reformulated as the following form

$$\sum_{n_1,\ldots,n_M} g(n_1,\ldots,n_M)\sum_{V'\subseteq V}\prod_{j=1}^{M}\mathbb{1}_{|V'\cap N_j|=n_j}Pr[\text{RC}=V']$$

Since, N_i does not overlap with each other and $\bigcup_{j=1}^{M}N_j=V$. We know that the probability $\sum_{V'\subseteq V}\prod_i\mathbb{1}_{|V'\cap N_j|=n_j}\cdot Pr[\text{RC}=V']$ can be expressed as the following closed form

$$\prod_{j=1}^{M}\binom{n_j}{|N_j|}p^{n_j}(1-p)^{|N_j|-n_j} \qquad \square$$

There are $n^{(m+1)!}$ different possibilities on all the values of $|N_j|$. Given the values of all $|N_j|$'s, it is straightforward to calculate the minimal total bribing cost needed to change preferences. Hence, Theorem 5 is true.

5 EXPERIMENTS

It is remarkable that our theoretical proofs for Theorem 2 and Theorem 3 already demonstrate the efficiency of our algorithms – they run in polynomial time even in worst case. In this section, we further show that NIP(ℓ), and hence BRSV-Plurality, can be solved efficiently on the data set by directly using common IP-solvers like Gurobi (albeit that such a method does not guarantee a worst case running time).

We generate instances as follows. We set three different values on the number of candidates, i.e., $m=3;5;10$, three different values on the number of voters, i.e., $n=100;300;500$ and three different values on the probability of being selected into the committee, i.e., $p=0.3;0.5;0.7$. To generate their preferences, we randomly select voters and candidates from the Sushi Data [30], which is a commonly used data set for generating preference (see, e.g. [38]). For each combination of (n,m), 50 instances are generated from the data set. In each instance, we randomly select a candidate as the designated candidate. The bribing cost of each voter is selected uniformly at random from $[50,100]$, and the total budget is randomly generated within the maximal value (which is the summation of all bribing cost).

For each generated instance, we run Gurobi[2] to solve NIP(ℓ) (the sophisticated nonlinear objective function is approximated through a piece-wise linear function within the error of 0.01). The running time is summarize in Table 1.

[2]Procedures are coded in C++ using Gurobi 9.0.3. All experiments are performed on a computer with an Intel Xeon-W2295 processor and 256GB memory.

average running time [s]			# of candidates		
			3	5	10
# of voters	100	$p = 0.3$	<2	<2	2
		$p = 0.5$	<2	<2	<2
		$p = 0.7$	<2	<2	<2
	300	$p = 0.3$	10	14	13
		$p = 0.5$	28	47	36
		$p = 0.7$	8	11	8
	500	$p = 0.3$	34	162	62
		$p = 0.5$	78	323	202
		$p = 0.7$	37	52	35

Table 1: Solving time of the BRSV-Plurality problem via NIP(ℓ) formulation

6 CONCLUSION

In this paper, we give a systematic study on the computational vulnerability/resistance of elections with random sample voting schemes under bribery attack, which incorporates the classical bribery problem as a special case. We show strong inapproximability results for k-approval where $k \geq 2$, and for k-veto where $k \geq 2$. We then complement our results by showing a greedy algorithm for BRSV-plurality with unit bribery costs, and polynomial time approximation schemes for BRSV-plurality and BRSV-veto for arbitrary bribery costs. Finally, we show that if the number of candidates is a constant, then BRSV can be solved for arbitrary scoring rule, which coincides with the deterministic bribery problem. One important open problem is whether the decision version of BRSV-plurality and BRSV-veto (e.g., given the threshold $T \in \mathbb{Q}^+$ whether is it possible to bribe a subset of voters such that the winning probability of the designated candidate is no less than T and satisfies the budget constraint) is NP-hard, despite that our algorithmic results already imply its vulnerability. Another interesting open problem is whether there exists an FPT (fixed-parameter tractable) algorithm parameterized by the number of candidates for BRSV under an arbitrary scoring rule. Finally, we consider the bribery model where a voter can pay a fixed amount of money to a voter to change his/her preference arbitrarily. It is interesting to consider other bribery models, especially the swap bribery model (see, e.g. [21]).

REFERENCES

[1] Haris Aziz, Serge Gaspers, Simon Mackenzie, Nicholas Mattei, Paul Stursberg, and Toby Walsh. 2018. Fixing balanced knockout and double elimination tournaments. *Artificial Intelligence* 262 (2018), 1–14.

[2] Dorothea Baumeister, Magnus Roos, and Jörg Rothe. 2011. Computational complexity of two variants of the possible winner problem. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems.* 853–860.

[3] Dorothea Baumeister and Jörg Rothe. 2012. Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules. *Inform. Process. Lett.* 112, 5 (2012), 186–190.

[4] Nadja Betzler and Britta Dorn. 2010. Towards a dichotomy for the possible winner problem in elections based on scoring rules. *J. Comput. System Sci.* 76, 8 (2010), 812–836.

[5] Nadja Betzler, Susanne Hemmann, and Rolf Niedermeier. 2009. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence.* 53–58.

[6] Daniel Binkele-Raible, Gábor Erdélyi, Henning Fernau, Judy Goldsmith, Nicholas Mattei, and Jörg Rothe. 2014. The complexity of probabilistic lobbying. *Discrete Optimization* 11 (2014), 1–21.

[7] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice.* Cambridge University Press.

[8] Robert Bredereck and Nimrod Talmon. 2016. NP-hardness of two edge cover generalizations with applications to control and bribery for approval voting. *Inform. Process. Lett.* 116, 2 (2016), 147–152.

[9] Eric Brelsford, Piotr Faliszewski, Edith Hemaspaandra, Henning Schnoor, and Ilka Schnoor. 2008. Approximability of manipulating elections. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, Vol. 8. 44–49.

[10] Vitalik Buterin. 2017. Ethereum sharding faq.

[11] Ioannis Caragiannis, Xenophon Chatzigeorgiou, George A Krimpas, and Alexandros A Voudouris. 2019. Optimizing positional scoring rules for rank aggregation. *Artificial Intelligence* 267 (2019), 58–77.

[12] Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. 2017. Elections with few voters: candidate control can be easy. *Journal of Artificial Intelligence Research* 60 (2017), 937–1002.

[13] Lin Chen, Lei Xu, Shouhuai Xu, Zhimin Gao, and Weidong Shi. 2019. Election with bribe-effect uncertainty: a dichotomy result. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence.* 158–164.

[14] Lin Chen, Lei Xu, Shouhuai Xu, Zhimin Gao, and Weidong Shi. 2019. Election with bribed voter uncertainty: hardness and approximation algorithm. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Vol. 33. 2572–2579.

[15] Yann Chevaleyre, Jérôme Lang, Nicolas Maudet, and Jérôme Monnot. 2010. Possible winners when new candidates are added: the case of scoring rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence.* 762–767.

[16] Vincent Conitzer, Toby Walsh, and Lirong Xia. 2011. Dominating manipulations in voting with partial information. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence.* 638–643.

[17] Jana Cslovjecsek, Friedrich Eisenbrand, Christoph Hunkenschröder, Lars Rohwedder, and Robert Weismantel. 2021. Block-structured integer and linear programming in strongly polynomial and near linear time. In *Proceedings of the 32nd ACM-SIAM Symposium on Discrete Algorithms.* 1666–1681.

[18] Palash Dey, Nimrod Talmon, and Otniel van Handel. 2017. Proportional representation in vote streams. *arXiv preprint arXiv:1702.08862* (2017).

[19] Friedrich Eisenbrand, Christoph Hunkenschröder, and Kim-Manuel Klein. 2018. Faster algorithms for integer programs with block structure. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming.* 49:1–49:13.

[20] Friedrich Eisenbrand, Christoph Hunkenschröder, Kim-Manuel Klein, Martin Koutecký, Asaf Levin, and Shmuel Onn. 2019. An algorithmic theory of integer programming. *arXiv preprint arXiv:1904.01361* (2019).

[21] Edith Elkind, Piotr Faliszewski, and Arkadii Slinko. 2009. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory.* Springer, 299–310.

[22] Ulle Endriss, Maria Silvia Pini, Francesca Rossi, and K Brent Venable. 2009. Preference aggregation over restricted ballot languages: sincerity and strategy-proofness. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence.* 122–127.

[23] Gabor Erdelyi, Edith Hemaspaandra, and Lane A Hemaspaandra. 2014. Bribery and voter control under voting-rule uncertainty. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems.* 61–68.

[24] Piotr Faliszewski, Edith Hemaspaandra, and Lane A Hemaspaandra. 2009. How hard is bribery in elections? *Journal of artificial intelligence research* 35 (2009), 485–532.

[25] Serge Gaspers, Victor Naroditskiy, Nina Narodytska, and Toby Walsh. 2013. Possible and necessary winner problem in social polls. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems.* 613–620.

[26] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles.* 51–68.

[27] Noam Hazon, Yonatan Aumann, Sarit Kraus, and Michael Wooldridge. 2012. On the evaluation of election outcomes under uncertainty. *Artificial Intelligence* 189 (2012), 1–18.

[28] Raymond Hemmecke, Shmuel Onn, and Lyubov Romanchuk. 2013. N-fold integer programming in cubic time. *Mathematical Programming* 137, 1 (2013), 325–341.

[29] Klaus Jansen, Alexandra Lassota, and Lars Rohwedder. 2020. Near-linear time algorithm for n-fold ILPs via color coding. *SIAM Journal on Discrete Mathematics* 34, 4 (2020), 2282–2299.

[30] Toshihiro Kamishima. 2003. Nantonac collaborative filtering: recommendation based on order responses. In *Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining.* 583–588.

[31] Batya Kenig and Benny Kimelfeld. 2019. Approximate inference of outcomes in probabilistic elections. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, Vol. 33. 2061–2068.

[32] Dušan Knop, Martin Koutecký, and Matthias Mnich. 2020. Combinatorial n-fold integer programming and applications. *Mathematical Programming* 184, 1 (2020), 1–34.

[33] Kathrin Konczak and Jérôme Lang. 2005. Voting procedures with incomplete preferences. In *Proceedings of IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling.* 130–135.

[34] Jérôme Lang. 2020. Collective decision making under incomplete knowledge: possible and necessary solutions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence*. 4885–4891.

[35] Andrew Lin. 2011. The complexity of manipulating κ-approval elections. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence*, Vol. 2. 212–218.

[36] Nicholas Mattei, Judy Goldsmith, Andrew Klapper, and Martin Mundhenk. 2015. On the complexity of bribery and manipulation in tournaments with uncertain information. *Journal of Applied Logic* 13, 4 (2015), 557–581.

[37] Fabian Schuh and Daniel Larimer. 2017. Bitshares 2.0: general overview. *accessed June-2017.[Online]. Available: http://docs. bitshares. org/downloads/bitshares-general. pdf* (2017).

[38] Hossein Azari Soufiani, David C. Parkes, and Lirong Xia. 2012. Random utility theory for social choice. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*. 126–134.

[39] Zoi Terzopoulou and Ulle Endriss. 2019. Aggregating incomplete pairwise preferences by weight. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 595–601.

[40] Toby Walsh and Lirong Xia. 2012. Lot-based voting rules. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*. 603–610.

[41] Krzysztof Wojtas and Piotr Faliszewski. 2012. Possible winners in noisy elections. In *Proceedings of 26th the AAAI Conference on Artificial Intelligence*. 1499–1505.

[42] Lirong Xia and Vincent Conitzer. 2011. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research* 41 (2011), 25–67.

[43] Lirong Xia, Jérôme Lang, and Jérôme Monnot. 2011. Possible winners when new alternatives join: New results coming up!. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*. 829–836.