Prices Matter for the Parameterized Complexity of Shift Bribery

Robert Bredereck[1], **Jiehua Chen**[1], **Piotr Faliszewski**[2], **André Nichterlein**[1], and **Rolf Niedermeier**[1]
[1]TU Berlin, Berlin, Germany
{robert.bredereck, jiehua.chen, andre.nichterlein, rolf.niedermeier}@tu-berlin.de
[2]AGH University of Science and Technology, Krakow, Poland
faliszew@agh.edu.pl

Abstract

In the SHIFT BRIBERY problem, we are given an election (based on preference orders), a preferred candidate p, and a budget. The goal is to ensure that p wins by shifting p higher in some voters' preference orders. However, each such shift request comes at a price (depending on the voter and on the extent of the shift) and we must not exceed the given budget. We study the parameterized computational complexity of SHIFT BRIBERY with respect to a number of parameters (pertaining to the nature of the solution sought and the size of the election) and several classes of price functions. When we parameterize SHIFT BRIBERY by the number of affected voters, then for each of our voting rules (Borda, Maximin, Copeland) the problem is W[2]-hard. If, instead, we parameterize by the number of positions by which p is shifted in total, then the problem is fixed-parameter tractable for Borda and Maximin, and is W[1]-hard for Copeland. If we parameterize by the budget for the cost of shifting, then the results depend on the price function class. We also show that SHIFT BRIBERY tends to be tractable when parameterized by the number of voters, but that the results for the number of candidates are more enigmatic.

1 Introduction

Rank aggregation and winner determination are of key importance in economical and political settings. For instance, there are product rankings based on comparing prices but also based on different product tests (performed by various institutions such as foundations, journals, etc.). A sophisticated way to deal with the various results is to compute a consensus ranking using preference-based rank aggregation.[1] In order to affect the outcome of the rank aggregation one has to influence the product rankings obtained from different sources (voters). Clearly, the cost of influencing may differ from source to source.

In this work, we study the computational complexity of "bribing" the outcome of the rank aggregation at minimum cost. Moreover, replacing "bribery" with "product development" it may also be important for producers/sellers to find out how much they need to invest into their product in order to achieve a better position in the aggregated ranking or maybe even become the winner. A natural and simple model in this context, using the formalisms of voting theory, is SHIFT BRIBERY as introduced by Elkind et al. (2009). We extend their studies in terms of charting the border of computational (worst-case) tractability, herein putting particular emphasis on the voter-specific "shifting prices" (how expensive is it to shift a candidate by x positions "up").

In SHIFT BRIBERY, we are given an election, that is, a set of candidates and a list of voters, each with a linear preference order over the candidate set.[2] Our goal is to ensure that our preferred candidate p wins. To achieve this, we can approach each of the voters, one-on-one, and try to convince[3] him or her to rank p higher. Naturally, the effect (the number of positions by which p is shifted in each voter's preference order) depends on the voter's character and situation and on the amount of effort we invest into convincing the voter. This "effort" could, for example, mean the amount of time spent, the price of preparing promotional materials or, in the bribery view of the problem, the payment to the voter. Thus, the computational complexity of the problem depends on the voting rule used in the election, on various election parameters such as the number of candidates and voters, and on the type of price functions describing the efforts needed to shift p up by a given number of positions in the voters' preference orders. Our goal is to unravel the nature of these dependencies.

Elkind et al. (2009) have shown that SHIFT BRIBERY is polynomial-time solvable for Plurality and k-Approval, but that it is NP-complete for the Borda, Copeland$^\alpha$, and Maximin voting rules. In addition, they provided a 2-approximation algorithm for SHIFT BRIBERY under Borda. Elkind and Faliszewski (2010) extended this last result to all

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]The German website idealo.de aggregates different product tests by first translating the test results into a unified rating system and then taking the "average" of all ratings. It would be very interesting, however, to utilize the rankings themselves instead of the ratings for the aggregation.

[2]We assume that we have the knowledge of the voters' preference orders (for example, from preelection polls).

[3]What "to convince" means can vary a lot depending on the application scenario. On the evil side we have bribery, but it can also mean things like product development or bug elimination or lower costs for customers. Clearly, different customers may appreciate different changes, which is modeled by the voter's individual price functions.

		$\mathcal{R}$ Shift Bribery				
parameter	$\mathcal{R}$	unit prices	convex prices	arbitrary prices	sortable prices	all-or-nothing
#shifts (t)	B/M	FPT (Thm. 1)				
	C	W[1]-hard (Thm. 2)				
#voters-affected (n_a)	B/M/C	W[2]-hard (Cor. 1)				
budget (B)	B/M	FPT (Cor. 2)		W[2]-hard (Cor. 2)		
	C	W[1]-hard (Cor. 3)		W[2]-hard (Cor. 3)		
#voters (n)	B/M	FPT-AS (Thm. 3)				FPT (Prop. 2)
	C	FPT-AS (Thm. 3) / W[1]-hard (Thm. 4)				FPT (Prop. 2)
#candidates (m)	B/M/C	FPT (Dorn and Schlotter 2012)	XP (Prop. 3)		FPT-AS (Thm. 5)	

Table 1: The parameterized complexity of $\mathcal{R}$ Shift Bribery for Borda (B), Maximin (M), and Copeland$^\alpha$ (C) (for each rational number α). Roughly speaking, for a given parameter, a problem is in FPT if an efficient algorithm exists as long as the parameter is small (or an efficient approximation algorithm in the case of FPT-AS). The problem is W[1]-hard or W[2]-hard if it is intractable even if the parameter is small. Formal definitions for FPT, W[1], W[2], and XP are provided in Section 2, and in Section 3 for FPT-AS. Note that when speaking of the FPT-AS result we mean the result for the optimization version of Shift Bribery where the goal is to minimize the budget.

scoring protocols and found approximation algorithms for Shift Bribery under Copeland$^\alpha$ and Maximin. Schlotter et al. (2011) complemented these results by showing that Shift Bribery is easy for the Bucklin and Fallback rules.

Thus, for the case of Borda, Copeland$^\alpha$, and Maximin, Elkind et al. (2010; 2009) have shown that Shift Bribery has high worst-case complexity, but that one can deal with it using polynomial-time approximation algorithms. However, to better understand where the intractability of Shift Bribery really lies in different special cases, we use another approach of dealing with computationally hard problems, namely parameterized complexity analysis and, more specifically, the notion of fixed-parameter tractability and correspondingly developed exact algorithms. For instance, almost tied elections are tempting targets for campaign management. An exact algorithm which is efficient in this case may be more attractive than a general approximation algorithm. In close-to-tied elections it might suffice, for example, to contact only a few voters or, perhaps, to shift the preferred candidate by only a few positions in total. Similarly, it is important to solve the problem exactly if the campaign manager has only a small budget at his or her disposal.This is captured by using various problem parameterizations and performing a parameterized complexity analysis.

Furthermore, it is natural to expect that the computational complexity of Shift Bribery depends on the nature of the voters' price functions and, indeed, there is some evidence for this fact: For example, if we assume that shifting p by each single position in each voter's preference order has a fixed unit price or, at the very least, if functions describing the prices are *convex*, then one can verify that the 2-approximation algorithm of Elkind and Faliszewski (2010) boils down to a greedy procedure that picks the cheapest available single-position shifts until it ensures the designated candidate's victory (such an implementation would be much faster than the expensive dynamic-programming algorithm that they use, but would guarantee a 2-approximate solution for convex price functions only). On the other hand, the hardness proofs of Elkind et al. (2009) all use a very specific form of price functions which we call *all-or-nothing* prices. See Section 3 for the definitions of the different price functions that we study.

We combine the above two sets of observations and we study the parameterized complexity of Shift Bribery for Borda, Maximin, and Copeland$^\alpha$, for parameters describing the number of affected voters, the number of unit shifts, the budget, the number of candidates, and the number of voters, under price functions that are either all-or-nothing, sortable, arbitrary, convex, or have a unit price for each single shift. The three voting rules that we select are popular in different kinds of elections apart from political ones. For instance, Borda is used by the X.Org Foundation to elect its board of directors, a slightly modified version of Copeland is used to elect the Board of Trustees for the Wikimedia Foundation.

We summarize our results in Table 1, and we discuss them throughout the paper. In short, it turns out that indeed both the particular parameters used and the nature of the price functions have strong impact on the computational complexity of Shift Bribery. Three key technical contributions of our work are

1. novel FPT *approximation schemes* exploiting the parameters "number of voters" and "number of candidates" (such schemes are rare in the literature and of significant practical interest),
2. a surprising W[1]-hardness for the parameter "number of voters" when using Copeland$^\alpha$ voting, and
3. a *partial kernelization* (polynomial-time data reduction) result for the parameter "number of unit shifts".

The paper is organized as follows. In Section 2 we present preliminary notions regarding elections and parameterized complexity theory, and in Section 3 we formally define the Shift Bribery problem, together with its parameterizations and definitions of price functions. Our results are in Sections 4 (parameterization by solution cost) and 5 (parameterization by election size), and we conclude in Section 6.

2 Preliminaries

Below we provide a brief overview of the notions regarding elections and parameterized complexity theory.

Elections and Voting Rules. We use the standard, ordinal model of elections where an election $E = (C, V)$ consists of a set $C = \{c_1, \dots, c_m\}$ of candidates and a list $V = (v_1, \dots, v_n)$ of voters. Each voter v provides a preference order over C, i. e., a linear order ranking the candidates from the most preferred one (position 1) to the least preferred one (position m). For example, if $C = \{c_1, c_2, c_3\}$ then writing $v\colon c_1 \succ c_2 \succ c_3$ indicates that voter v likes c_1 best, then c_2, and then c_3. For two candidates c_i, c_j and voter v we write $v\colon c_i \succ c_j$ to indicate that v prefers c_i to c_j. Further, we write $\succ_v$ to denote voter v's preference order. We often use multiset notation to speak of a list V. For example, we write $v \in V$ to indicate that some voter v is included in the list V and write $|V|$ to denote the number of voters in V. Given an election E, for each two candidates c and d, we define $N_E(c, d)$ to be the number of voters in E who prefer c over d.

A voting rule $\mathcal{R}$ is a function that given an election E outputs a non-empty set $\emptyset \neq \mathcal{R}(E) \subseteq C$ of the tied winners of the election. Note that we use the nonunique-winner model, where each of the tied winners is considered a winner and we disregard tie-breaking. Furthermore, we implicitly assume that all voting rules that we consider are anonymous, i.e., their outcomes depend only on the numbers of voters with particular preference orders. We consider Borda, Maximin, and the Copeland$^\alpha$ family of rules. These rules assign points to every candidate and pick as winners those who get most; we write $\mathrm{score}_E(c)$ to denote the number of points candidate c receives in election E—the voting rule will always be clear from the context.

Let E be an election with m candidates. Under Borda, each candidate c receives from each voter v as many points as there are candidates that v ranks lower than c. Formally, the Borda score of candidate c is $\mathrm{score}_E(c) = \sum_{d \in C \setminus \{c\}} N_E(c, d)$. Similarly, the Maximin score of a candidate c is the number of voters who prefer c to his or her "strongest competitor", formally, $\mathrm{score}_E(c) = \min_{d \in C \setminus \{c\}} N_E(c, d)$. Under Copeland$^\alpha$ we organize a *head-to-head* contest between each two candidates c and d; if c wins (i.e., if $N_E(c, d) > N_E(d, c)$) then c receives one point, if d wins then d receives one point, and if they tie (i.e., $N_E(c, d) = N_E(d, c)$) then they both receive α points. More precisely, for each rational number $\alpha \in [0, 1]$, Copeland$^\alpha$ assigns to each candidate $c \in C$ score $|\{d \in C \setminus \{c\}\colon N_E(c, d) > N_E(d, c)\}| + \alpha\,|\{d \in C \setminus \{c\}\colon N_E(c, d) = N_E(d, c)\}|$. Typical values of α are 0, 1/2, and 1, but there are cases where other values are used. All our results for COPELAND$^\alpha$ SHIFT BRIBERY hold for arbitrary rational number α. For brevity's sake, we write "Copeland" instead of "Copeland$^\alpha$ for arbitrary rational number α" throughout the remainder of this paper.

Parameterized Complexity. To speak of parameterized complexity of a given problem, we declare a part of the input as a so-called parameter (here we consider numerical parameters only, e.g., for SHIFT BRIBERY it could be the number of candidates or the budget; see the next section). We say that a problem parameterized by k is *fixed-parameter tractable*, that is, is in FPT, if there exists an algorithm that given input I with parameter k gives a correct solution in time $f(k) \cdot |I|^{O(1)}$, where $f(k)$ is an arbitrary computable function of k, $|I|$ is the length of the encoding of I. To describe the running times of our algorithms, we often use the $O^*(\cdot)$ notation. It is a variant of the standard $O(\cdot)$ notation where polynomial factors are omitted. For example, if the algorithm's running time is $f(k) \cdot |I|^{O(1)}$, where f is superpolynomial, then we would say that it is $O^*(f(k))$.

Parameterized complexity theory also provides a hierarchy of hardness classes, starting with W[1], such that FPT $\subseteq$ W[1] $\subseteq$ W[2] $\subseteq \dots \subseteq$ XP. For problems where an FPT algorithm and a hardness proof are elusive, one can at least try to show an XP algorithm whose running time is polynomial if one treats the parameter as a constant. As opposed to FPT, the degree of the polynomial describing the "XP" running time can depend on the parameter.

We point the reader to Downey and Fellows (2013), Flum and Grohe (2006), and Niedermeier (2006) for more information on parameterized complexity. There is also a survey on the applications of parameterized complexity theory for computational social choice (Betzler et al. 2012).

3 Shift Bribery

Given a voting rule $\mathcal{R}$, in $\mathcal{R}$ SHIFT BRIBERY the goal is to ensure that a certain candidate p (the *preferred* candidate) is an $\mathcal{R}$-winner of the election; this is done by shifting p forward in some of the voters' preference orders. Each shift may have a different price, depending on the voter and the length of the shift. (The problem was defined by Elkind et al. (2009). Here we follow the notation of Elkind and Faliszewski (2010); see the introduction for other related work.)

Let $E = (C, V)$ be some election where $p \in C$ is the preferred candidate. A SHIFT BRIBERY price function π_i for voter $v_i \in V$, $\pi_i\colon \mathbb{N} \to \mathbb{N}$, gives the price of shifting p forward in v_i's preference order a given number of positions. We require that $\pi_i(0) = 0$ and that $\pi_i(\ell) \leq \pi_i(\ell + 1)$ for each $\ell \in \mathbb{N}$. We also assume that if p is ranked on a position r in the preference order, then $\pi_i(\ell) = \pi_i(\ell - 1)$ whenever $\ell \geq r$. In other words, it costs nothing to keep a voter's preference order as is, it never costs less to shift p farther, and we cannot shift p beyond the top position in the preference order. For instance, let v_i be a voter with preference order $v_i\colon c_1 \succ c_2 \succ p \succ c_3$ and let π_i be v_i's SHIFT BRIBERY price function. Then, by paying $\pi_i(1)$ we can change v_i's preference order to $c_1 \succ p \succ c_2 \succ c_3$, and by paying $\pi_i(2)$ we can change it to $p \succ c_1 \succ c_2 \succ c_3$.

It is clear that we need at most $|C|$ values to completely describe each SHIFT BRIBERY price function. We write $\Pi = (\pi_1, \dots, \pi_n)$ to denote the list of SHIFT BRIBERY price functions for the voters in V.

A *shift action* $\vec{s}$ is a vector $(s_1, \dots, s_n)$ of natural numbers, describing how far p should be shifted in each of the n voters' preference orders. We define $\mathrm{shift}(E, \vec{s})$ to be the election $E' = (C, V')$ identical to E, except that p has been shifted forward in each voter v_i's preference order by s_i positions. If that would mean moving p beyond the top position in some preference order, we shift p up to the top position only. We write $\Pi(\vec{s}) = \sum_{i=1}^{n} \pi_i(s_i)$ to denote the price of a given shift action. A shift action s is *successful* if p is a winner in the election $\mathrm{shift}(E, \vec{s})$. The term *unit shift* refers to shifting p by one position in one preference order.

Given the above notation, the decision variant of $\mathcal{R}$ SHIFT BRIBERY is defined as follows.

$\mathcal{R}$ SHIFT BRIBERY
Input: An election $E = (C, V)$, with $V = (v_1, \dots, v_n)$, a list $\Pi = (\pi_1, \dots, \pi_n)$ of SHIFT BRIBERY price functions for V, a candidate $p \in C$, and an integer $B \in \mathbb{N}$.
Question: Is there a shift action $\vec{s} = (s_1, \dots, s_n)$ such that $\Pi(\vec{s}) \leq B$ and p is an $\mathcal{R}$-winner in $\mathrm{shift}(E, \vec{s})$?

The optimization variant is defined analogously, but we do not include B (the budget) in the input and we ask for a shift action $\vec{s}$ that ensures p's victory while minimizing $\Pi(\vec{s})$. For an instance I of the optimization variant of $\mathcal{R}$ SHIFT BRIBERY, we write $\mathrm{OPT}(I)$ to denote the cost of an optimal shift action for I (and we omit I if it is clear from the context). We sometimes also use "$\mathcal{R}$ SHIFT BRIBERY" when we refer to the optimization variant (and this is clear from the context).

An FPT-*approximation scheme* (FPT-AS) with parameter k for $\mathcal{R}$ SHIFT BRIBERY is an algorithm that, given an instance $I = (C, V, \Pi, p)$ and a number $\varepsilon > 0$, returns a successful shift action $\vec{s}$ such that $\Pi(\vec{s}) \leq (1 + \varepsilon)\mathrm{OPT}(I)$. This algorithm must run in $f(k, \varepsilon)(|I|)^{O(1)}$ time where f is a function depending on k and ε.

Parameters for Shift Bribery. So far, SHIFT BRIBERY has not been studied from the parameterized point of view. Dorn and Schlotter (2012) and Schlotter et al. (2011) have, however, provided parameterized complexity results for SWAP BRIBERY and for SUPPORT BRIBERY.

We consider two families of parameters, those referring to the properties of the successful shift action that we seek and those describing the input election. Regarding the first group, we study the total number t of unit shifts in the solution (#shifts), the total number n_a of voters whose preference orders are changed (#voters-affected), and the budget B. Regarding the second group, we consider the number m of candidates (#candidates) and the number n of voters (#voters). We assume that the values of these parameters are passed explicitly as part of the input.

Price Functions. We have argued in the introduction that the price functions used in SHIFT BRIBERY instances may strongly affect the complexity of the problem. Our study of the following families of price functions supports this view.

The NP-hardness proofs of Elkind et al. (2009) use, in essence, what we would call *all-or-nothing* price functions. A SHIFT BRIBERY price function π is all-or-nothing if there is a value c such that $\pi(0) = 0$ and for each $\ell > 0$, $\pi(\ell) = c$ (this value c can be different for each voter). All-or-nothing price functions are a special case of *concave* functions. We are also interested in *convex* price functions; π is convex if for each ℓ, $\pi(\ell+1) - \pi(\ell) \leq \pi(\ell+2) - \pi(\ell+1)$ (provided that it is possible to shift the preferred candidate by up to $\ell + 2$ positions in the given preference order). *Unit prices*, where $\pi(\ell) = \ell$ for each ℓ such that p can be shifted by ℓ positions are an extreme example of convex price functions.

Finally, we consider sortable price functions. A list $\Pi = (\pi_1, \pi_2, \dots)$ of price functions is called *sortable* if for each two voters $v_i, v_j \in V$ with the same preference order (that is $\prec_i = \prec_j$) it holds that $\forall 1 \leq \ell \leq m-2\colon \pi_i(\ell) > \pi_j(\ell) \rightarrow \pi_i(\ell+1) > \pi_j(\ell+1)$. Informally, this means that one can sort each list V' of voters having the same preference order so that the prices of shifting the preferred candidate by each ℓ positions are nondecreasing along the corresponding sorted order of V'. Many natural price function families are sortable. For example, a list of exponential functions of the form $\pi_i(\ell) = a_i^{\ell}$ (where each voter v_i may have an individual base a_i) or of polynomials of the form $\pi_i(\ell) = a_i \cdot \ell^b$ (where the exponent b is the same for the voters having the same preference order but each voter v_i may have an individual coefficient a_i) are sortable.

Given an election $E = (C, V)$, we consider all possible lists of price functions for the voters in V. Let Π_{all} be the set of the lists of all kinds of price functions, Π_{convex} be the set of the lists of convex price functions, Π_{unit} be the set of the lists of unit price functions, $\Pi_{0/1}$ be the set of the lists all all-or-nothing price functions, and Π_{sort} be the set of all sortable lists of price functions. We observe the following relations between these sets.

Proposition 1. *It holds that: (1)* $\Pi_{\mathrm{unit}} \subset \Pi_{\mathrm{convex}} \subset \Pi_{\mathrm{all}}$*, (2)* $\Pi_{0/1} \subset \Pi_{\mathrm{sort}} \subset \Pi_{\mathrm{all}}$*, and (3)* $\Pi_{\mathrm{unit}} \subset \Pi_{\mathrm{sort}}$.

4 Solution Cost Measures Parameterization

In this section we present our results for parameters measuring the solution cost, i.e., for the number of unit shifts, for the number of voters affected by at least one shift, and for the budget. It turns out that parameterization by the number of unit shifts tends to lower complexity (FPT algorithms for Borda and Maximin and W[1]-hardness for Copeland) than parameterization by the number of affected voters (W[2]-hardness). The case of parameterization by the budget lies in between, and the complexity depends on each particular price function family.

Unit Shifts. BORDA SHIFT BRIBERY and MAXIMIN SHIFT BRIBERY parameterized by the number t of unit shifts in the solution are in FPT for arbitrary price functions. The reason is that in these cases it is easy to isolate a small number of candidates on which one needs to focus. More precisely, we can shrink the number of candidates as well as the number of voters to be bounded by functions in t (in effect, achieving partial kernelization (Betzler et al. 2011)).

Theorem 1. BORDA *and* MAXIMIN SHIFT BRIBERY *parameterized by the number t of unit shifts is in* FPT *for arbitrary price functions. The running time of the algorithm is* $O^*((2^t(t+1)t)^t)$.

For Copeland, we do not get FPT membership, but we show W[1]-hardness even for unit prices and for all-or-nothing prices, which implies hardness for each of our price function families.

Theorem 2. COPELAND SHIFT BRIBERY *parameterized by the number of unit shifts is* $\mathrm{W}[1]$*-hard for each price function family that we consider.*

Number of Affected Voters. For the number of affected voters, SHIFT BRIBERY is W[2]-hard for each of Borda, Maximin, and Copeland$^\alpha$, for each family of price functions that we consider: The result for all-or-nothing prices follows almost directly from the NP-hardness proofs due to Elkind et al. (2009) (their reductions have to be adapted to work for SET COVER rather than its specialized variant, X3C, but this can be done quite easily). To obtain the result for unit prices, with some effort, it is still possible to carefully modify their proofs, maintaining their main ideas.

Corollary 1. BORDA, MAXIMIN, *and* COPELAND SHIFT BRIBERY *parameterized by the number of affected voters are* $\mathrm{W}[2]$*-hard for each price function family that we consider.*

Budget. When we parameterize SHIFT BRIBERY by the available budget, the results fall between those from the previous two paragraphs. In essence, the hardness proofs for all-or-nothing prices carry over from the number of affected voters case to the budget case (and this implies hardness for arbitrary prices and sortable prices), while the results for the number of unit shifts carry over to the setting with convex/unit prices.

The hardness results translate because in the construction for all-or-nothing prices behind Corollary 1, the budget equals the parameter value "solution size" of the SET COVER instance from which we reduce. The FPT results parameterized by the number of unit shifts translate because for convex/unit prices the budget is an upper bound on the number of possible unit shifts.

Corollary 2. BORDA *and* MAXIMIN SHIFT BRIBERY *parameterized by the budget B are* $\mathrm{W}[2]$*-hard for arbitrary, sortable, and all-or-nothing prices, and are in* FPT *for convex and unit prices with running time* $O^*((2^B(B+1)B)^B)$.

For COPELAND SHIFT BRIBERY for unit prices, the budget equals the number of unit shifts in the reduction behind Theorem 2. Altogether, this implies the following corollary.

Corollary 3. COPELAND SHIFT BRIBERY *parameterized by the budget is* $\mathrm{W}[2]$*-hard for arbitrary, sortable, and all-or-nothing prices, and is* $\mathrm{W}[1]$*-hard for convex and unit prices.*

5 Election-Size Measures Parameterization

In this section we consider SHIFT BRIBERY parameterized by either the number of candidates or the number of voters. Elections with few candidates are natural in politics (for example, there is typically only a handful of candidates in presidential elections) and elections with few voters arise naturally in multiagent systems (for example, Dwork et al. (2001) suggested election-based methods for aggregating results from several web search engines).

Number of Voters. Let us now consider SHIFT BRIBERY parameterized by the number of voters. We have not found FPT algorithms for our rules in this setting, but we did find a general FPT-approximation scheme.

Theorem 3. *Let $\mathcal{R}$ be a voting rule for which winner-determination parameterized by the number n of voters is in* FPT. *There is a factor-*$(1+\varepsilon)$ *approximation algorithm solving $\mathcal{R}$* SHIFT BRIBERY *in time* $O^*(\lceil n/\varepsilon + 1\rceil^n)$ *times the cost of $\mathcal{R}$'s winner determination.*

Theorem 3 follows by combining a brute-force search with price scaling.

For the case of all-or-nothing price functions we can obtain a very simple FPT algorithm.

Proposition 2. *Let $\mathcal{R}$ be a voting rule for which winner-determination parameterized by the number of voters is in* FPT. $\mathcal{R}$ SHIFT BRIBERY *parameterized by the number of voters is in* FPT *for all-or-nothing prices.*

In contrast to Proposition 2, fixed-parameter tractability for other price functions is not obvious. Indeed, for Copeland we can show W[1]-hardness for unit prices by a quite technically involved reduction from the W[1]-complete CLIQUE parameterized by the solution size.

Theorem 4. COPELAND SHIFT BRIBERY *parameterized by the number of voters is* $\mathrm{W}[1]$*-hard for unit prices.*

This result shows that Theorem 3 is essentially tight because winner determination for Copeland is polynomial time solvable (and thus in FPT).

Number of Candidates. As opposed to almost all other (unweighted) election problems ever studied in computational social choice, for SHIFT BRIBERY (and for bribery problems in general) the parameterization by the number of candidates is one of the most notorious ones. In other election problems, the natural, standard attack is to give integer linear program (ILP) formulations and use Lenstra's algorithm (Lenstra, Jr. 1983). For instance, this has been applied for winner determination (Bartholdi, Tovey, and Trick 1989), control (Faliszewski et al. 2007), possible winner (Betzler, Hemmann, and Niedermeier 2009), and lobbying (Bredereck et al. 2012) problems. This works because with m candidates there at most $m!$ different preference orders and we can have a variable for each of them in the ILP. However, in our setting this approach fails. The reason is that in bribery problems the voters are not only described by their preference orders, but also by their prices. This means that we cannot lump together a group of voters with the same preference order anymore and we have to treat each of them individually.

Dorn and Schlotter (2012) have already considered the complexity of swap bribery parameterized by the number of candidates. However, their proof implicitly assumes that each voter has the same price function and, thus, it implies that SHIFT BRIBERY (parameterized by the number m of candidates) is in FPT for unit prices, but not necessarily for the other families of price functions. Whenever the number of different prices or different price functions is bounded by some constant or, at least, by some function only depending on m, Dorn and Schlotter's approach can be adapted.

For the more general price function families, we were neither able to find alternative FPT attacks nor to find hardness

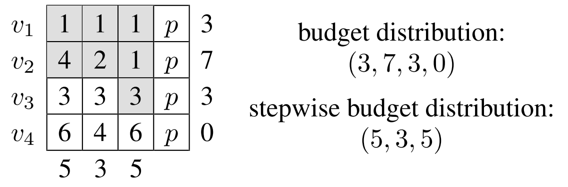
v_1 1 1 1 p 3
v_2 4 2 1 p 7
v_3 3 3 3 p 3
v_4 6 4 6 p 0
5 3 5
budget distribution: $(3, 7, 3, 0)$
stepwise budget distribution: $(5, 3, 5)$

Figure 1: Illustration of the shift action $(3, 3, 1, 0)$ restricted to a specific voter block, where p ranks fourth in the common preference order. $\pi_1(i) = i$, $\pi_2(i) = 2^i - 1$, $\pi_3(i) = 3i$, and $\pi_4(i) = i^2 + i + 4$ with $i < 4$. The voters are sorted descending with their prices. Each row in the matrix represents one voter. The entry i cells left of p contains the price for moving p from position $4 - i + 1$ to position $4 - i$. The cell in row j and column $4 - i$ is marked gray if we shift p by i positions in voter j's preference order. Counting the number of marked cells row-wise gives the shift action. Summing up the costs row-wise gives the budget distribution. Summing up the costs column-wise gives the stepwise budget distribution.

proofs (which, due to the limited number of candidates one can use and the fact that the voters are unweighted, seem particularly difficult to design). However, as in the case of parameterization by the number of voters (Theorem 3), we can show that there is an FPT-approximation scheme for the number of candidates when the prices are sortable.

Theorem 5. *Let $\mathcal{R}$ be a voting rule for which winner-determination parameterized by the number m of candidates is in* FPT. *There is a factor-$(1 + 2\varepsilon + \varepsilon^2)$ approximation algorithm solving $\mathcal{R}$* SHIFT BRIBERY *for sortable prices in time $O^*(M^{M\lceil\ln(M/\varepsilon)\rceil+1})$ (where $M = m \cdot m!$) times the cost of $\mathcal{R}$'s winner determination.*

Proof idea. We first give an algorithm that given a target budget B' either computes a successful shift action with cost at most $(1 + \varepsilon)B'$, or declares that a successful shift action with cost at most B' does not exist. It consists of two parts:

1. FPT part: Exhaustively search for a distribution of a large chunk of the budget among the voters, that leads to an almost successful shift action.
2. Approximation part: Spend several copies of the remaining (small) part of the budget to complete the shift action.

To find a successful shift action, one only has to know the distribution of the budget among the voters. Unfortunately, the number of vectors describing the distribution of the budget among the voters is not bounded by a function of m. Thus, the key idea is to have a more compact view on how the budget is used. We make the following observations.

First, instead of asking how much of the budget is used to bribe each voter, we can ask how much of the budget is used to convince each voter to shift p one step from the original position, how much of the budget is used to convince him to shift p one step further, and so on. (So far, this increases the amount of information needed, but see the next point.)

Second, we can group the voters with identical preferences into *voter blocks* and sort them within the blocks according to their price functions. This operation is well-defined because price functions are sortable. W.l.o.g., within each voter block, we never spend more units of the budget on a more expensive voter than on a less expensive one.

Hence, to describe a shift action, it suffices—for each block and each position i—to indicate the amount of budget spent on shifting p from position i to position $i - 1$ in this block. Thus, we ask for a distribution of the budget B' into at most $M := m! \cdot m$ slots. We call this *stepwise budget distribution*. Its relation to the ordinary budget distribution and to the shift action is illustrated in Figure 1.

The FPT part of our algorithm is based on the pigeonhole principle: If there is a successful shift action of cost at most, say, B^*, then at least one slot in the stepwise budget has at least B^*/M units of the budget. In more detail, the FPT part works as follows. We start with $B^* := B'$ and branch over the slot in the stepwise budget distribution which gets B^*/M and set $B^* := B^*(1 - 1/M)$. After $M\lceil\ln(\frac{M}{\varepsilon})\rceil + 1)$ iterations, an extant budget of $B^* \leq \varepsilon/M \cdot B'$ is left. In the approximation part, we spend this final part of the budget on *each* slot separately (i.e., we add B^* to each slot in the stepwise budget).

One can show that there is one branching path in the FPT part where each slot of the stepwise budget distribution gets at most as much as in some optimal solution. Moreover, at most $\varepsilon B'/M$ of the budget is left and hence each slot needs at most $\varepsilon B'/M$ to reach the value of an optimal solution. Thus, the final approximated budget distribution leads to a successful shift action with costs at most $(1 + \varepsilon)B'$. Let B be the cost of some optimal solution. We run this algorithm $O(\log B)$ times for an increasing sequence of budget bounds B'. The total cost B' satisfies $B \leq B' < (1 + \varepsilon)B$. ☐

Adopting some of the ideas of the above proof, we obtain the following for SHIFT BRIBERY with arbitrary prices.

Proposition 3. *Let $\mathcal{R}$ be a voting rule whose winner-determination procedure is in* XP *when parameterized by the number m of candidates. $\mathcal{R}$* SHIFT BRIBERY *parameterized by m is in* XP *for each price function family that we consider. The algorithm runs in time $O^*((n^m)^{m!})$ times the cost of $\mathcal{R}$'s winner determination.*

6 Conclusions

We have studied the parameterized complexity of SHIFT BRIBERY under the voting rules Borda, Copeland, and Maximin, for several natural parameters (either describing the nature of the solution or the size of the election) and for several families of price functions (arbitrary, convex, unit, sortable, and all-or-nothing). Our results confirmed the belief that the computational complexity depends on all three factors: the voting rule, the parameter, and the type of price function used.

Our work leads to some questions, of which the most pressing one asks about the exact complexity of SHIFT BRIBERY parameterized by the number of candidates; we only have XP and FPT-approximation schemes here (Table 1). Typically, this is the easiest parameter to deal with in election problems but since here voters have possibly different prices, it is one of the most challenging ones to work with.

Acknowledgments

Robert Bredereck was supported by the DFG project PAWS (NI 369/10). Jiehua Chen was supported by the Studienstiftung des Deutschen Volkes. Piotr Faliszewski was partially supported by the DFG project PAWS (NI 369/10).

References

Bartholdi, III, J.; Tovey, C.; and Trick, M. 1989. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare* 6(2):157–165.

Betzler, N.; Guo, J.; Komusiewicz, C.; and Niedermeier, R. 2011. Average parameterization and partial kernelization for computing medians. *Journal of Computer and System Sciences* 77(4):774–789.

Betzler, N.; Bredereck, R.; Chen, J.; and Niedermeier, R. 2012. Studies in computational aspects of voting—a parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, volume 7370 of *Lecture Notes in Computer Science*. Springer-Verlag. 318–363.

Betzler, N.; Hemmann, S.; and Niedermeier, R. 2009. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, 53–58. AAAI Press.

Bredereck, R.; Chen, J.; Hartung, S.; Niedermeier, R.; Suchý, O.; and Kratsch, S. 2012. A multivariate complexity analysis of lobbying in multiple referenda. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI '12)*, 1292–1298.

Dorn, B., and Schlotter, I. 2012. Multivariate complexity analysis of swap bribery. *Algorithmica* 64(1):126–151.

Downey, R. G., and Fellows, M. R. 2013. *Fundamentals of Parameterized Complexity*. Springer-Verlag.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, 613–622. ACM Press.

Elkind, E., and Faliszewski, P. 2010. Approximation algorithms for campaign management. In *Proceedings of the 6th International Workshop On Internet And Network Economics (WINE '10)*, 473–482. Springer-Verlag *Lecture Notes in Computer Science #6484*.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2009. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT '09)*, 299–310. Springer-Verlag *Lecture Notes in Computer Science #5814*.

Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2007. Llull and Copeland voting broadly resist bribery and control. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI '07)*, 724–730. AAAI Press.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer-Verlag.

Lenstra, Jr., H. 1983. Integer programming with a fixed number of variables. *Mathematics of Operations Research* 8(4):538–548.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Schlotter, I.; Faliszewski, P.; and Elkind, E. 2011. Campaign management under approval-driven voting rules. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI '11)*, 726–731. AAAI Press.