The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

Approximation and Hardness of Shift-Bribery

Piotr Faliszewski
AGH University
Krakow, Poland
faliszew@agh.edu.pl

Pasin Manurangsi
UC Berkeley, USA
pasin@berkeley.edu

Krzysztof Sornat
University of Wrocław, Poland and
IDSIA, USI-SUPSI, Switzerland
krzysztof.sornat@cs.uni.wroc.pl

Abstract

In the SHIFT-BRIBERY problem we are given an election, a preferred candidate, and the costs of shifting this preferred candidate up the voters' preference orders. The goal is to find such a set of shifts that ensures that the preferred candidate wins the election. We give the first polynomial-time approximation scheme for the case of positional scoring rules, and for the Copeland rule we show strong inapproximability results.

1 Introduction

We provide approximation algorithms and inapproximability results for the SHIFT-BRIBERY problem, introduced by Elkind et al. (2009) to capture the idea of campaigning in elections. Briefly put, we are given an election where each voter ranks the candidates from the most to the least appealing one, and our goal is to ensure that a given preferred candidate becomes the winner. To this end, we can shift this candidate up within the voters' preference orders, but each such shift comes with a price (which, for example, measures the difficulty of convincing the voter that our candidate is more appealing than the voter originally thought). Naturally, we are interested in finding as cheap a solution as possible.

In fact, bribery problems also have a number of applications beyond campaign management and voting (see, e.g., the works on the margin of victory problem (Magrino, Rivest, and Shen 2011; Cary 2011; Xia 2012) and on measuring candidate success (Faliszewski, Skowron, and Talmon 2017); see also the survey of Faliszewski and Rothe (2016)). For example, a Formula 1 season consists of about 20 races, where each race can be seen as a voter ranking the candidates (the drivers) in the order in which they finished the race. For each finishing position, there is an associated number of points and the driver who collects most points becomes the world champion (i.e., this "election" uses a *positional scoring rule* as a voting rule). We can use the SHIFT-BRIBERY problem to measure how close each driver was to winning the world championship. For example, we can set the price for shifting a driver up by some t positions in a given race to be the difference between the finishing times of the driver and whoever ranked t positions higher. Then, the cheapest shift bribery corresponds to the smallest speed-up that the driver needed to become the world champion. As argued by Faliszewski et al. (2017), such values can be far more informative than the score differences between the drivers. Bribery problems also appear in the contexts of lobbying (Binkele-Raible et al. 2007; 2014), rating systems (Grandi, Stewart, and Turrini 2018), or in combinatorial domains (Baumeister et al. 2015).

With the exception of a few simple voting rules, such as the k-Approval family of rules and the Bucklin rule, SHIFT-BRIBERY tends to be NP-hard (see the works of Elkind et al. (2009) and Schlotter et al. (2017)). Indeed, this is the case, e.g., for Borda, Copeland, Maximin (Elkind, Faliszewski, and Slinko 2009) and various elimination-based rules (Maushagen et al. 2018). Yet, in many cases it can be solved quite effectively. For example, for the case of Borda there is a polynomial-time 2-approximation algorithm of Elkind et al. (2009; 2010) and several FPT algorithms of Bredereck et al. (2016a). On the other hand, for the case of Copeland, SHIFT-BRIBERY is W[1]-hard for many natural parameters (Bredereck et al. 2016a)[1] and the best known polynomial-time approximation algorithm has linear approximation ratio (Elkind and Faliszewski 2010).

In fact, the difference between the Borda rule (and, in general, the positional scoring rules) and the Copeland rule is even more striking. We show that the former can be solved nearly perfectly in polynomial time, whereas for the latter we give strong inapproximability results:

1. Our main contribution is the first polynomial-time approximation scheme (PTAS) for SHIFT-BRIBERY for positional scoring rules (in fact, our algorithm works even for the case where the scoring vectors are different for different voters). Our algorithm uses linear programming and, in particular, basic solutions of linear programs. For the case of unit prices (i.e., for the case where each unit shift has the same cost) we even obtain an EPTAS, i.e., a PTAS for which the non-polynomial factors in the running time depend on the approximation ratio only. We also show a simple combinatorial PTAS for this case.
2. For the case of the Copeland rule, we give a reduction that preserves approximation ratios up to some polynomial from the DENSEST k-SUBGRAPH (DkS) problem to

[1] One notable exception is the parametrization by the number of candidates (Knop, Koutecký, and Mnich 2017).

SHIFT-BRIBERY. Since it is generally believed that Densest k-Subgraph is hard to approximate up to a polynomial factor (Bhaskara et al. 2012; Manurangsi 2017), the same beliefs transfer to the case of Copeland-SHIFT-BRIBERY. In particular, this gives an almost-polynomial ratio hardness of approximating Copeland-SHIFT-BRIBERY under the ETH and Gap-ETH assumptions. We also show that under Gap-ETH, Copeland-SHIFT-BRIBERY does not admit an FPT approximation scheme for the parametrization by the number of unit shifts, even for the case of unit prices, whereas for parameterizations by the number of voters and by the number of candidates such approximation schemes are known to exist (Bredereck et al. 2016a).

Together with the results of Elkind et al. (2009; 2010) and Bredereck et al. (2016a), our work gives a nearly complete view of the complexity and approximability of SHIFT-BRIBERY for positional scoring rules and the Copeland rule. We omit some of the proofs due to space restrictions, but all of them are available upon request.

2 Preliminaries

For each positive integer $r \in \mathbb{N}$, we write $[r]$ to denote the set $\{1, \dots, r\}$, and by $[0]$ we mean the empty set. For an event X, we write $\mathbb{1}[X]$ to denote the indicator function such that $\mathbb{1}[X] = 1$ if X occurs and $\mathbb{1}[X] = 0$ otherwise.

Elections. An *election* $E = (C, V, \{\succ_v\}_{v \in V})$ consists of a set C of m candidates, a set V of n voters, and the collection $\{\succ_v\}_{v \in V}$ of the voters' preference orders. For each voter v, preference order $\succ_v$ gives v's ranking of the candidates from the most to the least desirable one. For a preference order $\succ_v$, we write $\pi_v \colon [m] \to C$ to denote a function such that $\pi_v(1) \succ_v \pi_v(2) \succ_v \cdots \succ_v \pi_v(m)$ (in other words, $\pi_v(i)$ is the candidate that v ranks on the i-th position). Depending on the context, we either specify voters' preference orders directly or via the π functions.

Given two candidates $c, c' \in C$, we write $V_{c \succ c'}$ to denote the set of all voters $v \in V$ that prefer c over c'.

Voting Rules. A *voting rule* $\mathcal{R}$ is a function that for each election $E = (C, V, \{\succ_v\}_{v \in V})$ outputs the set $\mathcal{R}(E) \subseteq C$ of this election's tied winners. We focus on the class of positional scoring rules and on the Copeland rule.

Consider a setting with m candidates. Under a *positional scoring rule* $\mathcal{R}_{\mathbf{w}}$, we have a vector $\mathbf{w} = (w_1, \dots, w_m) \in \mathbb{R}^m$ of point values associated with the candidate positions in the preference orders. Each voter gives each candidate the number of points associated with this candidate's position, and the candidates with the highest total score are the winners. For example, the Plurality rule uses vectors of the form $(1, 0, \dots, 0)$, the k-Approval rule uses vectors with k ones followed by $m - k$ zeros, and the Borda rule uses vectors of the form $(m - 1, \dots, 1, 0)$.

Given an election $E = (C, V, \{\succ_v\}_{v \in V})$, we sometimes speak of a positional scoring rule $\mathcal{R}_{(\mathbf{w}^v)_{v \in V}}$, where each voter has a separate scoring vector $\mathbf{w}^v = (w_1^v, \dots, w_m^v)$. This is particularly useful, for example, to model weighted elections, where each voter v has a positive integer weight ω_v and is treated as ω_v copies of a unit-weight voter; then, instead of using some rule $\mathcal{R}_{\mathbf{w}}$ and incorporating weights directly into our algorithms, we can use rule $\mathcal{R}_{(\omega_v \cdot \mathbf{w})_{v \in V}}$. As an added benefit, our algorithms become more general. We will sometimes use Δw_ℓ^v as a shorthand for $w_\ell^v - w_{\ell+1}^v$.

The Copeland rule is based on the idea of pairwise elections among the candidates. Let E be an election and let c, c' be two candidates. By $N_E(c, c')$ we mean the number of voters who prefer c over c'. We say that a candidate c *wins pairwise election* against c' if $N_E(c, c') > N_E(c', c)$. Similarly, we say that c ties (resp. loses) pairwise election against c' if $N_E(c, c') = N_E(c', c)$ (resp. $N_E(c, c') < N_E(c', c)$).

For $\alpha \in [0, 1]$, the *Copeland*$^\alpha$ rule assigns to each candidate c one point for each candidate with whom c wins a pairwise election, and α points for each candidate with whom c ties. Formally, each candidate c receives $|\{c' \in C \setminus \{c\} : N_E(c, c') > N_E(c', c)\}| + \alpha|\{c' \in C \setminus \{c\} : N_E(c, c') = N_E(c', c)\}|$ points. The winners are all the candidates with the maximum score.

For a voting rule $\mathcal{R}$, we write $\mathrm{sc}_{E,\mathcal{R}}(c)$ to denote the score that candidate c receives in election E. We sometimes drop the subscript $\mathcal{R}$ when it is clear from the context.

Shift-Bribery. A SHIFT-BRIBERY instance $I = (E, p, \psi)$ consists of an election $E = (C, V, \{\succ_v\}_{v \in V})$, a preferred candidate $p \in C$, and a collection $\psi = \{\psi_v\}_{v \in V}$ of the voters' price functions. Each voter v has the price function $\psi_v \colon \{0\} \cup [\pi_v^{-1}(p) - 1] \to \mathbb{R}_0^+ \cup \{\infty\}$ and $\psi_v(t)$ specifies the cost of shifting the preferred candidate forward by t positions in v's preference order. We require that $\psi_v(0) = 0$ and that the function is nondecreasing ($\psi_v(0) \leqslant \psi_v(1) \leqslant \cdots \leqslant \psi_v(\pi_v^{-1}(p) - 1)$). If $\psi_v(t) = \infty$ for some voter v and value t, then it is impossible to shift the preferred candidate by t or more positions in the preference order of v. For an instance I, by $\psi^{\max}(I)$ we denote the highest non-infinity price that occurs within I. We write $\Delta\psi_v(\ell)$ and $|I|$ as shorthands for $\psi_v(\ell) - \psi_v(\ell - 1)$ and mn, respectively.

A *shift action* $\mathbf{s} = (s_v)_{v \in V}$ for an instance $I = (E, p, \psi)$ of $\mathcal{R}$-SHIFT-BRIBERY is a vector of non-negative integers such that for each voter v we have $s_v < \pi_v^{-1}(p)$. Intuitively, this vector specifies for each voter by how many positions we should shift the preferred candidate. We say that shift action $\mathbf{s}$ consists of $\sum_{v \in V} s_v$ *unit shifts* and we define its cost to be $\mathrm{cost}_I(\mathbf{s}) = \sum_{v \in V} \psi_v(s_v)$. We denote the election that results from applying $\mathbf{s}$ to E by $\mathrm{shift}(E, \mathbf{s})$.

Let $\mathcal{R}$ be a voting rule and let I be a SHIFT-BRIBERY instance with election E and preferred candidate p. Shift action $\mathbf{s}$ is *successful* for I under $\mathcal{R}$ if p is an $\mathcal{R}$-winner in $\mathrm{shift}(E, \mathbf{s})$, i.e., if $p \in \mathcal{R}(\mathrm{shift}(E, \mathbf{s}))$. $\mathcal{R}$-SHIFT-BRIBERY is an optimization problem where, given a SHIFT-BRIBERY instance I, we ask for a successful shift action with the lowest cost. We write $\mathrm{OPT}(I)$ to denote this lowest cost.

Special Price Functions. There are two particularly interesting families of price functions. A *unit price* function defines the cost of each unit shift to be one, i.e., if ψ_v is a unit price function then $\psi_v(\ell) = \ell$ for each legal shift value ℓ. An *all-or-nothing price* function is such that the cost of shifting the preferred candidate is the same, irrespective by how many positions we shift him or her. Formally, if ψ_v is an all-or-nothing price function then there is a value c_v

such that $\psi_v(\ell) = c_v$ for all positive integers ℓ that represent legal shifts (and, of course, $\psi_v(0) = 0$). An instance $I = (E, p, \psi)$ has $(1, \infty)$-all-or-nothing prices if it has all-or-nothing price functions and for each voter v the value c_v is in $\{1, \infty\}$. Given such an instance I, we define its *width* to be the maximum of $\pi_v^{-1}(p) - 1$ over all $v \in V$ such that $c_v = 1$. In other words, it is the maximum number of unit shifts possible to perform within a single vote by paying a unit of price.

Linear Programming. problem we are given an $m \times n$ matrix A, an m-dimensional column vector b, an n-dimensional column vector c, and we ask for an n dimensional column vector x that minimizes the value $c^T x$ subject to the condition that $Ax \geqslant b$. A *basic solution* to such a problem is a solution $x \in \mathbb{R}^n$ such that there are n linearly independent rows a_i of A with $a_i x = b_i$. It is known that when $\{x \in \mathbb{R}^n : Ax \geqslant b\}$ is bounded, there always is a basic solution that achieves the optimum, and it can be computed up to an arbitrary error in polynomial time (see, e.g., (Lau, Ravi, and Singh 2011) for the use of basic solutions in approximation algorithms).

3 Borda Rule

We now move on to our results. We first show approximation schemes for the case of the Borda rule, mostly focusing on the case of unit prices. We start with Borda because it is one of the simplest rules, for which we can present our ideas most clearly, and because it is a very practical rule (in particular, relevant to various competitions).

Initial Observations. We first define two values that will guide our algorithms, and we explain their usefulness.

Definition 1. *For an instance $I = (E, p, \psi)$ of Borda-*SHIFT-BRIBERY *and a non-negative integer k, we define:*

$$\textit{max-diff}(I) = \max_{c \in C}(\mathrm{sc}_E(c) - \mathrm{sc}_E(p)), \text{ and}$$

$$\textit{sum-diff}(I, k) = \sum_{c \in C} \max\{0, \mathrm{sc}_E(c) - \mathrm{sc}_E(p) - k\}.$$

The former value gives the score difference between the preferred candidate and his or her strongest opponent, whereas the latter measures the total number of points that the non-preferred candidates need to lose, provided that the preferred one gains k points.

Elkind et al. (2009) note that given an instance I of Borda-SHIFT-BRIBERY, if K is the smallest number of unit shifts in an optimal solution, then $\textit{max-diff}(I)/2 \leqslant K \leqslant \textit{max-diff}(I)$. Indeed, if the preferred candidate gains $\textit{max-diff}(I)$ points then he or she certainly matches his or her strongest opponent. On the other hand, the preferred candidate needs at least $\textit{max-diff}(I)/2$ unit shifts because each of them decreases the score difference between him or her and the strongest opponent by at most two. However, it turns out that $\textit{sum-diff}(I, k)$ provides an even more useful bound.

Lemma 2. *Let $I = (E, p, \psi)$ be an instance of Borda-*SHIFT-BRIBERY*, let $\mathbf{s}$ be a successful shift action for I, and let $k_{\mathbf{s}}$ be the number of unit shifts within $\mathbf{s}$. Then, $\textit{sum-diff}(I, k_{\mathbf{s}}) \leqslant k_{\mathbf{s}}$.*

Proof. After applying $\mathbf{s}$, the score of p is exactly $\mathrm{sc}_E(p) + k_{\mathbf{s}}$. Thus each candidate $c \in C$ must have lost at least $\max\{0, \mathrm{sc}_E(c) - \mathrm{sc}_E(p) - k_{\mathbf{s}}\}$ points. This indeed means that $k_{\mathbf{s}} \geqslant \sum_{c \in C} \max\{0, \mathrm{sc}_E(c) - \mathrm{sc}_E(p) - k_{\mathbf{s}}\} = \textit{sum-diff}(I, k_{\mathbf{s}})$, as desired. □

We will also make use of the following subroutine, which is based on a simple dynamic program.

Lemma 3. *There exists an algorithm that given an instance $I = (E, p, \psi)$ of Borda-*SHIFT-BRIBERY*, a subset $C' = \{c_1, \ldots, c_t\}$ of t candidates from E, and a vector $(s_1, \ldots, s_t)$ of non-negative integers, computes a minimum cost shift action that ensures that each candidate c_i loses at least s_i points. The algorithm runs in polynomial time with respect to $|I| + \prod_{i=1}^{t}(s_i + 1)$.*

A Combinatorial PTAS for Unit Prices. We now give a simple combinatorial PTAS for Borda-SHIFT-BRIBERY with unit prices. The main idea of our algorithm is as follows. If the optimal number of unit shifts needed is OPT, then, in total, the scores of other candidates decrease by at most OPT. This means that, once we guess OPT correctly, for each $\varepsilon > 0$ there can be at most $1/\varepsilon$ "bad" candidates, whose scores exceed that of the preferred candidate by more than $(1+\varepsilon)\,\mathrm{OPT}$. Since there are only $1/\varepsilon$ such candidates, we can use the algorithm from Lemma 3 to compute the cheapest set of (at most) OPT unit shifts that ensure that the preferred candidate defeats these candidates. Then, we shift the preferred candidate up further $\varepsilon\,\mathrm{OPT}$ times, which ensures that p also defeats all the other candidates. In total, we use only $(1+\varepsilon)\,\mathrm{OPT}$ shifts and hence we arrive at our PTAS for the unit prices case. This idea is formalized below.

Theorem 4. *For each $\varepsilon > 0$, there exists an algorithm that given an instance I of Borda-*SHIFT-BRIBERY *with unit prices runs in time $\mathrm{OPT}(I)^{O(1/\varepsilon)}\textit{poly}(|I|)$ and outputs a successful shift action of cost at most $(1+\varepsilon)\,\mathrm{OPT}(I)$.*

Proof. Let $I = (E, p, \psi)$ be an instance of Borda-SHIFT-BRIBERY and let $\varepsilon > 0$ be the desired approximation ratio. For every k between $\textit{max-diff}(I)/2$ and $\textit{max-diff}(I)$, such that $\textit{sum-diff}(I, k) \leqslant k$, we execute the following steps:

1. Let $C^k_{\mathrm{BAD}} = \{c_1, \ldots, c_{t(k)}\}$ be the set of candidates whose scores are greater than $\mathrm{sc}_E(p) + (1+\varepsilon)k$. We use the algorithm from Lemma 3 to find the least-cost shift action that decreases the score of each $c_i \in C^k_{\mathrm{BAD}}$ by at least $\mathrm{sc}_E(c_i) - \mathrm{sc}_E(p) - k$ points.
2. If the cost of this shift action is at most k, then we perform additional arbitrary unit shifts so that the total number of unit shifts is $\lfloor(1+\varepsilon)k\rfloor$ or, if not enough unit shifts are possible, we shift p to the top of every vote. We output all the performed unit shifts and terminate.

We first note that the algorithm indeed outputs a successful shift action. If p ends up being on top of all the votes then he or she clearly wins. On the other hand, if the total number of unit shifts performed is $\lfloor(1+\varepsilon)k\rfloor$, then the score of p is at least $\mathrm{sc}_E(p) + \lfloor(1+\varepsilon)k\rfloor$; this means that, for all the candidates $c \notin C^k_{\mathrm{BAD}}$, the new score of p is at least $\mathrm{sc}_E(c)$, which is at least as large as the score of c after the shifts.

Moreover, the algorithm from Lemma 3 ensures that after the shifts the score of p is at least as high as the scores of all the candidates from C^k_{BAD}. Thus, p is a winner.

Next, let us argue that the algorithm computes a $(1+\varepsilon)$-approximate solution. Recall that due to the results of Elkind et al. (2009), the number of unit shifts in the optimal solution is between $\mathit{max\text{-}diff}(I)/2$ and $\mathit{max\text{-}diff}(I)$. Therefore the algorithm must terminate at latest when considering $k = \mathrm{OPT}(I)$. Given this many shifts, it is—by definition—possible for p to obtain score higher than all the candidates from C^k_{BAD} and, so, the algorithm from Lemma 3 returns a shift action with at most $\mathrm{OPT}(I)$ unit shifts. Thus the algorithm terminates with at most $\lfloor (1+\varepsilon)\,\mathrm{OPT}(I) \rfloor$ unit shifts.

The running time of the algorithm follows from Lemma 3 and is bounded by a polynomial in $|I|$ and $\Pi_{c_i \in C^k_{\text{BAD}}}(\mathrm{sc}_E(c_i) - \mathrm{sc}_E(p) - k + 1) \leqslant (\mathit{max\text{-}diff}(I) + 1)^{t(k)}$. However, we only invoke Lemma 3 when $k \geqslant \mathit{sum\text{-}diff}(I,k)$. This means that:

$$k \geqslant \mathit{sum\text{-}diff}(I,k) = \sum_{c \in C} \max\{0, \mathrm{sc}_E(c) - \mathrm{sc}_E(p) - k\}$$
$$\geqslant \sum_{c_i \in C^k_{\text{BAD}}} (\mathrm{sc}_E(c_i) - \mathrm{sc}_E(p) - k) > t(k)\varepsilon k,$$

and we conclude that $t(k) < 1/\varepsilon$. Thus the running time is polynomial with respect to $|I| + (1 + \mathit{max\text{-}diff}(I))^{1/\varepsilon}$. □

EPTAS for Unit Prices. The main result of this subsection is an EPTAS (efficient polynomial-time approximation scheme) for Borda-SHIFT-BRIBERY with unit prices, that is, a PTAS for which the non-polynomial factors in the running time depend only on the required approximation ratio. Note that in the algorithm from Theorem 4 this factor was $\mathrm{OPT}(I)^{O(1/\varepsilon)}$ and, thus, did not depend on ε alone.

On the technical level, we first develop an algorithm that for an instance I of Borda-SHIFT-BRIBERY with unit prices outputs a solution with cost at most $\mathrm{OPT}(I) + \sqrt{\mathrm{OPT}(I)}$.

Lemma 5. *There is a polynomial-time algorithm that, given an instance I of Borda-*SHIFT-BRIBERY *with unit prices, outputs a successful shift action with cost at most* $\mathrm{OPT}(I) + \sqrt{\mathrm{OPT}(I)}$.

Using this algorithm and the combinatorial PTAS from Theorem 4, we obtain our EPTAS (in short, if $\mathit{max\text{-}diff}(I) < 2/\varepsilon^2$ then we run the algorithm from Theorem 4 and we run the algorithm from Lemma 5 otherwise).

Theorem 6. *There is an algorithm that, given an instance I of Borda-*SHIFT-BRIBERY *with unit prices and a positive number $\varepsilon > 0$, runs in time $2^{O(\log(1/\varepsilon)/\varepsilon)}\mathit{poly}(|I|)$ and outputs a successful shift action of cost at most* $(1+\varepsilon)\,\mathrm{OPT}(I)$.

Theorem 6 gives formal evidence that approximating Borda-SHIFT-BRIBERY is computationally easier for the case of unit prices than for the general case or, even, for the all-or-nothing prices case. The latter cases are W[1]-hard when parameterized by the budget (Bredereck et al. 2016a) and this means that no EPTAS exists for them unless $\mathrm{W}[1] = \mathrm{FPT}$. If there were an EPTAS for Borda-SHIFT-BRIBERY for the case of general prices, or for the all-or-nothing prices, which ran in time $f(\varepsilon)n^{O(1)}$, then we could plug in $\varepsilon = 1/(2B)$, where B would be the budget limit, and solve the problem exactly in time $f(\frac{1}{2B})n^{O(1)}$, implying that $\mathrm{W}[1] = \mathrm{FPT}$. Such connections between FPT algorithms and EPTASes are well-known in theoretical computer science (Cesati and Trevisan 1997), but, so far, have not found many applications in computational social choice.

$$\text{minimize} \sum_{v \in V} \sum_{j \in [\pi_v^{-1}(p)-1]} x_{(v,j)} \quad \text{s.t.:}$$
$$0 \leqslant x_{(v,1)} \leqslant \cdots \leqslant x_{(v,\pi_v^{-1}(p)-1)} \leqslant 1 \qquad ,\forall v \in V, \tag{1}$$
$$\sum_{v \in V_{c \succ p}} x_{(v,\pi_v^{-1}(c))} \geqslant \mathrm{sc}_E(c) - \mathrm{sc}_E(p) - k, \forall c \in C^k_{\text{BAD}}. \tag{2}$$

Figure 1: Program LP-U(I,k) from the proof of Lemma 5. For each voter v, we have variables $x_{(v,1)}$, $\ldots, x_{(v,\pi_v^{-1}(p)-1)}$. Constraints (1) ensure that an integral solution describes a valid shift action and Constraints (2) ensure that after applying such an action, each candidate in C^k_{BAD} has score no higher than p; recall that $V_{c \succ p}$ means the set of voters that prefer c to p. The optimization goal is to minimize the number of unit shifts in the shift action.

Let us now move on to the proof of Lemma 5. The general structure of our algorithm is similar to that of the algorithm from Theorem 4, but instead of invoking Lemma 3, we solve a linear program. We form this program in such a way that its basic solution has to consist almost entirely of integral values. Then, rounding and complementing the obtained shift action with arbitrary unit shifts gives the desired solution.

To state our linear program, we will model shift actions as boolean matrices $(x_{(v,j)})_{v \in V, j \in [m]}$ such that $x_{(v,j)}$ is 1 if after applying the shift action the preferred candidate is ranked on position j or better in vote v, and it is 0 otherwise (so we will always have $0 \leqslant x_{(v,1)} \leqslant x_{(v,2)} \leqslant \cdots \leqslant x_{(v,m)} \leqslant 1$). Formally, a shift action $\mathbf{s} = (s_v)_{v \in V}$ corresponds to the boolean matrix $x_{(v,j)} = \mathbb{1}[\pi_v^{-1}(p) - s_v \leqslant j]$.

Proof of Lemma 5. Let $I = (E, p, \psi)$ be an instance of Borda-SHIFT-BRIBERY with unit prices, where $E = (C, V, \{\succ_v\}_{v \in V(E)})$. We try all integers k between $\mathit{max\text{-}diff}(I)/2$ and $\mathit{max\text{-}diff}(I)$, such that $\mathit{sum\text{-}diff}(I,k) \leqslant k$, and for each of them we perform the following steps:

1. We form the set $C^k_{\text{BAD}} \subseteq C$ of those candidates c whose scores exceed value $\mathrm{sc}_E(p) + k + \sqrt{k}$.
2. We form the linear program LP-U(I,k) from Figure 1 and solve it for a basic solution (note that the objective function gives the number of unit shifts used). If LP-U(I,k) is infeasible or the cost of the solution exceeds k, then we skip this value of k. Let $(x^{\mathrm{OPT}}_{(v,j)})_{v \in V, j \in [\pi_v^{-1}(p)-1]}$ be the basic optimal solution found.
3. Let $\mathbf{s}^{LP}$ be the shift action corresponding to $(\lfloor x^{\mathrm{OPT}}_{(v,j)} \rfloor)_{v \in V, j \in [\pi_v^{-1}(p)-1]}$. (Note that the rounded

solution indeed correctly describes a shift action and that its cost, i.e., the number of unit shifts it contains, is at most k.) Form a shift action $\mathbf{s}$ by extending $\mathbf{s}^{LP}$ so that it contains $k + \lfloor\sqrt{k}\rfloor$ unit shift or, if this is impossible, then so that p is on the top of every preference order. Output $\mathbf{s}$ and terminate.

Since finding basic solutions for linear programs can be done in polynomial time, the whole algorithm runs in polynomial time. Further, the cost of the computed shift action $\mathbf{s}$ is at most $\mathrm{OPT}(I) + \sqrt{\mathrm{OPT}(I)}$ unit shifts. To see this, consider the step when the algorithm tries $k = \mathrm{OPT}(I)$; if it terminates earlier then our claim certainly is satisfied. For this value of k, LP-U(I, k) certainly has a solution of cost at most k because an optimal successful shift action for I is one of its feasible solutions. Thus the algorithm terminates for this value of k and (in Step 3) outputs a shift action with at most $\mathrm{OPT}(I) + \sqrt{\mathrm{OPT}(I)}$ unit shifts.

It remains to show that the computed shift action $\mathbf{s}$ is successful. If p ends up at the top of every preference order, then surely $\mathbf{s}$ is a successful shift action. Let us consider the case where $\mathbf{s}$ consists of exactly $k + \lfloor\sqrt{k}\rfloor$ unit shifts, i.e., where after applying $\mathbf{s}$, p has score $\mathrm{sc}_E(p) + k + \lfloor\sqrt{k}\rfloor$, for the value of k for which the algorithm terminates. Since prior to applying $\mathbf{s}$ each candidate $c \notin C^k_{\text{BAD}}$ had score at most $\mathrm{sc}_E(p) + k + \lfloor\sqrt{k}\rfloor$, after applying $\mathbf{s}$ candidate p certainly has score at least as high as theirs. Thus we only need to show that each candidate in C^k_{BAD} also ends up with score at most $\mathrm{sc}_E(p) + k + \lfloor\sqrt{k}\rfloor$.

Let us say that a voter $v \in V$ is *integral* if $x^{\mathrm{OPT}}_{(v,j)} \in \{0, 1\}$ for all $j \in [\pi_v^{-1}(p) - 1]$ and let V_{int} be the set of all integral voters. Since $(x^{\mathrm{OPT}}_{(v,j)})_{v\in V, j\in[\pi_v^{-1}(p)-1]}$ is a basic solution of LP-U(I, k), at least $\sum_{v\in V}(\pi_v^{-1}(p) - 1)$ inequalities must be tight. For each integral voter v, exactly $\pi_v^{-1}(p) - 1$ inequalities of the form (1) are tight. On the other hand, for each non-integral voter v, at most $\pi_v^{-1}(p) - 2$ inequalities in (1) are tight. Further, there are $|C^k_{\text{BAD}}|$ inequalities of the form (2) and $|C^k_{\text{BAD}}| \leqslant \lfloor\sqrt{k}\rfloor$, because each candidate $c \in C^k_{\text{BAD}}$ contributes at least $\sqrt{k}$ points to *sum-diff*(I, k) and *sum-diff*$(I, k) \leqslant k$. Altogether, this means that there are at most $\lfloor\sqrt{k}\rfloor$ non-integral voters, i.e., $|V \setminus V_{\text{int}}| \leqslant \lfloor\sqrt{k}\rfloor$. Intuitively, each tight inequality of the form (2) can lead to at most a single non-integral voter.

For each $c \in C^k_{\text{BAD}}$, the score of c after applying $\mathbf{s}^{LP}$ is as follows (for a number x, $0 \leqslant x \leqslant 1$, by frac$(x)$ we mean its fractional part, so that if $0 \leqslant x < 1$ then frac$(x) = x$ and if $x = 1$ then frac$(x) = 0$):

$$\begin{aligned}
&\mathrm{sc}_E(c) - \sum_{v\in V_{c\succ p}} \left\lfloor x^{\mathrm{OPT}}_{(v,\pi_v^{-1}(c))} \right\rfloor \\
&= \mathrm{sc}_E(c) - \sum_{v\in V_{c\succ p}} x^{\mathrm{OPT}}_{(v,\pi_v^{-1}(c))} + \sum_{v\in V_{c\succ p}} \mathrm{frac}\left(x^{\mathrm{OPT}}_{(v,\pi_v^{-1}(c))}\right) \\
&\overset{(2)}{\leqslant} \mathrm{sc}_E(p) + k + \textstyle\sum_{v\in V_{c\succ p}} \mathrm{frac}\left(x^{\mathrm{OPT}}_{(v,\pi_v^{-1}(c))}\right) \\
&= \mathrm{sc}_E(p) + k + \textstyle\sum_{v\in V_{c\succ p}\setminus V_{\text{int}}} \mathrm{frac}\left(x^{\mathrm{OPT}}_{(v,\pi_v^{-1}(c))}\right) \\
&\leqslant \mathrm{sc}_E(p) + k + |V \setminus V_{\text{int}}| \leqslant \mathrm{sc}_E(p) + k + \lfloor\sqrt{k}\rfloor.
\end{aligned}$$

This means that p is indeed a winner of the election. □

Uniform All-or-Nothing Prices. In addition to the above PTASes, we devise a simple greedy algorithm for the special case. The main idea is to simply shift the preferred candidate p to the top in the votes where p is ranked lowest.

Theorem 7. *Borda-*SHIFT-BRIBERY *with uniform all-or-nothing prices is* NP*-hard and there is a greedy algorithm that gives* 1.5*-approximate solution in polynomial time.*

4 Positional Scoring Rules

In this section we give our main result: A PTAS for the case of SHIFT-BRIBERY with an arbitrary positional scoring rule, whose scoring vectors are, possibly, different for different voters, and for arbitrary prices.

Theorem 8. *There is an algorithm that given $\varepsilon > 0$ and an instance I of $\mathcal{R}$-*SHIFT-BRIBERY*, where $\mathcal{R}$ is a given positional scoring rule with a possibly different scoring vector for each voter, outputs a successful shift action for I of cost at most $(1 + \varepsilon)\,\mathrm{OPT}(I)$ and runs in time $|I|^{O(1/\varepsilon^2)}$.*

We remark that a corollary of Theorem 8 is a PTAS for Borda-SHIFT-BRIBERY (for arbitrary prices).

An Algorithm with Additive Error. The crucial part of our algorithm is an approximation algorithm that yields a good solution in the case where $\psi^{\max}(I)$, the highest non-infinite price in the instance, is small.

Lemma 9. *There is an algorithm that given $\varepsilon > 0$ and an instance I of $\mathcal{R}$-*SHIFT-BRIBERY*, where $\mathcal{R}$ is a given positional scoring rule with a possibly different scoring vector for each voter, outputs a successful shift action for I of cost at most $(1 + \varepsilon)\,\mathrm{OPT}(I) + (1 + 1/\varepsilon)\psi^{\max}(I)$ and runs in time $|I|^{O(1)}$.*

The main complication in the general prices case, as opposed to the unit prices case, is that the cost of obtaining some $k + 1$ points for the preferred candidate can be far larger than the cost of obtaining k points. Thus the main trick used in the proofs from the previous section—deciding up front how many more points than in an optimal solution the preferred candidate would get—cannot be directly applied. We work around this problem by first solving a linear program which, roughly speaking, for a given value $\varepsilon > 0$ tells us how many extra points the preferred candidate needs to ensure so that he or she has score higher than all but at most $1/\varepsilon$ candidates. Then, using a technique similar to the one we used for Lemma 5—in particular, solving a second linear program, for which a basic solution contains a large number of integral variables—we find our approximate solution.

Proof of Lemma 9. We first describe the somewhat non-intuitive algorithm, then we explain its workings and argue why it produces the desired approximate solution. Let $I = (E, p, \psi)$ be an instance of $\mathcal{R}$-SHIFT-BRIBERY, where $E = (C, V)$ is an election, p is the preferred candidate, and $\psi = \{\psi_v\}_{v\in V(E)}$ is a collection of price functions. Further,

$$\text{minimize} \sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p)-j)\cdot x_{(v,j)} \quad \text{s.t.:}$$

$$0 \leqslant x_{(v,1)} \leqslant \cdots \leqslant x_{(v,\pi_v^{-1}(p)-1)} = x_{(v,\pi_v^{-1}(p))} = \cdots = x_{(v,m)} = 1 \qquad ,\forall v \in V \tag{3}$$

$$\mathrm{sc}_E(c) - \sum_{v\in V_{c\succ p}} \Delta w^v_{\pi_v^{-1}(c)} \cdot x_{(v,\pi_v^{-1}(c))} \leqslant \mathrm{sc}_E(p) + \sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta w^v_j \cdot x_{(v,j)} \ ,\forall c \in C \tag{4}$$

Figure 2: LP1 for the proof of Lemma 9. For each voter v, we have variables $x_{(v,1)},\ldots,x_{(v,m)}$. For an integral solution, Constraints (3) ensure that the variables specify a shift action, Constraints (4) ensure that this shift action is successful, and the optimization goal specifies its cost.

$$\text{minimize} \sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p)-j)\cdot y_{(v,j)} \quad \text{s.t.:}$$

$$0 \leqslant y_{(v,1)} \leqslant \cdots \leqslant y_{(v,j_v-1)} \leqslant y_{(v,j_v)} = \cdots = y_{(v,m)} = 1 \qquad ,\forall v \in V \tag{5}$$

$$\mathrm{sc}_E(c) - \textstyle\sum_{v\in V_{c\succ p}} \Delta w^v_{\pi_v^{-1}(c)} \cdot y_{(v,\pi_v^{-1}(c))} \leqslant \mathrm{sc}_E(p) + \sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta w^v_j \cdot y_{(v,j)}, \forall c \in C_{\text{BAD}} \tag{6}$$

$$\textstyle\sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta w^v_j \cdot y_{(v,j)} \geqslant \sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta w^v_j \cdot y^*_{(v,j)} \tag{7}$$

Figure 3: LP2 for the proof of Lemma 9. For each voter v, we have variables $y_{(v,1)},\ldots,y_{(v,m)}$. For an integral solution, Constraints (5) ensure that the variables specify a shift action that pushes p at least as far as shift action $\mathbf{s}^*$ does, Constraints (6) ensure that p's score at least matches the scores of candidates in C_{BAD}, and Constraint (7) ensures that p's score is higher than the scores of candidates not in C_{BAD}.

let $\mathcal{R}$ be a positional scoring rule specified via scoring vectors $(\mathbf{w}^v)_{v\in V}$ (with one vector for each voter in V). Our algorithm proceeds as follows:

1. We solve linear program LP1 from Figure 2. Let $(x^{\mathrm{OPT}}_{(v,j)})_{v\in V,j\in[m]}$ be the computed optimal solution found for this program. Note that the value of the optimization goal for LP1 is at most $\mathrm{OPT}(I)$, because this would be the cost of an optimal integral solution.
2. For every $v \in V, j \in [m]$, we let $y^*_{(v,j)} = \min\{1,(1+\varepsilon)x^{\mathrm{OPT}}_{(v,j)}\}$, and we let $j_v \in [m]$ be the smallest index such that $y^*_{(v,j_v)} = 1$. Intuitively, the shift action $\mathbf{s}^*$ that for each voter v shifts p to position j_v is our "first order approximation" of the shift action that we will eventually produce; its cost is at most $(1+\varepsilon)\,\mathrm{OPT}(I)$ but after applying it, p's score might still be lower than that of some of the candidates. Formally, we define set C_{BAD} to contain all the candidates c such that:

$$\mathrm{sc}_E(c) - \sum_{v\in V_{c\succ p}} \Delta w^v_{\pi_v^{-1}(c)} \cdot \mathbb{1}[\pi_v^{-1}(c) \geqslant j_v] > \mathrm{sc}_E(p) + \left(\sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta w^v_j \cdot y^*_{(v,j)}\right). \tag{8}$$

On the left-hand side of the above equation, candidate c loses as many points as indicated by shift action $\mathbf{s}^*$, but the score of p, on the right-hand side, is computed with respect to the possibly fractional values $y^*_{(v,j)}$.
3. We solve linear program LP2 from Figure 3 for an optimal, basic solution $(y^{\mathrm{OPT}}_{(v,j)})_{v\in V,j\in[m]}$. We output shift action $\mathbf{s}$ that corresponds to $(\lceil y^{\mathrm{OPT}}_{(v,j)}\rceil)_{v\in V,j\in[m]}$.

The algorithm certainly runs in polynomial time. Let us now explain why the shift action that it outputs indeed ensures that p is a winner. Foremost, due to Constraints (5), shift action $\mathbf{s}$ (weakly) dominates $\mathbf{s}^*$ (i.e., for each voter it shifts p at least as far as $\mathbf{s}^*$ does). Thus, after applying $\mathbf{s}$, each opponent of p who is not in C_{BAD} has score at most as high as in the left-hand side of Equation (8). Constraint (7) ensures that p obtains at least as high a score as on the right-hand side of Equation (8) and, thus, p does not lose against any candidate not in C_{BAD}. On the other hand, Constraints (6) ensure that p does not lose against anyone in C_{BAD}.

It remains to argue that $\mathrm{cost}_I(\mathbf{s})$ is at most as required in the lemma. To this end, we first claim that $|C_{\text{BAD}}| < 1/\varepsilon$ (we omit the technical proof due to space restriction).

We now proceed to bound $\mathrm{cost}_I(\mathbf{s})$. First, observe that an optimal integral solution for LP1 has cost $\mathrm{OPT}(I)$ and, thus, for our optimal, but perhaps non-integral, solution $(x_{(v,j)})_{v\in V,j\in[m]}$ we have:

$$\textstyle\sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p)-j)x^{\mathrm{OPT}}_{(v,j)} \leqslant \mathrm{OPT}(I). \tag{9}$$

For each voter $v \in V$, we say that v is *integral* if $y^{\mathrm{OPT}}_{(v,j)} \in \{0,1\}$ for all $j \in [m]$. Let V_{int} denote the set of all integral voters. Recall that $(y^{\mathrm{OPT}}_{(v,j)})_{v\in V,j\in[m]}$ is a basic solution for LP2, meaning that at least mn linearly independent inequalities must be tight (because we have mn variables).[2] For each non-integral voter $v \notin V_{\text{int}}$, only at most $m-1$ linearly independent inequalities in (5) are tight. However, there are only $1+|C_{\text{BAD}}| < 1+1/\varepsilon$ inequalities of the form (6) and (7). From this, we can conclude that less than $1+1/\varepsilon$ voters are not integral, i.e.:

$$|V \setminus V_{\text{int}}| < 1 + 1/\varepsilon. \tag{10}$$

As a result, we have that $\mathrm{cost}_I(\mathbf{s})$ equals:

$$\textstyle\sum_{v\in V}\sum_{j\in[\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p)-j)\cdot\lceil y^{\mathrm{OPT}}_{(v,j)}\rceil$$

[2] We stress here that the inequalities must be linearly independent because in Constraint (5) we have equalities, each defined by two linearly dependent inequalities; satisfying such an equality "counts" as only one tight inequality for a basic solution.

$$= \sum_{v \in V_{\text{int}}} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot y^{\text{OPT}}_{(v,j)}$$
$$+ \sum_{v \in V \setminus V_{\text{int}}} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil.$$

Now, observe that the first summation on the right hand side is upper bounded by the optimum of LP2. Note that $(y^*_{(v,j)})_{v \in V, j \in [m]}$ is a solution to LP2. Hence, we have:

$$\begin{aligned}
&\text{cost}_I(\mathbf{s}) \leqslant \sum_{v \in V} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) y^*_{(v,j)} \\
&+ \sum_{v \in V \setminus V_{\text{int}}} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil \\
&\leqslant (1+\varepsilon) \sum_{v \in V} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot x^{\text{OPT}}_{(v,j)} \\
&+ \sum_{v \in V \setminus V_{\text{int}}} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil \\
&\overset{(9)}{\leqslant} (1+\varepsilon)\,\text{OPT}(I) \\
&+ \sum_{v \in V \setminus V_{\text{int}}} \sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil.
\end{aligned}$$

Now, observe that for each $v \in V$, if $\sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil$ is ∞, then $\text{OPT}(I)$ must also be ∞ and the inequality we try to prove is trivially true. Hence, we may assume that $\sum_{j \in [\pi_v^{-1}(p)-1]} \Delta\psi_v(\pi_v^{-1}(p) - j) \cdot \lceil y^{\text{OPT}}_{(v,j)} \rceil$ is finite; in this case, this quantity is bounded by $\psi^{\max}(I)$. As a result, we can further bound $\text{cost}_I(\mathbf{s})$ by

$$\begin{aligned}
\text{cost}_I(\mathbf{s}) &\leqslant (1+\varepsilon)\,\text{OPT}(I) + |V \setminus V_{\text{int}}| \cdot \psi^{\max}(I) \\
&\overset{(10)}{\leqslant} (1+\varepsilon)\,\text{OPT}(I) + (1+1/\varepsilon)\psi^{\max}(I),
\end{aligned}$$

which concludes our proof. □

The Final PTAS. Now we use the approximation algorithm with additive error to derive an approximation algorithm with a purely multiplicative ratio. Since the algorithm from Lemma 9 works well when $\psi^{\max}(I)$ is small, we will first "preprocess" our instance so that $\psi^{\max}(I)$ is much smaller than $\text{OPT}(I)$. To do so, note that if we consider an optimal shift action $\mathbf{s}^{\text{OPT}}$, then there are only a few voters v such that $\psi_v(s_v^{\text{OPT}})$ is large; specifically, for every $\delta > 0$, only at most $1/\delta$ voters have $\psi_v(s_v^{\text{OPT}}) \geqslant \delta\,\text{OPT}(I)$. This means that if we guess such voters and the numbers of unit shifts that we apply to them, then we can reduce the instance I to another instance I', where $\psi^{\max}(I')$ is bounded by $\delta\,\text{OPT}(I)$. We then run the algorithm from Lemma 9 on I'. By selecting $\delta = O(\varepsilon^2)$, the additive error becomes $O((\delta/\varepsilon)\,\text{OPT}(I)) = O(\varepsilon\,\text{OPT}(I))$ as intended.

5 Copeland

For the case of Copeland$^\alpha$ family of rules, we show that the SHIFT-BRIBERY problem is hard to approximate even for the unit prices and for the all-or-nothing prices. Specifically, we show that an approximation algorithm for the Copeland$^\alpha$-SHIFT-BRIBERY implies an approximation algorithm for the DENSEST-k-SUBGRAPH problem, which is believed to be hard to approximate (Bhaskara et al. 2012).

Definition 10. *In the* DENSEST-k-SUBGRAPH *(DkS) problem, we are given an undirected graph $G = (V_G, E_G)$ and a positive integer k, and the goal is to output a k-vertex subgraph of G with as many edges as possible.*

Theorem 11. *Let τ be an arbitrary non-decreasing function. If there is a polynomial time $\tau(|I|)$-approximation algorithm for Copeland$^\alpha$-SHIFT-BRIBERY for some $\alpha \in [0,1]$, for the case of unit prices or all-or-nothing prices, then there is a polynomial time $O(\tau(|V_G|^{O(1)})^2)$-approximation algorithm for the* DkS.

Although hardness of approximation of DkS within up to polynomial factor is not known, inapproximability up to almost polynomial factor is known assuming the *exponential time hypothesis* (ETH) and its gap version (Gap-ETH).[3] Specifically, Manurangsi (2017) has shown that under the ETH assumption (the Gap-ETH assumption, respectively), DENSEST-k-SUBGRAPH is hard to approximate to within a factor of $n^{1/\text{poly}(\log\log n)}$ ($n^{o(1)}$, respectively). Together with Theorem 11, this implies the following corollary.

Corollary 12. *Assuming ETH, for some constant $c > 0$ there is no polynomial-time $|I|^{1/(\log\log|I|)^c}$-approximation algorithm for Copeland$^\alpha$-SHIFT-BRIBERY for any $\alpha > 0$, even for unit prices or all-or-nothing prices. Moreover, assuming Gap-ETH, the inapproximability ratio can be improved to $|I|^{f(|I|)}$ for any function $f = o(1)$.*

For the parametrization by the number of unit shifts, assuming Gap-ETH implies that there is no FPT approximation scheme for the problem even for the case of unit prices.

Theorem 13. *Assuming Gap-ETH, for every $\alpha \in [0,1]$, every $\varepsilon > 0$, and every computable function T, there is no algorithm that given a Copeland$^\alpha$-SHIFT-BRIBERY instance I with unit prices, runs in time $T(\text{OPT}(I)) \cdot \text{poly}(|I|)$ and outputs a successful shift action with at most $(2-\varepsilon)\,\text{OPT}(I)$ unit shifts.*

We are not aware of a constant factor FPT approximation algorithms for the problem and it is possible that the factor 2 above can be improved to larger constants, or even beyond a constant. This remains an interesting open question.

6 Conclusions

We have given the first PTAS for SHIFT-BRIBERY for the case of positional scoring rules, and we have shown severe limitations regarding approximability of Copeland$^\alpha$-SHIFT-BRIBERY. We have also shown more efficient versions of our algorithms for the case of the Borda rule with unit prices. Our PTAS improves upon the 2-approximation algorithm of Elkind and Faliszewski (2010), but their algorithm is quite robust and was used, e.g., for combinatorial shift bribery (Bredereck et al. 2016b) and bribery in approval elections (Faliszewski, Skowron, and Talmon 2017). It may be possible to apply our technique in these settings as well.

Another interesting direction is to see whether our ideas can be applied to the BRIBERY problem, where the goal is to minimize the number of bribed voters but we are allowed

[3] ETH (Impagliazzo and Paturi 2001; Impagliazzo, Paturi, and Zane 2001) states that there is no subexponential time algorithm that solves 3SAT. Gap-ETH (Dinur 2016; Manurangsi and Raghavendra 2017) states that no subexponential time algorithm can distinguish between a satisfiable 3CNF formula and one which is only $(1-\delta)$-satisfiable for some absolute constant $\delta > 0$.

to change each bribed voter's preference arbitrarily. On this front, Keller et al. (2018) give a PTAS for the problem for Borda, t-approval, and, more generally, any scoring rules that satisfy a certain technical condition. It remains open whether a PTAS exists for all scoring rules.

Acknowledgments

Piotr Faliszewski was supported by AGH University statutory research grant 11.11.230.337. Pasin Manurangsi was supported by NSF Awards CCF-1813188 and CCF-1815434. Krzysztof Sornat was partially supported by the National Science Centre, Poland, grant number 2015/17/N/ST6/03684 and the SNSF Grant APPROXNET 200021_159697/1. We would like to thank the anonymous reviewers for their helpful comments.

References

Baumeister, D.; Erdélyi, G.; Erdélyi, O. J.; and Rothe, J. 2015. Complexity of Manipulation and Bribery in Judgment Aggregation for Uniform Premise-Based Quota Rules. *Math. Soc. Sci.* 76:19–30.

Bhaskara, A.; Charikar, M.; Vijayaraghavan, A.; Guruswami, V.; and Zhou, Y. 2012. Polynomial Integrality Gaps for Strong SDP Relaxations of Densest k-Subgraph. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, 388–405.

Binkele-Raible, D.; Erdélyi, G.; Fernau, H.; Goldsmith, J.; Mattei, N.; and Rothe, J. 2007. On Complexity of Lobbying in Multiple Referenda. *Review of Economic Design* 11(3):217–224.

Binkele-Raible, D.; Erdélyi, G.; Fernau, H.; Goldsmith, J.; Mattei, N.; and Rothe, J. 2014. The Complexity of Probabilistic Lobbying. *Discrete Optimization* 11:1–21.

Bredereck, R.; Chen, J.; Faliszewski, P.; Nichterlein, A.; and Niedermeier, R. 2016a. Prices Matter for the Parameterized Complexity of Shift Bribery. *Inf. Comput.* 251:140–164.

Bredereck, R.; Faliszewski, P.; Niedermeier, R.; and Talmon, N. 2016b. Large-Scale Election Campaigns: Combinatorial Shift Bribery. *J. Artif. Intell. Res.* 55:603–652.

Cary, D. 2011. Estimating the Margin of Victory for Instant-Runoff Voting. In *Proceedings of Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 2011)*.

Cesati, M., and Trevisan, L. 1997. On the Efficiency of Polynomial Time Approximation Schemes. *Inf. Process. Lett.* 64(4):165–171.

Dinur, I. 2016. Mildly Exponential Reduction from Gap 3SAT to Polynomial-Gap Label-Cover. *Electronic Colloquium on Computational Complexity (ECCC)* 23:128.

Elkind, E., and Faliszewski, P. 2010. Approximation Algorithms for Campaign Management. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE 2010)*, 473–482.

Elkind, E.; Faliszewski, P.; and Slinko, A. M. 2009. Swap Bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT 2009)*, 299–310.

Faliszewski, P., and Rothe, J. 2016. Control and Bribery in Voting. In *Handbook of Computational Social Choice*. Cambridge Univ. Press. 146–168.

Faliszewski, P.; Skowron, P.; and Talmon, N. 2017. Bribery as a Measure of Candidate Success: Complexity Results for Approval-Based Multiwinner Rules. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2017)*, 6–14.

Grandi, U.; Stewart, J.; and Turrini, P. 2018. The Complexity of Bribery in Network-Based Rating Systems. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, 1047–1054.

Impagliazzo, R., and Paturi, R. 2001. On the Complexity of k-SAT. *J. Comput. Syst. Sci.* 62(2):367–375.

Impagliazzo, R.; Paturi, R.; and Zane, F. 2001. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63(4):512–530.

Keller, O.; Hassidim, A.; and Hazon, N. 2018. Approximating Bribery in Scoring Rules. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, 1121–1129.

Knop, D.; Koutecký, M.; and Mnich, M. 2017. Voting and Bribing in Single-Exponential Time. In *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, 46:1–46:14.

Lau, L. C.; Ravi, R.; and Singh, M. 2011. *Iterative Methods in Combinatorial Optimization*. Cambridge Univ. Press, 1st edition.

Magrino, T. R.; Rivest, R. L.; and Shen, E. 2011. Computing the Margin of Victory in IRV Elections. In *Proceedings of Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 2011)*.

Manurangsi, P., and Raghavendra, P. 2017. A Birthday Repetition Theorem and Complexity of Approximating Dense CSPs. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, 78:1–78:15.

Manurangsi, P. 2017. Almost-Polynomial Ratio ETH-Hardness of Approximating Densest k-Subgraph. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, 954–961.

Maushagen, C.; Neveling, M.; Rothe, J.; and Selker, A. 2018. Complexity of Shift Bribery in Iterative Elections. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2018)*, 1567–1575.

Schlotter, I.; Faliszewski, P.; and Elkind, E. 2017. Campaign Management Under Approval-Driven Voting Rules. *Algorithmica* 77(1):84–115.

Xia, L. 2012. Computing the Margin of Victory for Various Voting Rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC 2012)*, 982–999.