Algorithms for Swap and Shift Bribery in Structured Elections

Edith Elkind
University of Oxford
Oxford, UK
elkind@cs.ox.ac.uk

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

Sushmita Gupta
National Institute of Science Education and Research
Bhubaneswer, India
sushmitagupta@niser.ac.in

Sanjukta Roy
The Institute of Mathematical Sciences, HBNI
Chennai, India
sanjukta@imsc.res.in

ABSTRACT

In computational social choice, shift bribery is the procedure of paying voters to shift the briber's preferred candidate forward in their preferences so as to make this candidate an election winner; the more general swap bribery procedure also allows one to pay voters to swap other candidates in their preferences. The complexity of swap and shift bribery is well-understood for many voting rules; typically, finding a minimum-cost bribery is computationally hard. In this paper we initiate the study of swap and shift bribery in the setting where voters' preferences are known to be single-peaked or single-crossing. We obtain polynomial-time algorithms for several variants of these problems for classic voting rules, such as Plurality, Borda and Condorcet-consistent rules.

KEYWORDS

single-peaked preferences; single-crossing preferences; shift bribery; swap bribery

ACM Reference Format:
Edith Elkind, Piotr Faliszewski, Sushmita Gupta, and Sanjukta Roy. 2020. Algorithms for Swap and Shift Bribery in Structured Elections. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Alice, Bob, Claire and Dan have applied for a research job in Professor X's lab. Alice got the position; as Claire is still on the job market, she decides to write to Professor X and ask her to provide feedback on her performance in the interview. In her response, Professor X mentions that Claire has published fewer AAMAS papers than Alice and Bob, and her rating on Mathoverflow is less impressive than Dan's. Claire then realizes that she would have to work very hard to publish more papers in AAMAS, but she can increase her Mathoverflow rating by answering simple questions. While we may question Professor X's hiring methods, Claire's approach to improve her job prospects is very mature: she realizes that she is being ranked on several criteria, and there are costs to improving her position with respect to each of them. She can then allocate her efforts to improve her chances of being selected in the future.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Computational social choice offers relevant tools for modeling Claire's reasoning. Specifically, in the Bribery family of problems, introduced by Faliszewski et al. [16], we are given an election and we ask how many votes need to be changed—and to what extent—so that a specified candidate becomes a winner (see also the survey by Faliszewski and Rothe [18]). These problems model various real-life scenarios, ranging from actual bribery, where we try to find a small set of voters to bribe[1] [2, 7, 16], through campaign-management, where we invest various resources into convincing some voters to genuinely change their minds [10, 30], to margin-of-victory and measure-of-success settings, where the cost of the bribery corresponds to the difficulty of changing the election result in a given way [19, 23, 32]; it is the *measure of success* interpretation of Bribery that motivates our work (and models Claire's behavior).

Our primary focus is the Shift Bribery problem, where we are given an election—i.e., a collection of votes that rank the candidates from the most to the least appealing one—and the briber's preferred candidate denoted by p. Our goal is to ensure that this preferred candidate becomes a winner. To achieve this goal, we can pay the voters to shift p forward in their rankings; however, we would like to minimize the total expenditure. In the important special case of unit costs, where shifting a candidate by one position in a vote costs one unit of currency, the cost of bribery indicates how well p performed in the election: the winner needs zero shifts, the second-best candidate might need a few shifts, the third-best candidate needs a few more, and so on. We also consider the more general Swap Bribery problem where, in addition to shifting his preferred candidate, the briber can swap other candidates (at a cost).

There is a number of reasons why using Shift Bribery as a measure of candidate success is appealing (e.g., as opposed to directly comparing the scores provided by the voting rule used). For example, this measure enables us to check how well a given candidate would have performed if the election was conducted using a different rule. Another advantage is that the Shift Bribery problem allows us to specify, for each voter and each positive integer s, the price of shifting the preferred candidate by s positions. This allows us to model the candidate's performance in a very fine-grained way. For example, we can view a Formula 1 season as an election, where each race is a vote that ranks the drivers in the order of completing the race; the cost of shifting a driver A by a given number of positions then corresponds to the smallest improvement in A's finishing time that would allow him to finish at a given position.

[1]The references here are meant as examples only.

Yet, using Shift Bribery also has drawbacks. One of them is that it is NP-hard for many voting rules, including rules similar to the ones that determine Formula 1 winners. This in itself is not a major issue, as this problem admits good approximation and parameterized algorithms [5, 10, 17]. More fundamentally, though, Shift Bribery disregards the internal structure that the input election may have. Since many elections—such as those in politics or in various business settings—are indeed highly structured, shift bribery costs sometimes fail to capture the candidates' true performance. Our goal is to tackle this issue.

We model structural properties of elections using the notions of single-peaked [3] and single-crossing [25, 29] preferences. Intuitively, an election is single-peaked if there is a common ordering of the candidates, referred to as the societal axis (e.g., the political left-to-right spectrum), such that for each $t > 0$ the top t candidates in each vote form an interval with respect to the axis. For example, in a single-peaked political election with the left-to-right axis, a voter would not rank an extreme left and extreme right candidates next to each other (unless they were ranked at the two bottom positions).

On the other hand, an election is single-crossing if the voters can be ordered in such a way that for every pair of candidates (c, d), as we move from one end of this order to the other, the relative ranking of c and d changes at most once. While this property may appear less intuitive, single-crossing elections do appear, e.g., when considering taxation [25, 29]. There are also natural examples of elections that are both single-peaked and single-crossing [11]; this happens, e.g., in the 1D Euclidean domain, where both candidates and voters are points on a line and the voters rank the candidates according to their Euclidean distance from them [14].

We study the complexity of Swap and Shift Bribery for elections that are either single-peaked or single-crossing, and have to retain this property after the bribery (for the same societal axis or voter order—after all, we want to maintain their structure). We obtain the following results:

(1) For single-peaked preferences, we design polynomial-time algorithms for Swap Bribery under Plurality and all rules that are Condorcet-consistent, as well as for Shift Bribery for the Borda rule (with unit costs). In contrast, in the unrestricted setting Swap Bribery is NP-hard for Copeland and Maximin (which are Condorcet-consistent) and Shift Bribery is NP-hard for Borda, even with unit costs [5].
(2) For single-crossing preferences, we design polynomial-time algorithms for Shift Bribery under Plurality, Borda, and all Condorcet-consistent rules. For the Borda rule, we are also able to provide a significantly faster variant of our algorithm for the case of unit costs.

Some proofs are deferred until the full version of the paper.

2 PRELIMINARIES

For a positive integer t, we write $[t]$ to denote the set $\{1, \dots, t\}$. We use the Iverson bracket notation, i.e., for a logical expression P, we write $[P]$ to mean 1 when P is true, and to mean 0 otherwise. We assume general familiarity with notions regarding algorithms and computational complexity theory.

Elections An *election* $E = (C, V)$ is given by a set of *candidates* $C = \{c_1, \dots, c_m\}$ and a list $V = (v_1, \dots, v_n)$ of *votes*; we refer to elements of $[n]$ as *voters*, so v_i is the vote of voter i. Votes are preference orders, ranking the candidates from the most to the least appealing one for a given voter. For a candidate c_j and vote v_i, we write $\mathrm{pos}_{v_i}(c_j)$ to denote the position of c_j in v_i: e.g., if c_j is the most preferred candidate in v_i then $\mathrm{pos}_{v_i}(c_j) = 1$, and if c_j is the least preferred one then $\mathrm{pos}_{v_i}(c_j) = m$. We write $c_j = \mathit{top}(v_i)$ if $\mathrm{pos}_{v_i}(c_j) = 1$. We write $c_j \succ_{v_i} c_k$ if voter i prefers c_j to c_k, i.e., if $\mathrm{pos}_{v_i}(c_j) < \mathrm{pos}_{v_i}(c_k)$. Sometimes instead of writing $\mathrm{pos}_{v_i}(\cdot)$ or $\succ_{v_i}$ we simply write $\mathrm{pos}_i(\cdot)$ or $\succ_i$, provided that the context is clear. Further, we sometimes specify a preference order v by writing $v\colon c_1 \succ \cdots \succ c_m$, and write $v\colon c \succ d$ to indicate that in v candidate c is ranked above candidate d.

Voting Rules A voting rule is a function $\mathcal{R}$ that given an election $E = (C, V)$ outputs a set $W \subseteq C$ of candidates that tie as winners of this election. While in practical settings some tie-breaking mechanisms would be necessary, we disregard this issue; yet, we mention that algorithmic consequences of certain tie-breaking methods can be nontrivial [26, 28]. We focus on the following two rules (in both cases the candidate(s) with the highest score win):

(1) Under the Plurality rule, each candidate c obtains a single point from each vote that ranks c in the top position.
(2) Under the Borda rule, the number of points that candidate c receives from vote v is equal to the number of candidates that are ranked below c in v.

For a given election E and candidate c, we write $\mathrm{score}_E(c)$ to denote the score of c in this election. The voting rule will always be clear from the context.

Plurality and Borda are two examples of so-called *positional scoring rules*. We are also interested in Condorcet-consistent rules. We say that a candidate $c \in C$ is a *(weak) Condorcet winner* in an election $E = (C, V)$ if for each candidate $d \in C \setminus \{c\}$ more than (at least) half of the voters prefer c to d. A voting rule $\mathcal{R}$ is *(weakly) Condorcet-consistent* if for each election with (weak) Condorcet winners, it returns exactly the Condorcet winner (exactly the set of weak Condorcet winners).

Single-Peaked Elections Let v be some vote over candidate set C and let $\lhd$ be an order over C. We refer to $\lhd$ as the *societal axis*. We say that v is *single-peaked* with respect to axis $\lhd$ if for every pair of candidates $c, d \in C$ we have:

$$\big((c \lhd d \lhd \mathit{top}(v)) \vee (\mathit{top}(v) \lhd d \lhd c)\big) \implies \big(d \succ_v c\big).$$

Equivalently, v is single-peaked with respect to $\lhd$ if for each $t \in [|C|]$ it holds that the top t candidates with respect to $\succ_v$ form a consecutive segment within $\lhd$. An election (C, V) is *single-peaked with respect to an axis* $\lhd$ if all votes in V are single-peaked with respect to $\lhd$. An election is *single-peaked* if it is single-peaked with respect to some axis [3].

Single-Crossing Elections Let $E = (C, V)$ be an election with $C = \{c_1, \dots, c_m\}$ and $V = (v_1, \dots, v_n)$. Without loss of generality, we assume that the first voter has preference order $v_1\colon c_1 \succ \cdots \succ c_m$. We say that E is *single-crossing with respect to the order* $\lhd$ *of the voters implied by* V (we refer to this order as the *natural order of the voters*) if for every pair of candidates $c, d \in C$ there is a number $t_{c,d}$ such that the set $\{i \mid c \succ_i d\}$ is either of the form $\{1, \dots, t_{c,d}\}$ or of the form $\{t_{c,d}, \dots, n\}$. Equivalently, an election is single-crossing with respect to the natural order of the voters if for every pair of candidates $c, d \in C$, as we consider voters $v_1, \dots, v_n$ from left to right, the relative ranking of c and d changes at most once [25, 29]. We define single-crossing elections with respect to a given order of the voters (i.e., not necessarily the natural one) in a similar way. An election is *single-crossing* if there exists an order of the voters with respect to which it is single-crossing.

Recognizing Restricted Elections There are well-known algorithms that given an election E recognize if it is single-peaked [1, 8, 15] or single-crossing [6, 13] and, if so, provide the societal axis or the order of the voters that witness this fact. Yet, we focus on elections with prescribed orders witnessing their nature.

3 SHIFT AND SWAP BRIBERY PROBLEMS

We primarily focus on the Shift Bribery problem, but sometimes we also discuss the more general Swap Bribery problem. Both problems were introduced by Elkind et al. [12] and in both of them we look for a way to ensure that a given preferred candidate becomes a winner, at a minimum cost.

Shift Bribery Let $\mathcal{R}$ be some voting rule. In the $\mathcal{R}$ Shift Bribery problem we are given an election $E = (C, V)$ with m candidates and n voters, a preferred candidate $p \in C$, a list $\rho = (\rho_1, \dots, \rho_n)$ of shift-bribery price functions, and a budget $B \in \mathbb{N}$. Our goal is to ensure that p is a winner of this election and, to this end, we can shift him or her forward in some of the votes. However, each such action comes with a cost specified by the price functions, and we cannot exceed the budget. Formally, if we choose to shift p by some ℓ positions forward in the preference order of voter v_i, then we have to pay $\rho_i(\ell) \in \mathbb{N}$ units of budget for this. For each price function ρ_i we require that $\rho_i(0) = 0$ (i.e., not shifting the preferred candidate does not incur any cost) and for each ℓ, $\rho_i(\ell + 1) \geq \rho_i(\ell)$ (i.e., shifting the preferred candidate further cannot be cheaper than shifting him or her closer).

Families of Price Functions It is often useful to consider special families of price functions. In particular, we consider *unit prices*, *all-or-nothing prices*, and *arbitrary prices* [5]. In the first case, each unit shift has the same unit cost. Formally, for each vote v_i and each legal number ℓ of positions by which we may want to shift the preferred candidate, we have $\rho_i(\ell) = \ell$. For the case of all-or-nothing prices, each given voter i has a constant x_i (which may depend on i, i.e., different voters may have different constants) such that $\rho_i(\ell) = x_i \cdot [\ell > 0]$. In other words, if a voter has an all-or-nothing price function, then we pay the same amount irrespective of how far we push the preferred candidate (and so, in the setting where no constraints are put on the preference orders, it makes sense to push him or her to the top of the vote). Finally, under arbitrary prices we place no restrictions on the price functions.

Swap Bribery For a given voting rule $\mathcal{R}$, the $\mathcal{R}$ Swap Bribery problem is a generalization of $\mathcal{R}$ Shift Bribery. The difference is that in the former problem not only are we allowed to shift the preferred candidate forward, but also in each vote we can perform a sequence of swaps of candidates who are adjacent at the time of the swap; for each voter the price function specifies the cost of swapping every pair of candidates.

The Complexity of Swap and Shift Bribery Shift Bribery is NP-hard for many voting rules, including Borda and many of the Condorcet-consistent rules, such as Copeland and Maximin[2] [12]. Nonetheless, the problem is in P for Plurality [12] and for several other rules, such as Bucklin [30]. Further, for positional scoring rules there is a simple 2-approximation algorithm of Elkind et al. [10, 12] and a recent, more involved PTAS [17]. There are also several FPT algorithms [5], but the problem is also W[1]-hard for some parameters. Unfortunately, Shift Bribery tends to be computationally challenging for Condorcet-consistent rules (see [5, 10, 12, 17, 30]) and for elimination-based ones [24]. Swap Bribery inherits all hardness results of Shift Bribery and has some additional ones [12]. Yet, there are some FPT algorithms even in this setting [9, 21]. Finally, we note that both Shift Bribery [20] and Swap Bribery [31] tend to be easy in the destructive setting, where the goal is to prevent some candidate from winning (however, even then these problems are NP-hard for some rules, e.g., for Copeland).

Restricted Domains We focus on the Shift Bribery problem (and, to a lesser extent, on the Swap Bribery problem) for single-peaked and single-crossing elections. To this end, we employ the following conventions (the same conventions were assumed by Brandt et al. [4], who studied a different model of bribery in single-peaked elections):

(1) Our elections always come with societal axes or voter orders witnessing their structure.
(2) The elections have to remain single-peaked/single-crossing (for the same axis/voter order) after we perform all our operations, but may violate these conditions during the process (this convention is important for single-crossing elections, but vacuous for single-peaked ones: If an election ceases to be single-peaked during a shift, then it cannot become single-peaked again with more shifting).

The justification for these conventions is as follows. First, we assume that our elections are single-peaked or single-crossing due to some publicly known reason, and should remain so. Second, the intermediate elections we produce are internal to our algorithms and are never output (indeed, in the measure-of-success interpretation of shift bribery we do not actually bribe voters, but only measure to what extent they would have to be bribed).

4 SINGLE-PEAKED ELECTIONS

In this section we focus on single-peaked elections. We start by showing that we can solve the Swap Bribery problem for Plurality and Condorcet-consistent rules in polynomial time (which also implies the result for Shift Bribery). Then we consider the Borda rule and provide an algorithm for Shift Bribery with unit costs.

4.1 Plurality and Condorcet-Consistent Rules

For Plurality, the swap bribery problem is known to be in P for arbitrary elections [12]. We show that this still holds for single-peaked elections. This result is non-trivial, because the algorithm for the general case may output an election that is not single-peaked, even if its input is.

[2]We omit definitions of the rules that we mention in passing. See, e.g., the overview of Zwicker [33] for some details.

Proposition 4.1. *Consider a vote v over a set of candidates $C = \{c_1, \dots, c_m\}$, an axis $\lhd$ given by $c_1 \lhd \dots \lhd c_m$, a candidate $p \in C$ and a swap bribery price function $\pi : C \times C \longrightarrow \mathbb{Z}$, which for every pair of candidates specifies the cost of swapping them in v. We can compute in polynomial-time a minimum-cost swap bribery that transforms v into a vote that ranks p first and is single-peaked with respect to $\lhd$.*

Proof. Given two candidates $c_i, c_j \in C$, we write $c_i \succ c_j$ if c_i is ranked above c_j in v. For each $c_i, c_j \in C$ such that $c_i \lhd p \lhd c_j$, let $C_{i,j} = \{c_i, \dots, c_j\}$ and let $\overline{C_{i,j}} = C \setminus C_{i,j}$. Further, let $A[i, j]$ be the cost of a minimum-cost swap bribery that transforms v into a vote where p is ranked first, followed by the candidates in $C_{i,j} \setminus \{p\}$, and its restriction to $C_{i,j}$ is single-peaked with respect to $c_i \lhd \dots \lhd c_j$.

We will now explain how to compute $A[i, j]$ for all $c_i, c_j \in C$ such that $c_i \lhd p \lhd c_j$. We proceed by induction on $j - i$.

If $c_i = c_j = p$, we have $A[i, j] = \sum_{c_k \in C : c_k \succ p} \pi(p, c_k)$: we need to pay for moving p into the top position, and this involves moving it past all candidates ranked above p in v.

If $c_i \lhd p$, let $A_L = A[i + 1, j] + \sum_{c_k \in \overline{C_{i,j}} : c_k \succ c_i} \pi(c_i, c_k)$. This corresponds to the cost of moving candidates in $C_{i+1,j}$ into top $j - i$ positions so that p is ranked first (captured by $A[i + 1, j]$) and then moving c_i into position $j - i + 1$ (captured by the second summand).

Similarly, if $p \lhd c_j$, we set $A_R = A[i, j-1] + \sum_{c_k \in \overline{C_{i,j}} : c_k \succ c_j} \pi(c_j, c_k)$.

Now, if we have already computed $A[i', j']$ for all i', j' such that $c_{i'} \lhd p \le c_{j'}$ and $j' - i' < j - i$, we can compute $A[i, j]$ as follows: if $c_i = p$, set $A[i, j] = A_R$; if $c_j = p$, set $A[i, j] = A_L$; otherwise set $A[i, j] = \min\{A_R, A_L\}$. The correctness of this approach follows from the fact that if a vote is single-peaked with respect to an axis $\lhd$ then for each $k > 0$ the set of candidates ranked in top k positions in that vote forms a contiguous segment with respect to $\lhd$. □

Combining Proposition 4.1 with a flow-based argument of Elkind, Faliszewski and Slinko [12], we can compute a minimum-cost swap bribery that makes a given candidate an election winner, while keeping the profile single-peaked with respect to a given axis.

Theorem 4.2. *For Plurality both* Swap *and* Shift Bribery *are in* P *for single-peaked elections.*

A more involved application of Proposition 4.1, together with the classic median voter theorem, gives the next result (Brandt et al. [4] used a similar idea for a different model of bribery).

Theorem 4.3. *Let $\mathcal{R}$ be a weakly Condorcet-consistent rule. Both $\mathcal{R}$* Swap Bribery *and $\mathcal{R}$* Shift Bribery *problems for single-peaked elections are in* P.

Proof sketch. For readability, we focus on the case where the number of voters n is odd; the case where n is even requires additional effort, but can be handled similarly. Let $t = \frac{n+1}{2}$.

We can assume without loss of generality that the voters are ordered according to the position of their top candidate on the axis: if $i < j$ then $top(v_i) = top(v_j)$ or $top(v_i) \lhd top(v_j)$. Then, by the median voter theorem the candidate $top(v_t)$ is the Condorcet winner of the election. Hence, to make p the Condorcet winner, we need to ensure that p is the top candidate of the median voter. To this end, we proceed as follows.

(1) Set $C_L = \{c \in C : c \lhd p\}$, $C_R = \{c \in C : p \lhd c\}$. We guess two integers ℓ and r such that $0 \le \ell, r < t$; our aim is to find a minimum-cost swap bribery that results in ℓ voters ranking a candidate from C_L first and r voters ranking a candidate from C_R first. Note that for any such guess the median voter ranks p first.
(2) For each $i \in [n]$, let α_i^L (respectively, α_i^R) be the minimum cost of a swap bribery that transforms v_i into a vote that is single-peaked with respect to $\lhd$ and ranks a candidate from C_L (respectively, from C_R) first, and let α_i^p be the minimum cost of a swap bribery that transforms v_i into a vote that is single-peaked with respect to $\lhd$ and ranks p first; these quantities can be computed using Proposition 4.1.
(3) Construct an instance of minimum-cost flow with source s, sink w, one node for each voter, as well as nodes L, R, and p. There is an edge of capacity 1 from the source to each voter, an edge of capacity 1 from each voter to each of L, R, and p, and edges of capacity ℓ, r and $n - \ell - r$ from L, R and p to the sink; the costs of the edges from v_i to L, R and p are set to respectively, α_i^L, α_i^R and α_i^p, and the costs of other edges are set to 0. A minimum-cost flow of size n in this network corresponds to a minimum-cost swap bribery with ℓ voters ranking a candidate from C_L first and r voters ranking a candidate from C_R first.

Each guess for the values of ℓ and r provides a candidate solution to our problem; we pick the cheapest of these. □

4.2 Shift Bribery with Unit Costs for Borda

In this section we describe a polynomial-time algorithm for shift bribery with unit costs under the Borda rule when voters' preferences are single-peaked.

Theorem 4.4. *Borda* Shift Bribery *with unit costs is in* P *for single-peaked elections.*

Setup and Initial Observations Let $E = (C, V)$ be our input single-peaked election, where $C = \{a_\ell, \dots a_1, p, b_1, \dots, b_r\}$ and $V = (v_1, \dots, v_n)$, let $p \in C$ be the preferred candidate, let $\rho = (\rho_1, \dots, \rho_n)$ be the family of shift-bribery price functions, and let B be the available budget. Without loss of generality, we assume that E is single-peaked with respect to the axis:

$$a_\ell \lhd a_{\ell-1} \lhd \dots \lhd a_1 \lhd p \lhd b_1 \lhd b_2 \lhd \dots \lhd b_r.$$

Our analysis crucially relies on the following two observations.

Observation 1. *Let v be a single-peaked vote with respect to the axis $\lhd$. If candidate p is shifted forward in v, then the resulting vote is single-peaked with respect to $\lhd$ if and only if p does not pass a candidate c such that $top(v) \lhd c \lhd p$ or $p \lhd c \lhd top(v)$.*

Based on Observation 1, for each vote v in V we identify the set of candidates that p can pass so that the resulting vote remains single-peaked. Note that this set of candidates forms a contiguous segment within v. We refer to the sequence of candidates that forms this segment (starting with the candidate ranked right ahead of p) as the *passing sequence* of v, and denote it by $PS(v)$.

Observation 2. *If two votes v and v' have the property that $|PS(v)| \ge |PS(v')|$ and $PS(v)$ and $PS(v')$ start with the same candidate, then $PS(v')$ is a prefix of $PS(v)$.*

Example 4.5. Let us consider $C = \{a_5, a_4, a_3, a_2, a_1, p, b_1, b_2\}$ and the following four votes, which are single-peaked with respect to $a_5 \lhd a_4 \lhd a_3 \lhd a_2 \lhd a_1 \lhd p \lhd b_1 \lhd b_2$:

$$v_1\colon a_2 \succ a_1 \succ a_3 \succ p \succ a_4 \succ b_1 \succ a_5 \succ b_2,$$
$$v_2\colon a_2 \succ a_1 \succ a_3 \succ a_4 \succ a_5 \succ p \succ b_1 \succ b_2,$$
$$v_3\colon a_2 \succ a_3 \succ a_1 \succ a_4 \succ a_5 \succ p \succ b_1 \succ b_2,$$
$$v_4\colon b_1 \succ b_2 \succ p \succ a_1 \succ a_2 \succ a_3 \succ a_4 \succ a_5.$$

The passing sequences for these votes are, resp., (a_3), (a_5, a_4, a_3), (a_5, a_4), and (b_2, b_1). Note that if we wanted p to pass each of a_3, a_4, and a_5 once, then we could shift p by three positions in v_2; alternatively, we can shift p by one position in v_1 and then by two positions either in v_2 or in v_3.

Algorithm for Unit Prices Our algorithm has the following structure:

(1) We guess the number w of points that p is to gain.
(2) For each candidate c we compute how many points c has to lose (i.e., how many times p has to pass c), so that p is a winner, provided p indeed obtains w additional points. Formally, for each candidate $c \in C \setminus \{p\}$, we define its *point deficit* to be
$$d_w(c) := \max\left\{\mathrm{score}_E(c) - (\mathrm{score}_E(p) + w), 0\right\}$$
(3) We compute the set of shifts ensuring that p passes each candidate the required number of times.
(4) We complement our shifts in a cheapest possible way, so that p obtains at least w points. That is, if p obtained w_1 points in the previous step, then we have to ensure that he or she obtains at least $w_2 = w - w_1$ extra points.

It is clear how to perform the first two steps. As we consider unit prices, Step 4 is easy as well: It suffices to perform arbitrary w_2 legal unit shifts. Thus we focus on Step 3.

Let $\mathcal{A} = \{a_\ell, \dots, a_1\}$ and $\mathcal{B} = \{b_1, \dots, b_r\}$. We describe how to perform Step 3 for candidates from $\mathcal{A}$ only. Due to Observation 1 (and as shown in Example 4.5), for each vote v its passing sequence either includes candidates from $\mathcal{A}$ only or candidates from $\mathcal{B}$ only. Thus Step 3 for the candidates from $\mathcal{B}$ can be executed symmetrically and independently.

To execute Step 3, we consider candidates from $\mathcal{A}$ in the order $a_1, a_2, \dots, a_\ell$. While considering candidate a_i, we repeatedly execute the following steps:

(a) If $d_w(a_i) = 0$ then we move on to the next candidate.
(b) Else, we find the votes that remain single-peaked when p is shifted ahead of a_i[3]; among all such votes, let v be the one where shifting p ahead of a_i requires the smallest number of unit shifts. We shift p right ahead of a_i in this vote and update the values $d_w(\cdot)$ for the candidates in $\mathcal{A}$ (p never passes candidates from $\mathcal{B}$ at this step).

We now argue that the above algorithm is correct, i.e., it finds a minimum-cost set of shifts that guarantees that all the point deficit values for candidates in $\mathcal{A}$ drop to zero. We first make the following observation.

[3]If there is no such vote, then we reject the current guess for w and move on to the next guess.

Observation 3. *Each time the algorithm executes Step (b), it shifts p in some vote in which p was not shifted previously.*

Let us now consider the sequence of shifts that the algorithm performs (each of them is performed in Step (b) for some candidate $a_i \in \mathcal{A}$). For the sake of contradiction, let us consider the first time that the shift executed in Step (b) prevents us from computing an optimal solution (that is, before executing this shift it is still possible to perform some shifts and obtain a minimum-cost solution, but after performing this shift it is impossible) and let a_i be the candidate considered at this point. By definition, our algorithm chose a shift by $t + 1$ positions in some vote v of the form:

$$v\colon \cdots \succ a_i \succ a_{i+1} \succ \cdots \succ a_{i+t} \succ p \succ \cdots$$

and there is no vote where we can shift p by fewer than $t + 1$ positions so that p passes a_i and the vote remains single-peaked with respect to our axis. Consider an optimal solution that is still possible prior to this shift. This solution also shifts p ahead of a_i in some vote in order to decrease its point deficit. Suppose it does so in some vote v' of the form:

$$v'\colon \cdots \succ a_i \succ a_{i+1} \succ \cdots \succ a_{i+t+t'} \succ p \succ \cdots$$

where $t' > 0$ (this holds by Observation 2; t' is positive, as otherwise the shift performed by our algorithm would not preclude it from finding an optimal solution). This shift in the optimal solution pushes p forward by exactly $t + t' + 1$ positions: otherwise, after passing a_i, p would pass candidates with zero point deficit only, and undoing these shifts would give us a cheaper solution.

We modify the optimal solution as follows: (1) We undo the last $t + 1$ shifts of p in v'; (2) We shift p forward by $t + 1$ positions in v; (3) If the optimal solution shifts p in v, then instead of doing so, we additionally shift p by the same number of positions in v'. As a result, we obtain a new optimal solution which includes the shift performed by our algorithm. This is a contradiction with our assumption that the algorithm makes an incorrect move. We conclude that our algorithm is correct. A careful implementation achieves running time $O(mn^2 \log n)$ (details omitted).

4.3 Beyond Unit Prices?

It is natural to ask if the above algorithm can be extended to more general families of price functions. Indeed, there are polynomial-time algorithms for steps (3) and (4) that can handle non-unit prices (omitted due to space restrictions and their limited applicability). However, we will now show that for general price functions our approach fails, i.e., executing steps (3) and (4) independently may fail to produce an optimal solutions.

Let us consider an election $E = (C, V)$, where $C = \{a, b, c, d, p\}$, the societal axis is $a \lhd b \lhd c \lhd p \lhd d$, and the only votes where shifting p is within budget are:

$$v_1\colon c > b > a > p > d$$
$$v_2\colon c > b > p > a > d$$
$$v_3\colon d > p > c > b > a$$

The election includes other votes as well, and, in particular, we assume that we have guessed that p is to gain 2 points, and, if this is to make p be a winner, then p has to pass b at least once.

The price functions for voters v_1, v_2, and v_3 are, respectively, ρ_1, ρ_2, and ρ_3, specified as follows (let x be some large value and let ε be much smaller than x):

$$\rho_1(i) = (x - \varepsilon) \cdot i, \qquad \rho_2(i) = x \cdot i, \qquad \rho_3(i) = (x - {}^{3\varepsilon}/_{2}) \cdot i.$$

The cheapest way for p to pass b once is to shift p by one position in v_2. This comes at price x. However, then for p to get one more point we need to shift him or her once more, and the cheapest way to achieve this is to shift p in v_3. This costs $x - {}^{3\varepsilon}/_{2}$. The total cost is $2x - {}^{3\varepsilon}/_{2}$. However, if we shift p by two positions in v_1, then p gains 2 points and passes b once, and this costs $2x - 2\varepsilon$.

As in the above example all the price functions are linear (although with different coefficients), this shows that even if we wanted to consider single-peaked elections with price functions that are only slightly more general than the unit-cost ones, we would need to develop a different algorithmic technique. Interestingly, in the next section we will see that for the single-crossing case a similar approach works for all price functions.

5 SINGLE-CROSSING ELECTIONS

For single-crossing elections we also consider Plurality, weakly Condorcet-consistent rules and the Borda rule; in this setting, we are able to derive polynomial-time algorithms for shift bribery with arbitrary costs for all rules we consider; however, the complexity of swap bribery for any of these rules (including Plurality) in single-crossing elections remains an open problem.

5.1 Plurality and Condorcet-Consistent Rules

We start by considering the Plurality rule. Our algorithm works for a somewhat more general model of shift bribery, where we can shift the preferred candidate p both forward and backward in voters' preferences; to adapt the algorithm to the standard model, we can simply set the cost of backward shifts to be $+\infty$.

Theorem 5.1. *Plurality* Shift Bribery *is in* P *for single-crossing elections.*

Proof. For an election to be single-crossing, p has to appear in the top position in a contiguous block of votes. Thus, we guess two indices $k, \ell \in [n]$, $k \leq \ell$, such that after the bribery candidate p is ranked first in the preferences of voters $k, \dots, \ell$ (and in no other votes). For a bribery to be successful, p has to be ranked first in at least one vote, so the assumption $k \leq \ell$ is without loss of generality. We also guess p's position in the first and the last vote; denote these positions by s_L and s_R, respectively. We discard a guess if $k = 1$, but $s_L \neq 1$, of if $\ell = n$, but $s_R \neq 1$. Our guess determines the Plurality score of each candidate: for p it is $\ell - k + 1$, and for every $c \in C \setminus \{p\}$ it is c's score in the original profile minus the number of votes in $v_k, \dots, v_\ell$ where c is ranked first plus the number of votes in $v_1, \dots, v_{k-1}, v_{\ell+1}, \dots, v_n$ where c is ranked second after p. We discard the guess if it does not make p the Plurality winner. For $i \in \{1, n\} \cup \{k, \dots, \ell\}$, let u_i be the vote obtained from v_i by shifting p as prescribed by our guesses.

Let D_L be the set of all candidates ranked below p in u_1 and let D_R be the set of all candidates ranked below p in u_n. Note that if there is a candidate $a \in C \setminus \{p\}$ such that $a \notin D_L$, $a \notin D_R$, we can discard this guess, as a and p cross more than once (since p is ranked first in u_k). Thus, we can assume that $D_L \cup D_R = C \setminus \{p\}$.

If $a \in D_L$ then, for the election to be single-crossing, all voters between 1 and k must rank a below p; similarly, if $a \in D_R$, all voters between ℓ and n must rank a below p. Thus, for every candidate other than p its position with respect to p on at least one side of $(u_k, \dots, u_\ell)$ is fully determined by our guesses.

Now, for each $i = \ell + 1, \dots, n$ and each $s \in [m] \setminus \{1\}$, let $c[i, s]$ be the cost on a minimum-cost shift bribery of voters $\ell + 1, \dots, i$ that results in votes of voters $\ell, \dots, i$ forming a single-crossing election where p is ranked in position s in the vote of voter i and appears above all candidates in D_R in each of these votes; we set $c[i, s] = +\infty$ if no such bribery exists (note that we do not consider $s = 1$, as this would contradict our guess for positions where p is ranked first).

For $i = \ell + 1$, computing $c[i, s]$ is straightforward: we check if shifting p into position s results in p appearing above all candidates in D_R, set $c[i, s] = +\infty$ if this is not the case, and otherwise set $c[i, s]$ to be the cost of shifting p into position s in that vote. The correctness follows from the fact that any two-vote election is single-crossing.

To compute the quantities $c[i, s]$ for $i = \ell + 2, \dots, n$, we use the following observation (Lakhani et al., [22], Lemma 11).

Lemma 5.2. *Consider a single-crossing election $(v_1, \dots, v_n)$ and a vote v. Then the election $(v_1, \dots, v_n, v)$ is single-crossing if and only if (v_1, v_n, v) is single-crossing.*

Now, suppose we have computed $c[i - 1, t]$ for all $t \in [m] \setminus \{1\}$, and we would like to compute $c[i, s]$ for some $s \in [m] \setminus \{1\}$. Let u_i be the vote obtained by shifting p into position s in the i-th vote, and let γ be the cost of this shift. Again, we set $c[i, s] = +\infty$ if in u_i candidate p is not placed above all candidates in D_R. Otherwise, for each $t \in [m] \setminus \{1\}$ let u^t be the vote obtained from v_{i-1} by shifting p into position t, and let T be the set of all values of $t \in [m] \setminus \{1\}$ such that (u_ℓ, u^t, u_i) is single-crossing; if $T = \varnothing$, we set $c[i, s] = +\infty$ and otherwise we set $c[i, s] = \min\{t \in T : c[i - 1, t] + \gamma\}$. The correctness of this calculation follows from Lemma 5.2. Eventually, we will compute $c[n, s_R]$, where s_R is the position we guessed for p in the last vote.

In the same fashion, for each $i = k-1, \dots, 1$ and each $s \in [m] \setminus \{1\}$ we compute the cost of a minimum-cost shift bribery of voters $i, \dots, k-1$ that results in the first k votes forming a single-crossing election where p is ranked in position s in the i-th vote and appears above all candidates in D_L in each of these votes; we set $c[i, s] = +\infty$ if no such bribery exists. We can compute $c[1, s_L]$ by dynamic programming, where s_L is the position we guessed for p in the first vote.

The overall cost of the minimum-cost shift bribery associated with the current guess is the sum of $c[1, s_L]$, $c[n, s_R]$, and the cost of shifting p into top position in the preferences of voters $k, \dots, \ell$. It remains to argue that this cost corresponds to a single-crossing election. To this end, consider an arbitrary candidate $a \in C \setminus \{p\}$. If $a \in D_L \cap D_R$, then the bribery associated with this cost has the property that a is ranked below p in all votes, so a and p do not cross. Otherwise, we have $a \in D_L$, $a \notin D_R$ or $a \in D_R$, $a \notin D_L$; by symmetry, we can assume the former. Then the first $k - 1$ voters are bribed in such a way that a appears below p in their votes, voters $k, \dots, \ell$ are bribed to rank p first, and the last $n - \ell$ voters are bribed in such a way that candidates a and p cross at most once. Thus, after the bribery a and p cross at most once in the overall election.

To conclude, to find a minimum-cost shift bribery that results in a single-crossing election where p wins, it suffices to consider all guesses for k, ℓ, s_L and s_R, and, for each non-discarded guess, compute the minimum-cost bribery as described above. □

Using the same proof ideas in conjunction with the median voter theorem, we can extend our result to all Condorcet-consistent rules.

Theorem 5.3. *For every weakly Condorcet-consistent rule $\mathcal{R}$, the $\mathcal{R}$ Shift Bribery problem for single-crossing elections is in* P.

5.2 The Borda Rule

For single-crossing elections we obtain a polynomial-time algorithm that works for arbitrary price functions.

Theorem 5.4. *Borda* Shift Bribery *is in* P *for single-crossing elections.*

As a consequence, the problem is also polynomial-time solvable for elections that are both single-peaked and single-crossing [11] (in the model where they are to retain both of these properties). Indeed, it suffices to modify the price functions so that shifts that break the election's single-peakedness are too expensive.

Corollary 5.5. *Borda* Shift Bribery *is in* P *for elections that are both single-peaked and single-crossing.*

5.3 Structure of the Algorithms

Our algorithm for Borda Shift Bribery in single-crossing elections have the same general structure as the unit prices algorithm for single-peaked elections. Let $E = (C, V)$ be an input single-crossing election, where $C = \{c_1, \dots, c_m\}$ and $V = (v_1, \dots, v_n)$, let $p \in C$ be the preferred candidate, let $\rho = (\rho_1, \dots, \rho_n)$ be the family of shift-bribery price functions, and let B be the available budget. We execute the same four steps as in the algorithm in Section 4.2. The first two steps are clear and we focus on Steps 3 and 4.

5.4 Performing the Necessary Shifts (Step 3)

We classify the candidates according to their positions in v_1 and v_n relative to p. That is, for each candidate $c \in C \setminus \{p\}$ we say that:

(1) candidate c is of type A if $c \succ_1 p$ and $p \succ_n c$,
(2) candidate c is of type B if $p \succ_1 c$ and $c \succ_n p$,
(3) candidate c is of type C if $c \succ_1 p$ and $c \succ_n p$,
(4) candidate c is of type D if $p \succ_1 c$ and $p \succ_n c$,

Type D candidates do not present a challenge: As the election is single-crossing (with respect to the natural ordering of the voters), all voters rank p ahead of all such candidates and, thus, all of them have scores lower than that of p. On the other hand, type C candidates are somewhat problematic. To see why this is the case, let c be some type C candidate. As all the voters rank c ahead of p, we certainly need to ensure that p passes c in some votes, but it is not clear if it would be cheaper (in total) to modify the election so that after the shifts v_1 ranks p ahead of c, or v_n ranks p ahead of c (so, in other words, it is not clear if c would eventually behave like a type B candidate or like a type A candidate). Fortunately, we can deal with this issue in a simple way. We guess by how many positions p would be shifted in the vote v_1. Given this shift, we declare each type C candidate that p passes to be a type B candidate, and each type C candidate that p does not pass to be a type A candidate (however, we do not perform this shift; its only role is to partition the type C candidates into type A candidates and type B candidates). We refer to these declarations as the *type assignment.*

Let us now consider some voter i. If we shift p forward in v_i by some number of positions, then it is quite likely that the election will no longer be single-crossing. The following lemma shows that there is a unique minimal set of shifts that restore the single-crossing property for the given type assignment.

Lemma 5.6. *We can identify in polynomial time a minimal set of shifts that restores the single-crossing property of the input election after p has been shifted forward in the preference order of voter i.*

Proof. Our algorithm proceeds as follows. If p passes some candidate c of type A in the preference order or voter i then, for the election to remain single-crossing with respect to the given order of voters, p has to be ranked ahead of c by all the voters $i+1, i+2, \dots, n$; otherwise we would have $p \succ_i c$, $c \succ_{i+1} p$, and $p \succ_n c$, which would mean that the election is no longer single-crossing (note that for candidates of type C that have been assigned type A, we do not reach a contradiction per se; rather, we would obtain inconsistency with the type assignment). Thus, for each voter in the set $\{i+1, i+2, \dots, n\}$ who does not yet rank p ahead of c, we shift p to appear right in front of c. For candidates of type B that p passes in the i-th vote we proceed in a similar way, but focusing on voters $1, \dots, i-1$.

We repeat the above reasoning for all the votes where we shift p, until the election becomes single-crossing. This has to happen because in each iteration where the election is not single-crossing, we shift p forward in at least one vote. □

Now we are ready to describe Step 3 of our algorithm. For each candidate $c \in C \setminus \{p\}$ of type A with positive point deficit (i.e., such that $d_w(c) > 0$) we proceed as follows. Let t_c be the largest number such that in the original election c is ranked ahead of p in votes $v_1, \dots, v_{t_c}$. We shift p right ahead of c in the preference order of voter $t_c - (d_w(c) - 1)$ (unless p is already ranked ahead of c in this vote due to previous shifts), and we run the algorithm from Lemma 5.6. As a consequence, this algorithm ensures that p is shifted ahead of c in the preference orders of the voters $t_c - (d_w(c) - 1), \dots, t_c - 1, t_c$ and hence passes c at least $d_w(c)$ times. It is also easy to see that this is a minimal set of shifts that ensures that p passes c this many times (for the given assignment of types).

For each candidate of type B we proceed similarly. Let c be a candidate of type B and let t_c be the smallest number such that in the original election c is ranked ahead of p in votes $t_c, \dots, n$. If $d_w(c) > 0$ then we shift p to be right ahead of c in the preference order of voter $t_c + d_w(c) - 1$ (unless p is already ranked ahead of c in this vote due to previous shifts) and we run the algorithm from Lemma 5.6. All in all, we obtain the following result.

Lemma 5.7. *There is a polynomial-time algorithm that for a given assignment of types to the candidates computes a minimal set of shifts ensuring that for each $c \in C \setminus \{p\}$, p passes c at least $d_w(c)$ times.*

The set of shifts computed by the above algorithm is minimal in the sense that every set of shifts that guarantees that p passes each

$$\begin{array}{llrr}
v_1: & \underbrace{\cdots \succ c \succ \cdots \succ}_{\text{candidates in } U} & p & \underbrace{\succ \cdots \succ d \succ \cdots}_{\text{candidates in } D} \\
\vdots & & & \\
v_{k-1}: & \cdots \succ d \succ \cdots \succ & c \succ p & \succ \cdots \\
v_k: & \cdots \succ & d \succ p & \succ \cdots \succ c \succ \cdots \\
\vdots & & &
\end{array}$$

Figure 1: Illustration for the proof of Theorem 5.8.

candidate c at least $d_w(c)$ times (for the given type assignment) includes all the shifts computed by the algorithm from Lemma 5.7.

The above-described algorithm runs in polynomial time, but is very inefficient. Indeed, it is possible to replace it with a procedure that runs in time $O(mn^2)$ (briefly, for every vote and every candidate that p passes, we update the shifts in all other votes; we omit the detailed description).

5.5 Obtaining the Missing Points (Step 4)

To implement Step 4 of our algorithm, we have to provide a procedure that given an instance of Borda Shift Bribery finds the cheapest set of shifts which can ensure that the preferred candidate obtains additional w_2 points, where w_2 is part of the input (recall that in the algorithm our goal is to obtain w points and we have already obtained w_1 points in Step 3, so now we need to obtain $w_2 = w - w_1$ extra points). Below we give a greedy procedure for the case of unit costs (the discussion of more general price functions is postponed to the full version of the paper).

First, we show that it is always possible to shift the preferred candidate p forward in some vote so that the election remains single-crossing (unless p is already ranked first by all the voters).

Theorem 5.8. *For every single-crossing election $E = (C, V)$ and each candidate $p \in C$ who is not ranked first in every vote, there is a vote v such that if we shift p forward by one position in v then the election remains single-crossing with respect to the same voter order.*

Proof. Consider an election $E = (C, V)$ with a list of votes $V = (v_1, \ldots, v_n)$, and assume without loss of generality that E is single-crossing with respect to the natural voter order. Let p be an arbitrary candidate that is not ranked first in every vote.

Note that if p is ranked first in at least one vote, then we can shift p forward in a vote that is adjacent to the one where p is ranked first without violating the single-crossing property. Thus, from now, on we assume that no voter ranks p first. Let U and D be the sets of candidates ranked above p in the first and last vote, respectively.

If each voter ranks some member of U right ahead of p then we can shift p forward in the last vote without violating the single-crossing property (the candidate ranked right before p in the last vote is also ranked ahead of p in all the other votes). Suppose then that for some $j \in [n]$ the candidate ranked right above p in v_j is some candidate $d \in D$. Let v_k be the first such vote. Thus, in votes $v_1, \ldots, v_{k-1}$ either p is ranked first (but this is impossible by our assumptions) or the candidate ranked right above p is in U.

If d is ranked below p in v_{k-1}, then d is ranked below p in each of the first $k-1$ votes, so we can swap p and d in v_k without violating the single-crossing property. Now, suppose d is ranked above p in v_{k-1}, and let c be the candidate ranked right above p in v_{k-1} (note that $c \neq d$ by our choice of k; see Figure 1 for an illustration). As $c \in U$, the first voter ranks c above d, and the $(k-1)$-th voter ranks d above c. Since our profile is single-crossing and in v_k candidate p is ranked right below d, this means that in v_k candidate c appears below p. Thus, we can swap c and p in v_{k-1} without violating the single-crossing property. This completes the proof. □

Using Theorem 5.8 we obtain a very simple algorithm for the problem of obtaining additional w_2 points for p in an instance of Borda Shift Bribery.

Proposition 5.9. *There is an algorithm that given an instance of Borda* Shift Bribery *with single-crossing preferences and unit-cost price functions, and a nonnegative integer w_2, finds a minimum-cost set of unit shifts that increases the score of the preferred candidate p by w_2 (or to the point where p is ranked first in every vote) and runs in time $O(nm)$.*

All in all, our algorithm for unit-price Borda Shift Bribery for single-crossing elections (Steps 1–4) runs in time $O(m^2n)$.

The Space of Single-Crossing Elections Let us now consider Theorem 5.8 independently from the Shift Bribery problem. By applying this theorem repeatedly, we can find a sequence of swaps that transforms a given single-crossing election into one where all the voters have identical preference orders, while ensuring that each intermediate election is single-crossing (for the same voter order). Then, by applying such a transformation “in reverse,” we can obtain any other single-crossing election (even for a different voter order). That is, the space of all single-crossing elections is connected in the sense of Obraztsova et al. [27]. It would be interesting to understand if this space is also convex in their sense (Obraztsova et al. [27] show that the space of all single-peaked elections with a given axis is connected and convex, but for the space of all single-peaked elections this is not the case).

6 CONCLUSIONS

We have shown that the complexity of Swap and Shift Bribery drops significantly when the input election is single-peaked or single-crossing and needs to retain this property post-bribery with respect to the original societal axis or voter order. Remaining open problems include complexity of Borda Shift Bribery with arbitrary prices for single-peaked elections and of Plurality Swap Bribery for single-crossing elections. Another challenge is to extend our positive results for Borda to all positional scoring rules. It would also be interesting to explore the complexity of bribery in a Euclidean setting, where the briber can pay voters and/or candidates to move in the issue space. Another exciting direction is empirical study of shift bribery (both in the general setting and for the restricted domains).

ACKNOWLEDGMENTS

This work was supported by the European Research Council (ERC) under grant number 639945 (Elkind), and by the funds of the Polish Ministry of Science and Higher Education assigned to AGH University (Faliszewski).

REFERENCES

[1] J. Bartholdi, III and M. Trick. 1986. Stable Matching with Preferences Derived from a Psychological Model. *Operations Research Letters* 5, 4 (1986), 165–169.

[2] D. Baumeister, T. Hogrebe, and L. Rey. 2019. Generalized Distance Bribery. In *Proceedings of AAAI-2019*. 1764–1771.

[3] D. Black. 1958. *The Theory of Committees and Elections*. Cambridge University Press.

[4] F. Brandt, M. Brill, E. Hemaspaandra, and L. Hemaspaandra. 2015. Bypassing Combinatorial Protections: Polynomial-Time Algorithms for Single-Peaked Electorates. *Journal of Artificial Intelligence Research* 53 (2015), 439–496.

[5] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. 2016. Prices matter for the parameterized complexity of shift bribery. *Information and Computation* 251 (2016), 140–164.

[6] R. Bredereck, J. Chen, and G. Woeginger. 2013. A Characterization of the Single-Crossing Domain. *Social Choice and Welfare* 41, 4 (2013), 989–998.

[7] L. Chen, L. Xu, S. Xu, Z. Gao, N. Shah, Y. Lu, and W. Shi. 2018. Protecting Election from Bribery: New Approach and Computational Complexity Characterization. In *Proceedings of AAMAS-2018*. 1894–1896.

[8] J. Doignon and J. Falmagne. 1994. A Polynomial Time Algorithm for Unidimensional Unfolding Representations. *Journal of Algorithms* 16, 2 (1994), 218–233.

[9] B. Dorn and I. Schlotter. 2012. Multivariate Complexity Analysis of Swap Bribery. *Algorithmica* 64, 1 (2012), 126–151.

[10] E. Elkind and P. Faliszewski. 2010. Approximation Algorithms for Campaign Management. In *Proceedings of WINE-2010*. Springer-Verlag *Lecture Notes in Computer Science #6484*, 473–482.

[11] E. Elkind, P. Faliszewski, and P. Skowron. 2020. A Characterization of the Single-Peaked Single-Crossing Domain. *Social Choice and Welfare* 54, 1 (2020), 167–187.

[12] E. Elkind, P. Faliszewski, and A. Slinko. 2009. Swap Bribery. In *Proceedings of SAGT-2009*. Springer-Verlag *Lecture Notes in Computer Science #5814*, 299–310.

[13] E. Elkind, P. Faliszewski, and A. Slinko. 2012. Clone Structures in Voters' Preferences. In *Proceedings of ACM EC-2012*. 496–513.

[14] J. Enelow and M. Hinich. 1984. *The spatial theory of voting: An introduction*. Cambridge University Press.

[15] B. Escoffier, J. Lang, and M. Öztürk. 2008. Single-Peaked Consistency and its Complexity. In *Proceedings of ECAI-2008*. 366–370.

[16] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. 2009. How Hard is Bribery in Elections? *Journal of Artificial Intelligence Research* 35 (2009), 485–532.

[17] P. Faliszewski, P. Manurangsi, and K. Sornat. 2019. Approximation and Hardness of Shift-Bribery. In *Proceedings of AAAI-2019*. 1901–1908.

[18] P. Faliszewski and J. Rothe. 2016. Control and Bribery in Voting. In *Handbook of Computational Social Choice*, F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia (Eds.). Cambridge University Press, Chapter 7.

[19] P. Faliszewski, P. Skowron, and N. Talmon. 2017. Bribery as a Measure of Candidate Success: Complexity Results for Approval-Based Multiwinner Rules. In *Proceedings of AAMAS-2017*. 6–14.

[20] A. Kaczmarczyk and P. Faliszewski. 2019. Algorithms for Destructive Shift Bribery. *Autonomous Agents and Multiagent Systems* 33, 3 (2019), 275–297.

[21] D. Knop, M. Koutecký, and M. Mnich. 2017. Voting and Bribing in Single-Exponential Time. In *Proceedings of STACS-2017*. 46:1–46:14.

[22] F. Lakhani, D. Peters, and E. Elkind. 2019. Correlating Preferences and Attributes: Nearly Single-Crossing Profiles. In *Proceedings of IJCAI-2019*. 414–420.

[23] T. Magrino, R. Rivest, E. Shen, and D. Wagner. 2011. Computing the Margin of Victory in IRV Elections. (Aug. 2011). Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections.

[24] C. Maushagen, M. Neveling, J. Rothe, and A.-K. Selker. 2018. Complexity of Shift Bribery in Iterative Elections. In *Proceedings of AAMAS-2018*. 1567–1575.

[25] J. Mirrlees. 1971. An Exploration in the Theory of Optimal Income Taxation. *Review of Economic Studies* 38 (1971), 175–208.

[26] S. Obraztsova and E. Elkind. 2011. On the Complexity of Voting Manipulation under Randomized Tie-Breaking. In *Proceedings of IJCAI-2011*. 319–324.

[27] S. Obraztsova, E. Elkind, P. Faliszewski, and A. Slinko. 2013. On Swap-Distance Geometry of Voting Rules. In *Proceedings of AAMAS-2013*. 383–390.

[28] S. Obraztsova, E. Elkind, and N. Hazon. 2011. Ties Matter: Complexity of Voting Manipulation Revisited. In *Proceedings of AAMAS-2011*. 71–78.

[29] K. Roberts. 1977. Voting over income tax schedules. *Journal of Public Economics* 8, 3 (1977), 329–340.

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

[31] D. Shiryaev, L. Yu, and E. Elkind. 2013. On Elections with Robust Winners. In *Proceedings of AAMAS-2013*. 415–422.

[32] L. Xia. 2012. Computing the Margin of Victory for Various Voting Rules. In *Proceedings of ACM EC-2012*. 982–999.

[33] W. Zwicker. 2016. Introduction to the Theory of Voting. In *Handbook of Computational Social Choice*, F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia (Eds.). Cambridge University Press, Chapter 2.