Winner Robustness via Swap- and Shift-Bribery: Parameterized Counting Complexity and Experiments

Niclas Boehmer*, Robert Bredereck, Piotr Faliszewski, and Rolf Niedermeier

Abstract

We study the parameterized complexity of counting variants of Swap- and Shift-Bribery, focusing on the parameterizations by the number of swaps and the number of voters. Facing several computational hardness results, using sampling we show experimentally that counting variants of Swap-Bribery offer a new approach to the robustness analysis of elections.

1 Introduction

Consider a university department which is about to hire a new professor. There are m candidates and the head of the department decided to choose the winner by Borda voting. Each faculty member (i.e., each voter) ranked the candidates from the most to the least appealing one, each candidate received $m - i$ points for each vote where he or she was ranked as the i-th best, and the candidate with the highest score was selected. After the results were announced, some voters started wondering if, perhaps, some *other* voters accidently "misranked" some of the candidates (worrying about mistakes in the votes is an old democratic tradition). For instance, if some voter viewed two candidates as very similar, then he or she could have ranked them either way, depending on an impulse. Or, some voter would have ranked two candidates differently if he or she had more information on their merits. It is, thus, natural to ask for the probability of changing the election outcome by making some random swaps. Indeed, this approach was recently pursued by Baumeister and Hogrebe [3] and we follow-up on it, but with a different focus (see the discussion of related work).

Specifically, in our model for each $r \in \mathbb{N}$ and each candidate c, we let $P_c(r)$ be the probability that c wins an election obtained by making r random swaps of candidates ranked on adjacent positions in the votes (we refer to such elections as being at swap distance r from the original one). Such values can be quite useful. For example, if for each r we had (some estimate of) the probability that in total there are r accidental swaps in the votes, then we could compute the probability of each candidate's victory. If it were small for the original winner, then we might want to recount the votes or reexamine the election process. The values $P_c(r)$ are also useful without the distribution of r's. For example, we may want to find the smallest number of (random) swaps for which the probability of the original winner's victory drops below some value (such as 50%) or for which he or she is no longer the most probable winner. As we show in our experiments, this approach provides new insights on the robustness of election results.

To determine the value $P_c(r)$, we need to divide the number of elections at swap distance r where c wins, by the total number of elections at this distance. While computing the latter is easy—at least in the sense that there is an efficient algorithm for this task—computing the former requires solving the counting variant of the Swap-Bribery problem (denoted #Swap-Bribery). Briefly put, in the decision variant of the problem, we ask if it is

*Supported by the DFG project MaMu (NI 369/19).

possible to ensure that a designated candidate wins a given election by making r swaps of adjacent candidates in the votes (we assume the unit prices setting; see Section 2). In the counting variant, we ask how many ways there are to achieve this effect (using exactly r swaps). Unfortunately, already the decision variant is NP-hard for many voting rules and the counting one is hard even for Plurality. On the positive side, we can get a good estimate of $P_c(r)$ by sampling (see Footnote 3 in Section 4).

	Plurality		Borda	
	decision	counting	decision	counting

SWAP-BRIBERY	P	#P-hard
		#W[1]-h.(r)
		FPT(n)
			NP-hard
			FPT(r)
			W[1]-h.(n)
				#P-hard
				?
				#W[1]-h.(n)

CONST. SHIFT-BRIBERY	P	P	NP-hard
			FPT(r)
			W[1]-h.(n)
				#P-hard
				FPT(r)
				#W[1]-h.(n)

DEST. SHIFT-BRIBERY	P	P	P	#P-hard #W[1]-h.(r) #W[1]-h.(n)

Table 1: (Parameterized) complexity of SWAP- and SHIFT-BRIBERY with unit prices; r and n refer to the parameterizations by the swap/shift radius and by the number of voters, respectively. Results for the counting variants are new (see also the work of Baumeister and Hogrebe [3] for results related to #P-hardness of SWAP-BRIBERY); results for the decision variants are due to Elkind et al. [16], Bredereck et al. [7, 6], and Kaczmarczyk and Faliszewski [21].

We also consider the SHIFT-BRIBERY problem, a variant of SWAP-BRIBERY where we can only shift the designated candidate forward (in the constructive case) or backward (in the destructive one, where the goal is to ensure that the designated candidate loses). These problems also can be used to evaluate robustness of election results but, to maintain focus, in our experiments we only consider SWAP-BRIBERY. Yet, we include SHIFT-BRIBERY in our complexity analysis because it illustrates some interesting phenomena.

Main Contributions. We focus on #SWAP- and #SHIFT-BRIBERY for the Plurality and Borda voting rules (for unit prices). We consider their computational complexity for parameterizations by the number of unit swaps/shifts (which we refer to as the swap/shift radius) and by the number of voters (see Table 1). We also present experiments, where we use #SWAP-BRIBERY to evaluate the robustness of election results. Our main results are as follows:

1. For Plurality, SWAP-BRIBERY is known to be in P, but we show that the counting variant is #P-hard and #W[1]-hard when parameterized by the swap radius.

2. For Borda, hardness results for #SWAP-BRIBERY follow from those for #SHIFT-BRIBERY, which themselves are intriguing: E.g., the destructive variant parameterized by the shift radius is #W[1]-hard, but the constructive one is in FPT; in the decision setting the former is easier.[1]

3. Using sampling, we estimate the candidate's winning probabilities in elections from a synthetic dataset generated by Szufa et al. [30]. One of the high-level conclusions is that the score differences between the election winners and the runners-up can be quite disconnected from their strengths (measured using #SWAP-BRIBERY). Afterwards, we show that our conclusions also extend to real-life elections by considering a new dataset consisting of elections obtained from two annual cycling competitions, that is, from Tour de France and from Giro d'Italia.

The proofs of results marked by (★) and some further analyses are available in the appendix.

Related Work. Our work is most closely related to the papers of Hazon et al. [20], Bachrach et al. [1], and Baumeister and Hogrebe [3]. Similar to our approach, the authors study the complexity of computing the probability that a given candidate wins, provided that

[1] We abuse the notation here and use P and FPT to refer to the classes of, respectively, polynomial-time computable and fixed-parameter tractable counting problems.

the votes may change according to some probability distribution. In particular, Hazon et al. [20] assume that each voter is endowed with an explicitly encoded list of possible votes, each with its own probability of being cast, Bachrach et al. [1] consider elections where the votes are partial and all completions are equally likely, and Baumeister and Hogrebe [3] consider both these models, as well as a third one, which—in terms of computational complexity—is equivalent to our model.

There are two methodological differences between our work and the three above-discussed papers. Foremost, we provide an experimental analysis showing that counting variants of Swap-Bribery are indeed helpful for evaluating the robustness of election winners. In contrast, Bachrach et al. [1] and Baumeister and Hogrebe [3] focus entirely on the complexity analysis, whereas Hazon et al. [20] also provide experiments, but their focus is on the running time and memory consumption of their algorithm.

The second difference is that we focus on a parameterized complexity analysis—with an explicit focus on establishing FPT and #W[1]-hardness results—which was not done in previous works (except for obtaining XP algorithms for the number of candidates or voters).

Swap- and Shift-Bribery were introduced by Elkind et al. [16]. Various authors studied these problems for different voting rules (see, e.g., the works of Maushagen et al. [26] and Zhou and Guo [36] regarding iterative elections), sought approximation algorithms [15, 18], established parameterized complexity results [14, 6, 23], considered restricted preference domains [17], and extended the problem in various ways [7, 21, 4, 35]. The idea of using Swap-Bribery to measure the robustness of election results is due to Shiryaev et al. [29], but is also closely related to computing the margin of victory [25, 11, 34, 10]; recently it was also applied to committee elections [8].

So far, the complexity of counting problems received limited attention in the context of elections. In addition to the works of Hazon et al. [20], Bachrach et al. [1], and Baumeister and Hogrebe [3], we mention two more: Wojtas and Faliszewski [33] studied counting solutions for control problems, and Kenig and Kimelfeld [22] followed up on the work of Bachrach et al. [1] and provided approximation algorithms for their setting.

2 Preliminaries

For each integer k, by $[k]$ we mean the set $\{1, \dots, k\}$.

Elections. An election $E = (C, V)$ consists of a set $C = \{c_1, \dots, c_m\}$ of candidates and a collection $V = (v_1, \dots, v_n)$ of voters. Each voter v_i has a preference order, which ranks all candidates from the most to the least desired one (we sometimes refer to preference orders as votes). For a voter v_i, we write $v_i \colon c_1 \succ c_2 \succ \cdots \succ c_m$ to indicate that he or she ranks c_1 first, then c_2, and so on. If we put a subset of candidates in such a description of a preference order, then we mean listing its members in an arbitrary order.

Voting Rules. A voting rule $\mathcal{R}$ is a function that, given an election, returns a set of candidates that tie as winners. We focus on Plurality and Borda, which assign scores to the candidates and select those with the highest ones. Under Plurality, each voter gives one point to the top-ranked candidate. Under Borda, each voter gives $|C| - 1$ points to the top-ranked candidate, $|C| - 2$ points to the next one, and so on. We write $\mathrm{score}_E(c)$ to denote the score of candidate c in election E (the voting rule will be clear from the context).

Swap Distance. Let u and v be two votes over the same candidate set. The swap distance between u and v, denoted $d_{\mathrm{sw}}(u, v)$, is the length of the shortest sequence of swaps of adjacent candidates whose application transforms u into v. Given elections $E = (C, V)$ and $E' = (C, V')$, where $V = (v_1, \dots, v_n)$ and $V' = (v'_1, \dots, v'_n)$, their swap distance is $\sum_{i=1}^{n} d_{\mathrm{sw}}(v_i, v'_i)$. By $R(E, r)$, we denote the set of elections at swap distance r from E.

Swap- and Shift-Bribery. Let $\mathcal{R}$ be a voting rule. In the decision variant of the $\mathcal{R}$ SWAP-BRIBERY problem, we are given an election E, a designated candidate p, and a budget r. Further, for each voter v and each two candidates c and d, we have a nonnegative price $\pi_v(c, d)$ for swapping them in v's preference order (a swap is legal if at the time of its application c and d are adjacent). We ask if there is an election E' where p is an $\mathcal{R}$-winner, such that E' can be obtained from E by performing a sequence of legal swaps of cost at most r. In the counting variant, we ask for the number of such elections, and we require the cost of swaps to be exactly r (the last condition is for our convenience and all our results would still hold if we asked for cost at most r; the same would be true if instead of counting elections where p won, we would count those where he or she lost). Since we are interested in computing the candidates' probabilities of victory in elections at a given swap distance, we focus on the case where each swap has the same, unit price.

CONSTRUCTIVE SHIFT-BRIBERY is a variant of SWAP-BRIBERY where all swaps must involve the designated candidate, shifting him or her forward. DESTRUCTIVE SHIFT-BRIBERY is defined analogously, except that our goal is to preclude the designated candidate's victory, and we can only shift him or her backward [21]. Counting variants are defined in a natural way. We focus on the case where each unit shift has a unit price. In general, we speak of shift or swap radius r instead of budget.

Counting Complexity. We assume basic familiarity with (parameterized) complexity theory, including classes P, NP, FPT, and W[1], and reducibility notions.

Let X be a decision problem from NP, where for each instance we ask if there exists some mathematical object with a given property. In its counting variant, traditionally denoted #X, we ask for the number of such objects. For example, in MATCHING we are given an integer k and a bipartite graph G—with vertex set $U(G) \uplus V(G)$ and edge set $E(G)$—and we ask if G contains a matching of size k (i.e., a set of k edges, where no two edges touch the same vertex). In #MATCHING we ask how many such matchings exist.

The class #P is the counting analog of NP; a problem belongs to #P if it can be expressed as the task of counting the number of accepting computations of a nondeterministic polynomial-time Turing machine. We say that a counting problem #A (polynomial-time) Turing reduces to #B if there exists an algorithm that solves #A in polynomial time, provided that it has oracle access to #B. A problem is #P-hard if every problem from #P Turing reduces to it. While MATCHING is in P, it is well known that #MATCHING is #P-hard and #P-complete [31].

#W[1] relates to W[1] in the same way as #P relates to NP. As examples of #W[1]-hard problems, we mention counting size-k cliques in a graph, parameterized by k [19] and #MATCHING, parameterized by the size of the requested matching [13]. Formally, #W[1]-hardness is defined using a slightly more general notion of a reduction, but for our purposes polynomial-time Turing reductions (where the parameters in the queried instances are bounded from above by a function of the parameter in the input instance) will suffice.

3 Algorithms and Complexity Results

In this section, we present our results regarding the complexity of #SWAP- and #SHIFT-BRIBERY. We first consider Plurality, mostly focusing on #SWAP-BRIBERY, and then discuss Borda, mostly focusing on #SHIFT-BRIBERY.

3.1 Plurality and #Swap-Bribery

We start with bad news. While there is a polynomial-time algorithm for the decision variant of PLURALITY SWAP-BRIBERY [16], the counting variant is intractable, even with unit prices

(#P-hardness for a slightly different but computationally equivalent model is also reported by Baumeister and Hogrebe [3]).

Theorem 1. PLURALITY #SWAP-BRIBERY *is* #P*-hard and* #W[1]*-hard when parameterized by the swap radius, even for unit prices.*

Proof. We give a reduction from #MATCHING. We will use a swap radius upper-bounded by a function of the desired matching size, so we will obtain both #P- and #W[1]-hardness.

Let (G, k) be an instance of #MATCHING, where G is a bipartite graph with vertex set $U(G) \uplus V(G)$ and k is the size of matchings that we are to count. Assume that $U(G) = \{u_1, \dots, u_n\}$, $V(G) = \{v_1, \dots, v_n\}$, and $k \leq n$. To form an election, denoted E, we let the candidate set be $C := U(G) \uplus V(G) \uplus \{p, a, b\} \uplus X$, where $X := \{x_1, \dots, x_{3k+1}\}$. The candidates in $U(G) \uplus V(G)$ will model the graph, p will be our designated candidate, a and b will control the size of the matching, and the candidates in X will block undesirable swaps. We will have the following scores of the candidates:

$$\forall c \in C \setminus \{a, b\} \colon \operatorname{score}_E(c) = n, \ \operatorname{score}_E(a) = n - k, \ \operatorname{score}_E(b) = n + k.$$

We form the following four groups of voters:

1. For each $\{u_i, v_j\} \in E(G)$, there is an *edge* voter e_{ij} with preference order $e_{ij} \colon u_i \succ v_j \succ X \succ \cdots$.
2. For each $j \in [n]$, we have an *a-voter* a_j with preference order $a_j \colon v_j \succ a \succ X \succ \cdots$.
3. For each $i \in [n]$, we have a *b-voter* b_i with preference order $b_i \colon b \succ u_i \succ X \succ \cdots$.
4. Finally, the *score voters* implement the desired Plurality scores. For each candidate $c \in U(G) \cup V(G) \cup \{p\}$, there are exactly as many voters with preference order $c \succ X \succ \cdots$ as necessary to ensure that in total c has score n. Similarly, for each $x_i \in X$ there are n voters with preference order $x_i \succ X \setminus \{x_i\} \succ \cdots$. There are also $n - k$ voters with preference order $a \succ X \succ \cdots$ and k voters with preference order $b \succ X \succ \cdots$.

Let E be an election with the above-described candidates and voters. We form an instance I of PLURALITY #SWAP-BRIBERY with this election, unit prices, and swap radius $r := 3k$. Then, we make an oracle query for I and return its answer. In the following, we argue that this answer is equal to the number of size-k matchings in G. The idea is that to make p a winner, we have to transfer k points from b to a via swaps that correspond to a matching.

Let E' be some election in $R(E, r)$, i.e., an election at swap distance r from E, where p wins. We note that p and the candidates from X have score n in E' (indeed, in elections from $R(E, r)$, p has score at most n and the average score of the candidates in X is at least n). Further, in E' each edge voter, a-voter, and b-voter either ranks on top the same candidate as in E or the candidate that he or she ranked second in E, and each score voter ranks the same candidate on top as in E (otherwise some candidate in X would have score above n). We call this the *top-two rule*.

Since b must have at most n points in E', by the top-two rule, there must be at least k b-voters that rank members of $U(G)$ on top. Let U_b be the set of these members of $U(G)$. As each member of $U(G)$ can be swapped with b at most once in the b-votes, we have $|U_b| \geq k$.

Compared to E, in E' each member of U_b gets an additional point from the b-voters. Thus, for each $u_i \in U_b$ there must be a voter that ranked u_i on top in E but does not do so in E'. By the top-two rule, this must be an edge voter. Let M be the set of pairs $\{u_i, v_j\}$ such that in E edge voter e_{ij} ranks u_i on top, but in E' he or she ranks v_j on top. Naturally, we must have $|M| \geq |U_b|$.

For each pair $\{u_i, v_j\} \in M$, there must be a voter who swapped v_j out of the top position in E', because otherwise v_j would have more than n points. By the top-two rule, this must be voter a_j. Let V_a be the set of those members of $V(G)$ that in E' are swapped out of the top positions in the corresponding a-vote. It must be that $|V_a| \geq |M|$.

Thus, we have $|V_a| \geq |M| \geq |U_b| \geq k$ and, in fact, each of these sets must have exactly k elements (because they are disjoint). Further, M is a matching. If it were not, then some member of $U(G) \uplus V(G)$ would appear in two pairs in M, but then we would have to have two a-voters or two b-voters corresponding to this candidate, which is not possible in our construction.

This way we have shown that for each election in $R(E, r)$ where p wins, there is a corresponding size-k matching. As the other direction is immediate, the proof is complete. □

A natural way to circumvent such intractability results is to seek FPT algorithms parameterized by the number of candidates or by the number of voters. For the former, one typically expresses SWAP-BRIBERY problems as integer linear programs (ILPs) and invokes the classic algorithm of Lenstra, Jr. [24], or some more recent one; see, e.g., the work of Knop et al. [23]. Unfortunately, in case of counting there are two issues. First, counting analogues of these algorithms, dating back to the seminal work of Barvinok [2], have XP running times. However, fortunately, the ILPs used for SWAP-BRIBERY have such a special form that in their case Barvinok's algorithm would run in FPT time for the parameterization by the number of candidates. The second obstacle is more serious. Even though we could count the number of solutions to our ILPs, each of these solutions would potentially correspond to a different number of solutions for SWAP-BRIBERY. Dealing with this problem, so far, remains elusive and we leave it as an open problem.

Yet, for unit prices we do show an FPT algorithm parameterized by the number of voters.

Theorem 2 (★). *For unit prices,* PLURALITY #SWAP-BRIBERY *parameterized by the number of voters is in* FPT.

Proof sketch. Consider an instance I of PLURALITY #SWAP-BRIBERY with election $E = (C, V)$, where $C = \{c_1, c_2, \ldots, c_m\}$ and V contains n voters. Let r be the swap radius and, w.l.o.g., let $p = c_1$ be the designated candidate.

The core idea is to go over all possible sequences $\mathcal{V} = (V_1, \ldots, V_{m'})$ such that (a) $m' \leq \min(m, n)$, (b) each V_i is a subcollection of V (consisting of not necessarily consecutive voters), (c) each voter belongs to exactly one V_i, and (d) group V_1 has at least as many voters as every other group. For each such sequence, we solve the following *global counting problem*: Count the number of ways to perform exactly r swaps so that (i) within each V_i, each two voters rank the same candidate, denoted $c(V_i)$, on top, (ii) all voters in V_1 rank p on top (i.e., $c(V_1) = p$), and (iii) for each two groups V_i and $V_{i'}$, where $i < i'$ and we have $c_j = c(V_i)$ and $c_{j'} = c(V_{i'})$, it holds that $j < j'$. These conditions ensure that after performing the swaps, each group votes for a different candidate, each candidate $c(V_i)$ receives exactly $|V_i|$ points, and p wins. One can verify that every solution for our input instance corresponds to exactly one sequence $\mathcal{V}$. In other words, to obtain the answer for I, we need to sum up the answers for the global counting problems for each $\mathcal{V}$.

To solve a given global counting problem in polynomial time, we define table $T[i, \ell, s]$ to be the number of ways to perform exactly s swaps within the first i voter groups, so that conditions (i)–(iii) hold for $V_1, \ldots V_i$, and so that $c(V_i) = c_\ell$. The solution is then $\sum_{\ell \leq m} T[m', \ell, r]$. We compute these values using dynamic programming (which requires solving a *local counting problem*, also via dynamic programming, to count for each voter group the number of ways to ensure that all its members rank a given candidate on top).

As the number of global counting problems to solve is bounded from above by a function of n, and each such problem is solvable in polynomial time, the algorithm runs in FPT time with respect to the number of voters. □

The restriction to unit prices in Theorem 2 is necessary. Otherwise, a reduction from the problem of counting linear extensions of a partially ordered set [9] shows #P-hardness even for a single voter.

Theorem 3 (★). PLURALITY #SWAP-BRIBERY *is* #P*-hard even for a single voter and unary-encoded prices.*

We conclude with a brief mention of #SHIFT-BRIBERY. Both the constructive and the destructive variant are in P, even with arbitrary unary-encoded prices (for the binary encoding, #P-hardness follows by a reduction from #PARTITION). Our algorithms use dynamic programming over groups of voters with the same candidate as their top choice.

Theorem 4 (★). *For unary-encoded prices, both the constructive and the destructive variant of* PLURALITY #SHIFT-BRIBERY *are in* P*.*

3.2 Borda and #Shift-Bribery

Our results for BORDA #SWAP-BRIBERY follow from those for #SHIFT-BRIBREY, so we mainly focus on the latter problem.

In the decision setting, the constructive variant of BORDA SHIFT-BRIBERY is NP-hard (and is in FPT when parameterized by the shift radius, but is W[1]-hard for the number of voters), whereas the destructive variant is in P. In the counting setting, both variants are #P-hard and #W[1]-hard for the parameterization by the number of voters; the result for the constructive case follows from a proof for the decision variant due to Bredereck et al. [7], and for the destructive case we use a similar approach with a few tricks on top.

Theorem 5 (★). *Both the constructive and the destructive variant of* BORDA #SHIFT-BRIBERY *are* #P*-hard and* #W[1]*-hard when parameterized by the number of voters.*

Surprisingly, when parameterized by the shift radius, the constructive variant is in FPT and the destructive variant is #W[1]-hard. Not only does the problem that was easier in the decision setting now become harder, but also—to the best of our knowledge—it is the first example where a destructive variant of an election-related problem is harder than the constructive one. Yet, SHIFT-BRIBERY is quite special as the two variants differ in the available actions, i.e., either shifting the designated candidate forward or backward (typically, voting problems for both variants have the same sets of actions).

The FPT algorithm for the constructive case relies on the fact that if we can ensure victory of the designated candidate by shifting him or her by r positions forward, then there are at most r candidates that we need to focus on (the others will be defeated irrespective of what exact shifts we make). There are no such bounds in the destructive setting.

Theorem 6 (★). *Parameterized by the shift radius,* BORDA #CONSTRUCTIVE SHIFT-BRIBERY *is in* FPT *(for unary-encoded prices), but the destructive variant is* #W[1]*-hard, even for unit prices.*

For BORDA #SWAP-BRIBERY, we obtain #P-hardness and #W[1]-hardness when parameterized by the number of voters by noting that the proofs for #SHIFT-BRIBERY still apply in this case. The parameterization by the swap radius remains open, though (the proof of Theorem 6 does not work as, for swap bribery, many new, hard to control, solutions appear).

Corollary 1. BORDA #SWAP-BRIBERY *is* #P*-hard and* #W[1]*-hard when parameterized by the number of voters, even for the case of unit prices.*

4 Experiments

In the following, we use #Swap-Bribery to analyze the robustness of election winners experimentally. For clarity, in this section we use normalized swap distances, which specify the fraction of all possible swaps in a given election. We start by considering a dataset consisting of synthetic data before we turn to elections obtained from cycling races.

4.1 Synthetic Data

We used a synthetic dataset of 800 elections, each with 10 candidates and 100 voters, prepared by Szufa et al. [30] (see `https://mapel.readthedocs.io/`). This dataset contains elections generated from various statistical cultures, of which for us the most relevant are the following ones (for details, we point to Appendix B.1 and the work of Szufa et al. [30]):

1. The impartial culture model (IC), where each election consists of preference orders chosen uniformly at random.
2. The urn model, with the parameter of contagion $\alpha \in \mathbb{R}_+$; for $\alpha = 0$ the model is equivalent to IC, but as α grows, larger groups of identical votes become more probable.
3. The Mallows model, with dispersion parameter $\phi \in [0, 1]$, where the votes are generated by perturbing a given central one; for $\phi = 0$ only the central vote appears, and for $\phi = 1$ the model is equivalent to IC.
4. The tD-Cube/Sphere models, where the candidates and voters are points in a t-dimensional hypercube/sphere and the voters rank the candidates by distance (we refer to 1D/2D-Cube elections as 1D-Interval/2D-Square ones).

Szufa et al. [30] present their elections as a *map* (see Figure 1). Note that the map contains elections from a number of distributions beyond those mentioned above (see Appendix B.1 for a full list); "IC and similar" refers to IC elections, Mallows elections with ϕ values close to 1, and a few other similar elections. Later we will use the map to present our results.

Computations. For each election E and each candidate c, let $P_{E,c}(r)$ be the probability that c wins—under a given voting rule—in an election chosen uniformly at random from $R(E, r)$. Unfortunately, since #Swap-Bribery is #P-hard for both our rules (and none of our FPT algorithms is practical enough), instead of computing these values exactly, we resorted to sampling. Specifically, for each election E and each normalized swap distance $r \in \{0.05, 0.1, \ldots, 1\}$ we sampled 500 elections from $R(E, r)$ and for each candidate recorded the proportion of elections where he or she won[2] (see Appendix B.2 for the sampling procedure). For each election, we quantified the robustness of its winner by identifying the smallest swap distance r, among the considered ones, for which he or she has a winning probability below 50%. We refer to this value as the 50%-*winner threshold* (or, threshold, for short).

Results. In the following, we present several findings from our experiments, each followed by supporting arguments. We start by analyzing the relation between the 50%-winner threshold and two other measures of winner robustness, namely, the score difference between the winner and the runner-up (i.e., the candidate ranked in the second place) and the minimum number of swaps of adjacent candidates that are necessary to change the election winner (this is simply the optimal cost of a Destructive Swap-Bribery with unit prices; see the work of Shiryaev et al. [29][3]). To this end, let us turn to Figure 2a (for Plurality)

[2]By Hoeffding's inequality, the probability that the estimated winning probability for a given candidate at some swap distance deviates by more than 10% from the true one can be upper bounded by 0.1%.

[3]Notably, both Plurality Destructive Swap-Bribery and Borda Destructive Swap-Bribery are in P [29]. For both problems, it suffices to iterate over all candidates $d \neq c$ (where c is the original winner) and calculate the minimum cost of modifying the election so that d has a higher score than c.

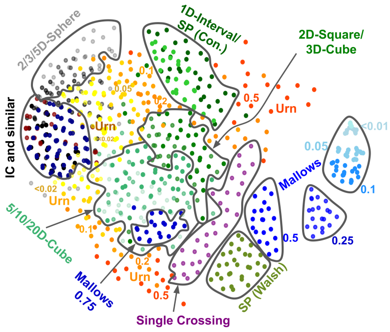
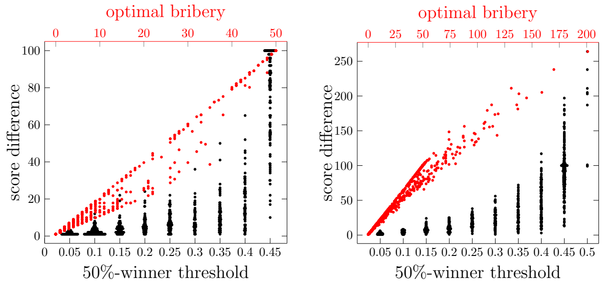


Figure 1: The map of elections, due to Szufa et al. [30]. Each point corresponds to an election and its color gives the model from which it came. Generally, the closer two points are, the more similar are the corresponding elections in their metric.



(a) Plurality

(b) Borda

Figure 2: Each election is represented by a red dot and a black dot. The y-coordinate of both dots gives the difference between the scores of the winner and the runner-up in the election. The x-coordinate of the black dot gives the 50%-winner threshold (perturbed, if many elections would overlap), while the x-coordinate of the red dot gives the minimum cost of a destructive swap bribery.

and Figure 2b (for Borda), which are both split into a black and a red part. In both parts, each election is represented as a dot whose y-coordinate is the score difference between the winner and the runner up, and the x-coordinate is either:

1. the 50%-winner threshold (in the black part, perturbed a bit if many elections were to take the same place), or
2. the minimum cost of a successful destructive swap bribery (in the red part).

(Each election is represented by two dots, one in the black part and one in the red part.)

Finding 1. *The score difference between the winner and the runner-up strongly correlates with the minimum cost of a successful destructive bribery. In contrast to this, the score difference has a limited predictive value for the* 50%*-winner threshold.*

Examining the figures, we see that the score difference is very strongly correlated with the cost of the destructive swap bribery, but that correlation between the score difference and the 50%-winner threshold is far less pronounced. Indeed, the same score difference may lead to a wide range of 50%-winner thresholds (e.g., for Plurality a score difference of 10 may lead to the threshold being anything between 0.1 and 0.4). Thus our framework adds a new dimension to the robustness analysis of election winners.

We now turn to the differences between Plurality and Borda when it comes to the typical robustness of election winners:

Finding 2. *The Borda winner of an election is usually more robust against random swaps than the Plurality winner.*

For Borda, 230 elections out of the 800 considered have a 50%-winner threshold of 0.45, while every other threshold occurs fewer than 90 times. In contrast, for Plurality the distribution is more uniform (with small spikes of around 110 elections at thresholds 0.1 and 0.45). So, Plurality elections are more likely to change results after relatively few swaps than the Borda ones. Two explanations are that (a) under Plurality there can be "strong contenders" who do not win, but who are often ranked close to the first place and, thus, can overtake the original winner after a few swaps, and (b) the Plurality winner has the highest

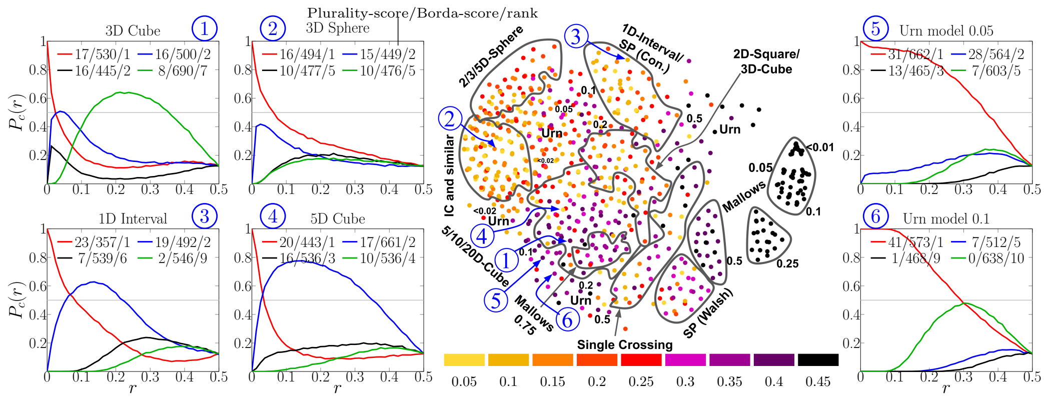


Figure 3: Map of elections visualizing the 50%-winner threshold for Plurality and six plots showing $P_{E,c}(r)$ as a functon of r, for six selected elections and the four most successful candidates in each (see the paragraph preceding Finding 4 for details).

chance of losing points, as he or she is ranked first most often. Under Borda, the candidates usually have similar chances of both gaining and losing a point with a single swap.

From now on, we focus on Plurality, but most of our conclusions also apply to Borda (we do mention some differences though; for details, see Appendix B.3). In Appendix B.4, we also show that our framework can be useful for voting rules based on pairwise comparisons, that is, we consider a variant of the Copeland rule.

In Figure 3, we show the map of elections for Plurality, with colors corresponding to each election's 50%-winner threshold. The figure also includes six plots, each showing the values of $P_{E,c}(r)$ for four candidates in six selected elections (we discuss them later).

Finding 3. *Positions of elections on the map correlate with their* 50%*-winner thresholds. Elections sampled from the same model tend to have similar thresholds.*

Consider the map in Figure 3. As we move from left to right, the 50%-winner threshold tends to increase. Not surprisingly, it is low for IC elections (as they are completely random, it is natural that few changes can affect the result) and it is high for Mallows elections with low ϕ (most preference orders in these elections are identical, up to a few swaps). For urn elections, the threshold tends to increase with parameter α (as the votes become less varied with larger α). Interestingly, the threshold is somewhat more varied among tD-Cube elections (as compared to the other models), and one can notice that for 1D-Interval elections it tends to be slightly lower than for higher-dimensional tD-Cube ones (this effect is much stronger for Borda). Yet, typically elections generated from a given model (with a given parameter) tend to have similar threshold values.

For further insights, we turn to the six plots in Figure 3. Each of them regards a particular election and four of its candidates. The candidates are marked with colors and the original winner is always red. For each considered election E and each candidate c in the plot, we show $P_{E,c}(r)$ for values of r between 0 and 0.5 (specifically, for these six elections, we estimated $P_{E,c}(r)$ for $r \in \{0.0125, 0.025, \ldots, 0.5\}$ using 10'000 samples in each case). We limited the range of r because above 0.5, the votes are becoming similar to the reverses of the original ones. For each of the six elections, we sorted the candidates with respect to $\max_{r \in \{0, 0.0125, \ldots, 0.5\}} P_{E,c}(r)$ and chose the top four to be included in the plot. For each of the candidates, in the legend we provide his or her Plurality score, Borda score, and the rank in the original election (we use the Borda scores in further discussions). The elections were chosen to show interesting phenomena (thus the patterns they illustrate are not always the most common ones, but are not outliers either). The following discussion refines the observations from Finding 1.

Finding 4. *Winners winning by a small margin are not necessarily close to losing. Winners winning by a large margin are robust but not necessarily very robust winners.*

In Elections 1 to 4, the winners are very sensitive to random swaps: The blue candidate already wins a considerable proportion of elections even if only a 0.0125 fraction of possible swaps are applied (i.e., about half a swap per vote, on average), and the red candidate quickly drops below 50% winning probability. It is quite surprising that so few random swaps may change the outcome with fairly high probability. There are also differences among these four elections. For example, in Elections 1 and 2 the candidates have similar scores, but in Election 2 the red candidate stays the most probable winner until swap distance 0.4, whereas in Election 1, the most probable winner changes quite early. The plots for Elections 3 and 4 are similar to that for Election 1, but come from tD-Cube elections of different dimension; this pattern appears in elections from other families of distributions too, but less commonly.

In Elections 1 to 4, the original winner has at most four Plurality points of advantage over the next candidate, so one could argue that scores suffice to identify close elections. Yet, in Election 5 the difference between the scores of the winner and the runner-up is 3, but the red candidate stays a winner with probability greater than 50% until swap distance 0.35. Thus, looking only at the scores can be misleading. Nonetheless, if the score difference is large (say, above 25), the 50%-winner threshold is always above 0.2 (see Figure 2a). But, as witnessed in Election 6, even in such seemingly clear elections, around 10% of random swaps suffice to change the outcome with a non-negligible probability.

Finding 5. *The score of a non-winning candidate has a limited predictive value for his or her probability of winning if some random swaps are performed.*

Perhaps surprisingly, in some elections the most probable winner at some (moderately low) swap distance is not necessarily ranked highly in the original election. For instance, in Election 1 the green candidate is originally ranked seventh (out of ten candidates), but becomes the most probable winner already around swap distance 0.1. Here, this can be explained by the fact that he or she has a significantly higher Borda score than the other candidates. So, he or she is ranked highly in many votes and can reach the top positions with only a few swaps. Yet, not all patterns can be explained this way. For example, in Elections 1 and 2, the first two candidates have similar Plurality and Borda scores but still behave quite differently, even at small swap distances.

4.2 Real-World Cycling Data

In this section, we consider a dataset of real-world elections obtained from cycling races. We use a dataset of 67 elections, each consisting of 20 candidates and between 20 and 23 voters. The data comes from the Tour de France (TdF) and the Giro d'Italia (GdI) races, and was obtained from `procyclingstats.com`. Both the TdF and the GdI are annual cycling competitions where riders compete in multiple stages. We created a separate election for each of the last 100 editions of the two contests, with the riders as candidates and the stages as voters (which rank the riders according to their finishing times in the corresponding stages). We deleted all elections which had fewer than 20 candidates that appeared in all the votes. Subsequently, to obtain comparable results, for each election, we deleted all but the 20 candidates that appeared in all the votes and had the highest Borda scores.

For each election E out of the 67 that remained, we sampled 10'000 elections at swap distance $r \in \{0.0125, 0.025, \ldots, 0.5\}$ to compute $P_{E,c}(r)$ for each candidate c. Overall, the

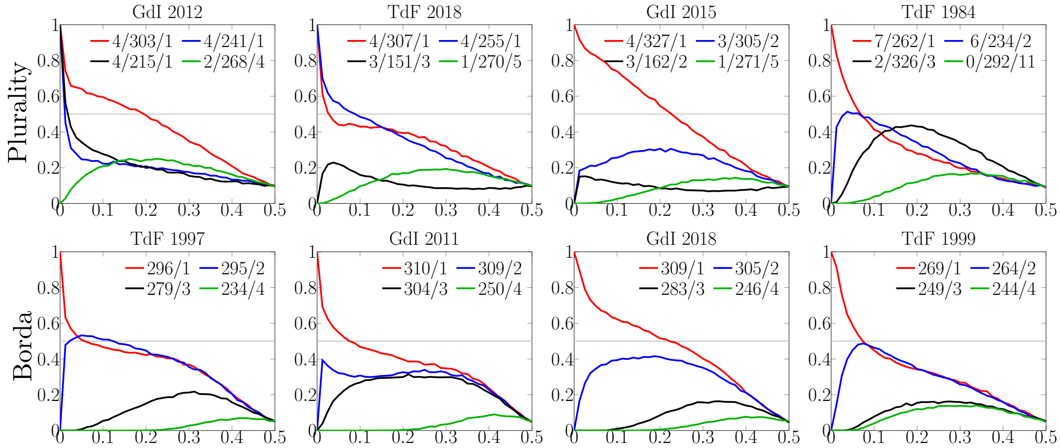


Figure 4: Each row presents results for four elections from the cycling dataset. The top row regards the Plurality rule and the bottom one regards Borda. We plot $P_{E,c}(r)$ as a function of r for the four most successful candidates under the given voting rule. In the legend, for each candidate, the first entry contains its score under the relevant voting rule and the last entry gives its rank in the original election. For Plurality, the second entry contains the Borda score of the candidate.

results are surprisingly similar to the ones on the synthetic dataset in the sense that most of the phenomena we observed and described earlier can also be witnessed in some of the 67 elections. In the following, we use our approach to analyze four elections for the Plurality rule and four elections for the Borda rule. The respective elections are displayed in Figure 4.

Plurality. GdI 2012 and TdF 2018 are two examples of elections with tied Plurality winners. Both these elections look similar when considering the Plurality scores and the Borda scores of the candidates, but while the red candidate in GdI 2012 seems to be a more robust winner than the other two, the red and blue candidate in TdF 2018 look equally strong (but one could argue that the blue one has some small advantage).

In both GdI 2015 and TdF 1984, the red candidate wins one more race than the blue one and has a similar lead in the Borda score. However, while the red candidate is a robust and strong winner in GdI 2015, this is not the case in TdF 1984. Remarkably, in TdF 1984, the black candidate, which is initially five points behind the red candidate, already wins some elections at swap distance 0.0125 and around 40% of elections at swap distance 0.125.

Borda. In all four selected elections, the Borda scores of the top candidates are relatively similar. In particular, in TdF 1997 and GdI 2011, the red candidate wins by one point, while in GdI 2018 and TdF 1999, the red candidate wins by four or five points. Looking at the plots, however, the elections do not longer look that similar: GdI 2011 and 2018 seem to be quite close but the victory of the red candidate seems to be nevertheless relatively robust. In contrast to this, the blue candidate has the same probability of winning as the red candidate at swap distance 0.0375 in TdF 1997 and at swap distance 0.075 in TdF 1999.

5 Conclusions

We have shown that the counting variants of Swap-Bribery have high worst-case complexity, but, nonetheless, are very useful for analyzing the robustness of election winners. In particular, we have observed that neither the scores of the candidates nor Destructive-Swap-Bribery costs suffice to evaluate their strengths. We left open the issue of establishing the complexity of Borda #Swap-Bribery parameterized by the swap radius.

References

[1] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *Proceedings of AAAI-2010*, pages 697–702, 2010.

[2] A. Barvinok. A polynomial time algorithm for counting integral points in polyhedra when the dimension is fixed. *Math. Oper. Res.*, 19(4):769–779, 1994.

[3] D. Baumeister and T. Hogrebe. Complexity of election evaluation and probabilistic robustness: Extended abstract. In *Proceedings of AAMAS-2020*, pages 1771–1773, 2020.

[4] D. Baumeister, T. Hogrebe, and L. Rey. Generalized distance bribery. In *Proceedings of AAAI-2019*, pages 1764–1771, 2019.

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

[6] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. Prices matter for the parameterized complexity of shift bribery. *Inf. Comput.*, 251:140–164, 2016.

[7] R. Bredereck, P. Faliszewski, R. Niedermeier, and N. Talmon. Complexity of shift bribery in committee elections. In *Proceedings of AAAI-2016*, pages 2452–2458, 2016.

[8] R. Bredereck, P. Faliszewski, A. Kaczmarczyk, R. Niedermeier, P. Skowron, and N. Talmon. Robustness among multiwinner voting rules. *Artif. Intell.*, 290:103403, 2021.

[9] G. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.

[10] M. Brill, U. Schmidt-Kraepelin, and W. Suksompong. Refining tournament solutions via margin of victory. In *Proceedings of AAAI-2020*, pages 1862–1869, 2020.

[11] D. Cary. Estimating the margin of victory for instant-runoff voting. Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trushworthy Elections, 2011.

[12] V. Conitzer. Eliciting single-peaked preferences using comparison queries. *J. Artif. Intell. Res.*, 35:161–191, 2009.

[13] R. Curticapean and D. Marx. Complexity of counting subgraphs: Only the boundedness of the vertex-cover number counts. In *Proceedings of FOCS-2014*, pages 130–139, 2014.

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

[15] E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *Proceedings of WINE-2010*, pages 473–482, 2010.

[16] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proceedings of SAGT-2009*, pages 299–310, 2009.

[17] E. Elkind, P. Faliszewski, S. Gupta, and S. Roy. Algorithms for swap and shift bribery in structured elections. In *Proceedings of AAMAS-2020*, pages 366–374, 2020.

[18] P. Faliszewski, P. Manurangsi, and K. Sornat. Approximation and hardness of shift-bribery. In *Proceedings of AAAI-2019*, pages 1901–1908, 2019.

[19] J. Flum and M. Grohe. The parameterized complexity of counting problems. *SIAM J. Comput.*, 33(4):892–922, 2004.

[20] N. Hazon, Y. Aumann, S. Kraus, and M. Wooldridge. On the evaluation of election outcomes under uncertainty. *Artif. Intell.*, 189:1–18, 2012.

[21] A. Kaczmarczyk and P. Faliszewski. Algorithms for destructive shift bribery. *Auton. Agents Multi-Agent Syst.*, 33(3):275–297, 2019.

[22] B. Kenig and B. Kimelfeld. Approximate inference of outcomes in probabilistic elections. In *Proceedings of AAAI-2019*, pages 2061–2068, 2019.

[23] D. Knop, M. Koutecky, and M. Mnich. Voting and bribing in single-exponential time. *ACM Trans. Econ. Comput.*, 8(3):12:1–12:28, 2020.

[24] H. Lenstra, Jr. Integer programming with a fixed number of variables. *Math. Oper. Res.*, 8(4):538–548, 1983.

[25] T. Magrino, R. Rivest, E. Shen, and D. Wagner. Computing the margin of victory in IRV elections. Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trushworthy Elections, 2011.

[26] C. Maushagen, M. Neveling, J. Rothe, and A.-K. Selker. Complexity of shift bribery in iterative elections. In *Proceedings of AAMAS-2018*, pages 1567–1575, 2018.

[27] OEIS Foundation Inc. The on-line encyclopedia of integer sequences, 2020. URL http://oeis.org/A008302.

[28] D. Peters and M. Lackner. Preferences single-peaked on a circle. *J. Artif. Intell. Res.*, 68:463–502, 2020.

[29] D. Shiryaev, L. Yu, and E. Elkind. On elections with robust winners. In *Proceedings of AAMAS-2013*, pages 415–422, 2013.

[30] S. Szufa, P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. Drawing a map of elections in the space of statistical cultures. In *Proceedings of AAMAS-2020*, pages 1341–1349, 2020.

[31] L. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8(2): 189–201, 1979.

[32] T. Walsh. Generating single peaked votes. Technical Report arXiv:1503.02766 [cs.GT], arXiv.org, March 2015.

[33] K. Wojtas and P. Faliszewski. Possible winners in noisy elections. In *Proceedings of AAAI-2012*, pages 1499–1505, 2012.

[34] L. Xia. Computing the margin of victory for various voting rules. In *Proceedings of EC-2012*, pages 982–999, 2012.

[35] Y. Yang, Y. Raj Shrestha, and J. Guo. On the complexity of bribery with distance restrictions. *Theor. Comput. Sci.*, 760:55–71, 2019.

[36] A. Zhou and J. Guo. Parameterized complexity of shift bribery in iterative elections. In *Proceedings of AAMAS-2020*, pages 1665–1673, 2020.

Niclas Boehmer
Algorithmics and Computational Complexity, TU Berlin
Berlin, Germany
Email: niclas.boehmer@tu-berlin.de

Robert Bredereck
Humboldt-Universität zu Berlin
Berlin, Germany
Email: robert.bredereck@hu-berlin.de

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

Rolf Niedermeier
Algorithmics and Computational Complexity, TU Berlin
Berlin, Germany
Email: rolf.niedermeier-berlin.de

Appendix

A Missing Proofs from Section 3

In this section, we provide missing details and proofs from Section 3.

A.1 Auxiliary Algorithms

In this section we provide a number of polynomial-time algorithms for solving problems of the following form: Given an election and a particular budget (or, number of swaps) compute the number of ways of performing exactly this many swaps so that the election has some given shape (e.g., all the voters ranks the same given candidate on top). We refer to such problems as *voter group contribution counting problems*

A.1.1 Swap Contribution for Plurality (Unit Prices).

Given an election $E = (C, V)$, a budget r, and a distinguished candidate $p \in C$, $\mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}(V, r, p)$ denotes the number of possibilities to perform exactly r swaps so that p is the top choice of every voter within V.

Lemma 7. *One can compute $vgc^{Swap}_{Plurality}(V, r, p)$ in time $O(n \cdot m^4)$.*

Proof. Let $C = \{p, c_2, \ldots, c_m\}$, $V = \{v_1, \ldots, v_n\}$, r, and p be given as described above. We define the following dynamic programming table L. An entry $L[i, r']$ denotes the number of possibilities to perform exactly r' swaps within the first i votes in V so that p is the top choice for these i voters.

Let $r^*(v)$ denote the number of swaps required to push candidate p to the top position in vote v. We initialize the table via:

$$L[1, r'] = \begin{cases} 0 & \text{if } r' < r^*(v_1) \\ Y(m-1, r' - r^*(v_1)) & \text{otherwise,} \end{cases}$$

where $Y(m, k)$ denotes the number of permutations of swap distance k from a given permutation with m elements (algorithms for computing this value in polynomial time are well known). We update the table with increasing i via

$$L[i, r'] = \sum_{r^*(v_i) \le r'' \le r'} Y(m-1, r'' - r^*(v_i)) \cdot L[i-1, r' - r''].$$

Finally, $\mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}(V, r, p) = L[n, r]$ gives the solution.

The initialization is correct, because we have to push candidate p to the top position at cost $r^*(v_1)$. This fixes the first position and the other $m-1$ positions can be freely rearranged. Naturally, there are $Y(m-1, r' - r^*(v_1))$ possibilities to do this. Similarly, in the update step we sum over all possibilities to distribute our r' swaps among the first $i-1$ votes and the ith vote (again, we need at least $r^*(v_i)$ swaps to push p to the top). In each case, the number of possibilities is the product of all possibilities to spend r'' for voter i and $r' - r''$ swaps for the first $i-1$ voters.

The table is of dimension $O(n \cdot m^2)$. Computing each single table entry can be done in time $O(m^2)$: Precomputing the table Y with all entries takes $O(m^2)$ time (see also Appendix B.2 where a recurrence is given) and with this being done, computing each single table entry of L takes $O(m^2)$ time since there are at most m^2 values for r'' and for each such value we have to do only a constant number of arithmetic operations. □

A.1.2 Shift Contribution for Plurality (Arbitrary Prices Encoded in Unary.

At first, let us consider the constructive variant of SHIFT-BRIBERY, where we can shift forward the preferred candidate p.

Let $E = (C, V)$ be an election where every voter prefers the same candidate $d \in C$. Let r be a budget, $p \in C$ be a distinguished candidate, and s be an integer score value. Moreover, we are given some cost function $c\colon (V, \mathbb{N}) \to \mathbb{N}$ describing the costs $c(v, \ell)$ of shifting p by ℓ positions forward. We define $\mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}(V, r, p, s)$ as the number of possibilities to shift candidate p forward at total costs r within V so that p is ranked at the top position exactly s times.

Lemma 8. *One can compute $vgc^{\mathit{Shift+}}_{\mathit{Plurality}}(V, r, p, s)$ in time $O(n^2 \cdot r^2)$.*

Proof. We assume that $V = \{v_1, \ldots, v_n\}$ and compute $\mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}$ using standard dynamic programming. Let $L[j, s', r']$ be the number of ways to shift p forward in the first j votes in V at total cost of r', so that p obtains s' additional Plurality points.

We introduce two auxiliary functions, $\tau(v, x)$ and $\nu(v, x)$. Function $\nu(v, x)$ indicates whether spending cost x for voter v is valid (e.g., we cannot spend more than necessary to push p to the top position) and function $\tau(v, x)$ indicates whether spending costs x for voter v is successful (i.e., pushes p to the top position). Formally, we have:

$$\nu(v, x) = \begin{cases} 1 & \text{it is valid to spend cost } x \text{ at voter } v, \\ 0 & \text{otherwise.} \end{cases}$$

$$\tau(v, x) = \begin{cases} 1 & p \text{ becomes top choice in } v \text{ at cost } x, \\ 0 & \text{otherwise.} \end{cases}$$

We initialize our table with:

$$L[1, s', r'] = \begin{cases} \nu(v_1, r') \cdot \tau(v_1, r') & s' = 1 \\ \nu(v_1, r') \cdot (1 - \tau(v_1, r')) & s' = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We update $L[j, s', r']$ for $j > 1$ via $L[j, s', r'] =$

$$\sum_{r'' \leq r'} \big(L[j-1, s'-1, r''] \cdot \tau(v_j, r' - r'') \cdot \nu(v_j, r' - r'') \\ + L[j-1, s', r''] \cdot (1 - \tau(v_j, r' - r'')) \cdot \nu(v_j, r' - r'') \big).$$

The table L is of size $n^2 \cdot r$. Computing a single table entry requires at most $2r$ table lookups and at most $2r$ arithmetic operations. □

Let us now consider the destructive case, where we can push a given candidate backward. Consider an election $E = (C, V)$ with two distinguished candidates, p and d, such that every voter either prefers p the most while ranking candidate d in the second position, or prefers d the most. Let r be the budget, and s be an integer score value. Moreover, we are given some cost function $c\colon (V, \mathbb{N}) \to \mathbb{N}$ describing the costs $c(v, \ell)$ of shifting p by ℓ positions backward. We define $\mathrm{vgc}^{\mathrm{Shift-}}_{\mathrm{Plurality}}(V, r, p, s)$ as the number of possibilities to shift candidate p backward at total costs r within V such that p is ranked exactly s times at the top position.

Lemma 9. *One can compute $vgc^{\mathit{Shift-}}_{\mathit{Plurality}}(V, r, p, s)$ in time $O(n^2 \cdot r^3)$.*

Proof. We assume that $V = \{v_1, \dots, v_n\}$ and compute $\mathrm{vgc}^{\mathrm{Shift-}}_{\mathrm{Plurality}}$using yet again standard dynamic programming, defining our table L as follows. An entry $L[j, s', r']$ contains the number of ways to shift p backward in the first j votes from V at total cost of r', so that p ends up with exactly s' points.

As in the constructive case, we introduce two auxiliary functions, $\tau(v, x)$ and $\nu(v, x)$. Function $\nu(v, x)$ indicates that spending cost x for voter v is valid and function $\tau(v, x)$ indicates that spending cost x for voter v is successful (pushes d to the top position). Formally:

$$\nu(v, x) = \begin{cases} 1 & \text{it is valid to spend cost } x \text{ at voter } v \\ 0 & \text{otherwise.} \end{cases}$$

$$\tau(v, x) = \begin{cases} 1 & d \text{ becomes top choice in } v \text{ at cost } x \\ 0 & \text{otherwise.} \end{cases}$$

We initialize our table with:

$$L[1, s', r'] = \begin{cases} \nu(v_1, r') \cdot (1 - \tau(v_1, r')) & s' = 1 \\ \nu(v_1, r') \cdot \tau(v_1, r') & s' = 0 \\ 0 & \text{otherwise.} \end{cases}$$

We update $L[j, s', r']$ for $j > 1$ via $L[j, s', r'] =$

$$\textstyle\sum_{r'' \leq r'} \big(L[j-1, s', r'']\tau(v_j, r' - r'')\nu(v_j, r' - r'') \\ + L[j-1, s'-1, r''](1 - \tau(v_j, r' - r''))\nu(v_j, r' - r'')\big).$$

The table L is of size $n^2 \cdot r$. Computing a single table entry requires at most $2r$ table lookups and at most $2r$ arithmetic operations. □

A.2 Missing Details for the Proof of Theorem 2

For the proof of Theorem 2, we need to discuss how to compute the table T.

Lemma 10. *Table T can be computed in time $O(n \cdot m^7)$.*

Proof. We initialize the table by setting $T[1, 1, r'] = \mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}(V_1, r', p)$ and $T[1, \ell, r'] = 0$, $\forall \ell > 1$. The table is filled with increasing i by setting $T[i, \ell, r']$ to be:

$$\sum_{\ell' < \ell, r'' \leq r'} (T[i-1, \ell', r''] \cdot \mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}(V_i, r' - r'', c_\ell)).$$

Note that the initialization is correct by the definition of $\mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}(V, r, c)$ and the fact that the first group must consistently vote for $c_1 = p$. For updating the table, we sum up over all possibilities to split the swap budget r' between the ith voter group and first $i-1$ voter groups, in combination with each possible candidate $c_{\ell'}$ that may have been pushed to the top position by all voters of group $i-1$.

Computing table T requires filling in $m \cdot m \cdot nm^2$ entries, and each entry takes $O(m^3)$ time (due to the number of terms in the sum in the update step). All in all, the algorithm takes time $O(n \cdot m^7)$, assuming that each of at most m functions $\mathrm{vgc}^{\mathrm{Swap}}_{\mathrm{Plurality}}$ was computed in time $O(n \cdot m^4)$. □

A.3 Proof of Theorem 3

Theorem 3 (★). Plurality #Swap-Bribery *is* #P*-hard even for a single voter and unary-encoded prices.*

Proof. In an instance of #Linear Extensions we are given a set $Z = \{z_1, \ldots, z_m\}$ of items and a set $O \subseteq X \times X$ of constraints; we ask for the number of linear orders over Z such that for each constraint $(x, y) \in O$, x precedes y. We reduce this problem to Plurality #Swap-Bribery with 0/1 prices (i.e., each swap either has a unit cost or is free) and budget $r := 0$ (if one preferred to avoid zero prices, then doing so would require only a few adaptations in the proof).

Given an instance of #Linear Extensions, as specified above, first we compute a single order $\succ$ that is consistent with the constraints (doing so is easy via standard topological sorting; if no such order exists, then we return zero and terminate). Next, we form an election E with candidate set $C = \{p\} \uplus Z$ and a single vote v, where p is ranked first and all the other candidates are ranked below, in the order provided by $\succ$. We set the swap prices so that:

1. For each candidate $z \in Z$, the price for swapping him or her with p is one.
2. For each pair $(x, y) \in O$, the price for swapping x and y is one.
3. All other prices are zero.

We form an instance of Plurality #Swap-Bribery with this election, prices, and budget $r := 0$. We make a single query regarding this instance and output the obtained value.

To see that the reduction is correct, we notice that for every preference order that v may have after performing swaps of price zero, it holds that (a) p is ranked first (because swapping p out of the first position has nonzero price) and (b) for all pairs $(x, y) \in O$, x is ranked ahead of y (because swapping x and y has nonzero cost). On the contrary, for every linear order $\succ'$ that is consistent with O, it is possible to transform the preference order of v so that p is ranked first, followed by members of Z in the order specified by $\succ'$ (for each two candidates $x, y \in Z$ such that $x \succ y$ but $y \succ' x$, the cost of swapping them is zero, and if we have not transformed our vote into the desired form yet, then there are always two such candidates that are ranked consecutively). □

A.4 Proof of Theorem 4

Theorem 4 (★). *For unary-encoded prices, both the constructive and the destructive variant of* Plurality #Shift-Bribery *are in* P.

Proof for the constructive case. A core observation for our algorithm is that, under #Shift-Bribery, every voter will either vote for p (if we shift p to the top position) or for its original top choice. This allows us to group the voters according to their top choices as follows. Let $(V_0, V_1, V_2, \ldots, V_{m'})$, where $m' \leq m - 1$, be a partition of voters into groups so that (in the original election) every two voters within each group V_i share the same top choice, while every two voters from different groups have different top choices; additionally, we require that the voters in group V_0 rank the distinguished candidate p on the top position.

For each candidate c, let $s(c)$ be the original score of c. The idea of our algorithm is to count, for each possible final score $s^* \geq s(p)$ of p, the number of ways to spend the given budget r, so that p obtains s^* points while no other candidate obtains more than s^* points. For each possible s^* we create one global dynamic programming table T_{s^*}.

Our global tables are defined as follows. An entry $T_{s^*}[i, s', r']$ contains the number of ways to shift p forward in the voter groups $V_1, \ldots, V_i$ at total cost of r', so that p obtains

s' additional points while no top choice from any voter group $V_1, \ldots, V_i$ receives more than s^* points.

To compute the values in table T, we will use the following *local counting problem*, maintaining the *voter group contribution*: Given a voter group V', a budget r', a distinguished candidate p, and a score s', compute the number of possibilities $\mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}(V', r', p, s')$ to shift candidate p by in total r' positions within V', so that p is ranked exactly s' times at the top position. This number can be computed in polynomial time (see Lemma 8 in Appendix A.1).

The initialization of T_{s^*} is straight-forward, by setting:

$$T_{s^*}[1, s', r'] = \mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}(V_1, r', p, s')$$

when $|V_1| - s' \leq s^*$, and by setting $T_{s^*}[1, s', r'] = 0$ when $|V_1| - s' > s^*$ (the condition $|V_1| - s' \leq s^*$ is to ensure that the top-ranked candidate of the voters from group V_1 obtains no more than s^* points). We update the tables for $i > 1$ by setting $T_{s^*}[i, s', r']$ to be:

$$\textstyle\sum_{s''=|V_i|-s^*}^{s'} \sum_{r'' \leq r'} \big(T_{s^*}[i-1, s'-s'', r'-r''] \\ \cdot \mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}(V_i, r'', p, s'')\big).$$

(Again, the lower bound on s'' ensures that the candidate the voters from group V_i vote for (if not p) obtains no more than s^* points.) It is not hard to see that this indeed computes the values in the table correctly, without double-counting.

Assuming $\mathrm{vgc}^{\mathrm{Shift+}}_{\mathrm{Plurality}}$ and global tables are computed correctly, it is not hard to verify that the overall solution is:

$$\sum_{s^*:=s(p)}^{s(p)+r} T_{s^*}[m', s^* - s(p), r].$$

Indeed, between two different "guesses" of the final score s^*, double-counting is impossible. □

Proof for the destructive case. The main ideas behind the destructive case are very similar to those behind the constructive one. The core observation is that under destructive PLURALITY #SHIFT-BRIBERY every voter will either vote for the distinguished candidate p (if p is the voter's top choice and is not shifted backward) or for some other candidate $d \neq p$ (either if p were not the voter's top choice but d were, or if p were the top choice but was shifted backward so that d, which was originally in the second position, was moved to the top). In either case, we call candidate d the non-p choice of voter v. This allows us to group all voters according to their non-p choices as follows. Let $(V_1, V_2, \ldots, V_{m'}), m' \leq m$, be a partition of voters into groups V_i such that every two voters within the same group V_i have the same non-p choice while every two voters from different groups have different non-p choices.

Let $s(c)$ be the original score of candidate c. The idea of our algorithm is to count, for each possible final score $s^* \leq s(p)$ of p the number of ways to spend the given budget r so that p obtains s^* points while at least one other candidate obtains more than s^* points. For each possible s^* we create one global dynamic programming table T_{s^*}.

Our global tables are defined as follows. An entry $T_{s^*}[i, s', r', \mathrm{false}]$ contains the number of ways to shift p backward in the voter groups $V_1, \ldots, V_i$ at total cost of r', so that p obtains s' points from $V_1, \ldots, V_i$ while no non-p choice of any voter group $V_1, \ldots, V_i$ receives more than s^* points. An entry $T_{s^*}[i, s', r', \mathrm{true}]$ contains the number of ways to shift p backward in the voter groups $V_1, \ldots, V_i$ at total cost of r', so that p obtains s' points from $V_1, \ldots, V_i$ while a non-p choice of at least one voter group from $V_1, \ldots, V_i$ receives more than s^* points.

To compute the entries of the table T, we will use the following *local counting problem* maintaining the *voter group contribution*: Given a voter group V', a budget r', a distinguished candidate p, and a score s', compute the number $\mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V', r', p, s')$ of possibilities to shift candidate p backward at total costs r' within V', so that p is ranked exactly s' times at the top position. This number can be computed in polynomial time (see Lemma 9 in Appendix A.1).

The initialization of T_{s^*} is as follows:

1. If $|V_1| - s' \leq s^*$, then
$$T_{s^*}[1, s', r', \text{false}] = \mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V_1, r', p, s'),$$
and otherwise $T_{s^*}[1, s', r', \text{false}] = 0$.
2. If $|V_1| - s' > s^*$ then
$$T_{s^*}[1, s', r', \text{true}] = \mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V_1, r', p, s'),$$
and otherwise $T_{s^*}[1, s', r', \text{true}] = 0$.

We compute the table entries for $i > 1$ as follows. We set $T_{s^*}[i, s', r', \text{false}]$ to be
$$\textstyle\sum_{s''=|V_i|-s^*}^{s'} \sum_{r'' \leq r'} \big(T_{s^*}[i-1, s'-s'', r'-r'', \text{false}] \cdot \mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V_i, r'', p, s'')\big).$$
(The lower bound on s'' ensures that the non-p choice obtains no more than s^* points.) And we set $T_{s^*}[i, s', r', \text{true}]$ to be:
$$\begin{aligned}\sum_{s'' \leq s', |V_i|-s''>s^*} \sum_{r'' \leq r'} &(T_{s^*}[i-1, s'-s'', r'-r'', \text{false}] \\ &\cdot \mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V_i, r'', p, s'')) \\ + \sum_{s'' \leq s'} \sum_{r'' \leq r'} &(T_{s^*}[i-1, s'-s'', r'-r'', \text{true}] \\ &\cdot \mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}(V_i, r'', p, s''))\end{aligned}$$

The first two sums account for the case that the non-p choice of group V_i is the first candidate non-p candidate to obtain more than s^* points. The second two sums account for the case where already some non-p choice of some previous group obtained more than s^* points. One can verify that this indeed computes the table correctly without double-counting.

Assuming $\mathrm{vgc}^{\text{Shift}-}_{\text{Plurality}}$ and global tables are computed correctly, it is not hard to verify the the overall solution is:
$$\sum_{s^*:=s(p)-r}^{s(p)} T_{s^*}[m', s^*, r, \text{true}].$$
Indeed, between two different "guesses" of the final score s^*, double-counting is impossible. □

A.5 Proof of Theorem 5

Theorem 5 (★). *Both the constructive and the destructive variant of* Borda #Shift-Bribery *are* #P*-hard and* #W[1]*-hard when parameterized by the number of voters.*

Proof. For the constructive case, it suffices to follow the proof of Bredereck et al. [7]. For the destructive case, we give a Turing reduction from the #Multicolored Independent Set problem, which is well-known to be #W[1]-complete (indeed, #Independent Set is equivalent to #Clique, which is a canonical #W[1]-complete problem; the multicolored variants of these problem remain #W[1]-complete).

Let $I = (G, h)$ be an instance of #Multicolored Independent Set, where $G = (V(G), E(G))$ is a graph where each vertex has one of h colors; we ask for the number of size-h independent sets (i.e., sets of vertices such that no two vertices have a common edge) such that each vertex has a different color. Without loss of generality, we assume that there are no edges between vertices of the same color and that the number of vertices of each color is the same, denoted by n. For each color $\ell \in [h]$, let $V(\ell) := \{v_1^\ell, \ldots, v_n^\ell\}$ denote the set of vertices with color ℓ. For each vertex $v_i^\ell \in V(G)$, let $E(v_i^\ell)$ denote the set of edges incident to this vertex. Finally, let $\Delta := \max_{v \in V(G)} |E(v)|$ be the highest degree of a vertex in G.

Our reduction proceeds as follows. Let $r := h(n(\Delta + 1) + \Delta)$ be our shift radius. We form an election with the following candidates. First, we add a candidate d, who will be the original winner of the election, and we treat sets $V(G)$ and $E(G)$ as sets of vertex and edge candidates. For each vertex $v_i^\ell \in V(G)$, we form a set $F(v_i^\ell)$ of $\Delta - |E(v_i^\ell)|$ *fake-edge* candidates, so we will be able to pretend that all vertices have the same degree; we write $F(G)$ to denote the set of all fake-edge candidates. Next, we form a *blocker* candidate b and a set B of r additional blocker candidates, whose purpose will be to limit the extent to which we can shift d in particular votes. Finally, we let $X = \{x_1, \ldots, x_5\}$ be a set of five candidates that we will use to fine-tune the scores of the other candidates. Altogether, the candidate set is:

$$C := \{d\} \uplus V(G) \uplus E(G) \uplus F(G) \uplus B \uplus \{b\} \uplus X.$$

For each vertex v_i^ℓ, by $H(v_i^\ell)$ we mean the (sub)preference order where v_i^ℓ is ranked on top and is followed by the candidates from $E(v_i^\ell) \uplus F(v_i^\ell)$ in some arbitrary order. We write $\overleftarrow{H(v_i^\ell)}$ to denote the corresponding reverse order. We form the following $4h + 2$ voters:

1. For each color $\ell \in [h]$, we introduce voters $e(\ell)$ and $f(\ell)$ with preference orders:

$$e(\ell)\colon b \succ d \succ H(v_1^\ell) \succ \cdots \succ H(v_n^\ell) \succ B \succ X,$$
$$f(\ell)\colon b \succ d \succ \overleftarrow{H(v_n^\ell)} \succ \cdots \succ \overleftarrow{H(v_1^\ell)} \succ B \succ X.$$

   We also introduce voters $e'(\ell)$ and $f'(\ell)$, whose preference orders are obtained by reversing those of $e(\ell)$ and $f(\ell)$, respectively, and shifting the candidates from X to the back (the exact order of the candidates from X in the last five positions is irrelevant).

2. Let σ be the following preference order:

$$\sigma\colon V(G) \succ x_1 \succ E(G) \succ F(G) \succ x_2 \succ x_3 \succ x_4 \succ x_5 \succ B \succ d \succ b.$$

   We introduce four voters, s_1, s_2, s_3, s_4. Voter s_1 has preference order σ, except that members of B are shifted ahead of x_5, and voter s_2 has the preference order obtained from σ by (a) shifting x_1 to the top, (b) shifting x_2, x_3, and x_4 ahead of the candidates from $E(G) \uplus F(G)$, and (c) shifting d ahead of B. Voters s_3 and s_4 have preference orders that are reverses of σ.

Let E be the just-constructed election. If s_1 and s_2 had preference order σ, then all candidates, except those in X, would have the same score (because for each voter there would be a matching one, with the same preference order but reversed, except that both voters might rank members of X on the bottom); let this score be L. Due to the changes in s_1's and s_2's preference orders:

1. candidate d has score $L + r$,
2. every vertex candidate has score $L - 1$,
3. every edge and fake-edge candidate has score $L - 3$,
4. every blocker candidate has score L, and
5. every candidate in X has score much below $L - r$.

Let I' be an instance of Borda #Destructive Shift-Bribery with election E, designated candidate d, and shift radius r. Further, let M be the set of elections that can be obtained from E by shifting d back by r positions in total, let $f(I)$ be the number of solutions for I (i.e., the number of multicolored independent sets of size h in G), and let $g(I')$ be the number of solutions for I' (i.e., the number of elections in M where d is not a winner). We claim that $f(I) = |M| - g(I')$. In other words, we claim that each election where d wins and which can be obtained from E by shifting him or her back by r positions in total, corresponds to a unique multicolored independent set in G. Since $|M|$ can be computed in polynomial time using a simple dynamic program, showing that our claim holds will complete the proof. First, in Step 1, we show that each solution for I corresponds to a unique election from M where d wins (and which is obtained by shifting d by r positions back), and then, in Step 2, we show that the reverse implication holds.

Step 1. Let $S = \{v^1_{i_1}, \dots, v^h_{i_h}\}$ be some multicolored independent set of G. We obtain a corresponding solution for I' as follows: For each $\ell \in [h]$, we shift d in $e(\ell)$ to be right in front of $v^\ell_{i_\ell+1}$ (or, to be right in front of the first blocker candidate, if $i_\ell = n$), and we shift d in $f(\ell)$ to be right in front of $v^\ell_{i_\ell}$. Doing so requires $n(\Delta+1)+\Delta$ unit shifts for each $\ell \in [h]$, so , altogether, we make $r = h(n(\Delta + 1) + \Delta)$ unit shifts. As a consequence, d has score L and every other candidate has score at most L. Indeed, d passes each vertex candidate exactly once, and each edge and fake-edge candidate at most three times. The former is readily verifiable. The latter can be seen as follows: Fix a color $\ell \in [h]$ and consider voters $e(\ell)$ and $f(\ell)$. In their preference orders, d passes each edge candidate incident to a vertex in $V(\ell) \setminus \{v^\ell_{i_\ell}\}$ exactly once (either in $e(\ell)$ or in $f(\ell)$), and d passes each edge candidate incident to $v^\ell_{i_\ell}$ exactly twice (once in $e(\ell)$ and once in $f(\ell)$). Thus, each edge candidate is passed by d at most three times (if neither of its endpoints is in S, then it is passed twice, and if one of its endpoints is in S, then it is passed three times; both of its endpoints cannot belong to S by definition of an independent set). Similarly, d passes each fake-edge candidate at most three times. Finally, d never passes any of the blocker candidates. Thus, d is a winner in the resulting election.

Step 2. For the other direction, consider an election E' obtained from E by shifting d backward by r positions in total, where d still is a winner. We will show that E' corresponds to a unique, size-h, multicolored independent set in G. First, we recall that d has score L in E' (this is so because he or she had score $L{+}r$ in E and was shifted by r positions backward). This means that to remain a winner, d could not have passed any of the blocker candidates, because then some blocker candidate would have more than L points. As a consequence, the only votes in which d could have been shifted are $e(\ell)$ and $f(\ell)$, for each $\ell \in [h]$. Further, in each of these votes d could have been shifted by at most $n(\Delta + 1)$ positions. In fact, it must have been the case that for each $\ell \in [h]$, the total number of positions by which d was shifted in $e(\ell)$ and $f(\ell)$ was exactly $r/h = n(\Delta+1)+\Delta$. If this were not the case, then for some ℓ, candidate d would have been shifted by more than $n(\Delta+1)+\Delta$ positions in total in $e(\ell)$ and $f(\ell)$ and, as a consequence, d would have passed at least one vertex candidate from $V(\ell)$ twice. Such a vertex candidate would end up with score at least $L+1$ and d would not have been a winner. To convince oneself that this is the case, let y be some positive integer and consider vote $e(\ell)$ with d shifted by $n(\Delta+1)$ positions to the back (right in front of the first blocker candidate), and vote $f(\ell)$ with d shifted by $\Delta + y$ positions to the back. Initially d passes v_n^ℓ in both $e(\ell)$ and $f(\ell)$. Now consider the process of repeatedly undoing a single unit shift in $e(\ell)$ and performing a single additional unit shift in $f(\ell)$. At each point of this process, there is at least one vertex candidate in $V(\ell)$ such that d is ranked behind this vertex in both votes.

Analogous reasoning shows that for each $\ell \in [h]$, there is a number $i_\ell \in [n]$ such that in $e(\ell)$ candidate d is shifted back by exactly $i_\ell(\Delta+1)$ positions, and in $f(\ell)$ candidate d is shifted back by $(n-i_\ell)(\Delta+1)+\Delta$ positions (indeed, it suffices to repeat the reasoning from the end of the above paragraph for $y=0$ to see that these are the only numbers of shifts for which d never passes any of the candidates from $V(\ell)$ twice). Now we note that the set $S := \{v_{i_1}^1, \dots, v_{i_h}^h\}$ is a size-h, multicolored, independent set: The first two observations are immediate; for the latter, we note that—analogously to the reasoning in Step 1—if S were not an independent set, then d would pass some edge candidate four times, giving him or her score $L+1$, which would prevent d from being a winner. $\square$

A.6 Proof of Theorem 6

Theorem 6 (★). *Parameterized by the shift radius,* Borda #Constructive Shift-Bribery *is in* FPT *(for unary-encoded prices), but the destructive variant is* #W[1]*-hard, even for unit prices.*

Proof (constructive case). We show how Borda #Constructive Shift-Bribery parameterized by the radius r can be solved in FPT time using dynamic programming. We start with the assumption of unit costs and later explain how to extend the dynamic program to work with arbitrary unary-encoded costs.

First, observe that, given some budget r and unit costs, we know the final score $s^*$of candidate p after shifting it forward by r position in total (p gains one point with each position it is shifted forward). Our problem becomes very easy when every other candidate already has score at most s^*(before shifting p), because clearly no candidate other than p may gain a point.

In general, since other candidates loose r points in total, there may be up to r *critical candidates* that have score greater than s^*(before shifting p). Moreover, for each critical candidate c we can compute a *demand value* $d(c)$ which denotes the number of times p must get shifted ahead of c (equivalently, $d(c)$ is the original score of c minus s^*).

Let the candidate set be $C = \{p, c_1, \dots, c_m\}$ and, for the ease of presentation, assume that the candidates $c_1, \dots, c_m$ are sorted by their demand values (with non-critical candidates having demand zero). We define the *initial demand vector* $\vec{d_0}$ to be an r-dimensional vector of natural numbers where the i-th component $\vec{d}[i]$ specifies how many times candidate p needs to pass candidate c_i to ensure that c_i has at most score s^* (if r is larger than the number of non-p candidates, we pad the demand vector with zeros; for simplicity, in the further discussion we assume that r is at most as larger as the number of non-p candidates). Thus we have $\vec{d_0} = (d(c_1), \dots, d(c_r))$.

For each voter v and non-negative integer $r' \leq r$ we define the *gain vector* as

$$\vec{g}(v, r') = (g(c_1, v, r') \dots, g(c_r, v, r')),$$

where $g(c_i, v, r') = 1$ if p passes candidate c_i when p is shifted forward at cost r' in vote v. If spending cost r' in vote v is impossible (e.g., because p would already be pushed to the top position at a lower cost), then we set the gain vector to $(-2r, \dots, -2r)$. This way, we later ensure that such "invalid actions" are never counted. Naturally, given some voter v and some non-negative integer $r' \le r$, the vector $\vec{d'} = \vec{d_0} - \vec{g}(v, r')$ describes the demand vector assuming that p was shifted forward by r' position in vote v. Note that there are at most $(r+1)^{\min(r,m)}$ possible demand vectors.

We are now ready to solve our problem via dynamic programming, using table T of FPT(r) size. More precisely, let $T[i, r', \vec{d}]$ denote the number of ways to shift p by r' positions in total, within the first i voters, and ending up with demand vector $\vec{d}$. The overall solution for our problem will be in the entry $T[n, r, \vec{0}]$. It remains to show how to compute the entries of T.

We do so with increasing i, going over all combinations of r' and $\vec{d}$. Clearly, $T[1, \dots]$ has to be initialized mostly with zero entries since at most r different demand vectors can be realized. More precisely, we have:

$$T[1, r', \vec{d}] = \begin{cases} 1 & \text{if } \vec{d} = \vec{d_0} - \vec{g}(v_1, r') \\ 0 & \text{otherwise.} \end{cases}$$

We fill-in the table for each $i > 1$ using formula:

$$T[i, r', \vec{d}] = \sum_{\substack{r'' \le r', \\ \vec{d'} \le^1 \vec{d}}} T[i-1, r' - r'', \vec{d'}] \cdot [\vec{d} = (\vec{d'} - g(v_i, r''))],$$

where $\vec{d'} \le^1 \vec{d}$ holds if vector $\vec{d'}$ is component-wise equal or smaller by at most one compared to $\vec{d}$ (formally, $\vec{d'} \le^1 \vec{d} \Leftrightarrow \forall j \in [r] : \vec{d}[j] - 1 \le \vec{d'}[j] \le \vec{d}[j]$), and where $[[X]]$ is one if equation X holds and zero otherwise. Note that this recurrence goes over all possible ways to distribute the budget r' among the first $i-1$ voters and voter v_i, while only considering demand vectors that can be reached with the respective budget for voter v_i.

The table size is upper-bounded by $n \cdot r \cdot (r+1)^{\min(r,m)}$. Initializing a table entry works in $O(m)$ time (compute the gain vector and compare the demand vectors). While updating the table, an entry can be computed in time $r \cdot 2^{\min(r,m)} \cdot O(m)$ because there are at most r possibilities for r'' and at most $2^{\min(r,m)}$ possibilities for $\vec{d'}$. Altogether, this means we can compute T and solve our problem in FPT time with respect to the shift radius r.

Finally, we explain how to extend the FPT-algorithm to also work with arbitrary unary encoded costs. The crucial difference for non-unit costs is that we cannot compute the final score of p from our budget r. Instead, we guess (that is, go through all possibilities) the final score and then apply the algorithm described above with small modifications. To ensure that p indeed ends up with the desired final score, we have to keep track of the score p obtains. This can easily be done by extending the demand (resp. gain) vector by one more component that stores the number of times p has to pass (resp. passes) some candidate. $\square$

Proof (destructive case). We give a polynomial-time Turing reduction from #MATCHING to BORDA #DESTRUCTIVE SHIFT-BRIBERY. Let (G, k) be an instance of #MATCHING, where G is a bipartite graph with vertex set $U(G) \uplus V(G)$, and k is a positive integer. Without loss of generality, we assume that $U(G) = \{u_1, \dots, u_n\}$, $V(G) = \{v_1, \dots, v_n\}$, and $k \le n$.

Our reduction proceeds as follows. First, we form the set of relevant candidates $R := \{d, p\} \uplus U(G) \uplus V(G)$, where d is the designated candidate. Moreover, for each relevant candidate $r \in R$, we form a set $D(r)$ of $3k+1$ dummy ones. We will form an election E, where these candidates will have the following Borda scores (X is some positive integer, whose value depends on the specifics of the construction; we will be counting ways in which d can cease to be a winner by shifting him or her backward by $3k$ positions):

$$\text{score}_E(d) = X + 3k, \tag{1}$$
$$\text{score}_E(p) = X - k + 1, \tag{2}$$
$$\text{score}_E(u_1) = \cdots = \text{score}_E(u_n) = X - 1, \tag{3}$$
$$\text{score}_E(v_1) = \cdots = \text{score}_E(v_n) = X - 1, \text{ and} \tag{4}$$
$$\text{each dummy candidate has score at most } X - 3k - 1. \tag{5}$$

To achieve these scores, we build the voter collection V as follows:

1. For each edge $e = \{u_i, v_j\}$ of the input graph, there is an *edge voter* v_e with preference order $v_e\colon d \succ u_i \succ v_j \succ p \succ D(p) \succ \cdots$.
2. We also have a group of *score voters*, who ensure that conditions (1)–(5) hold. Let π be the following preference order, giving the basic pattern for forming all score voters:

$$\pi\colon D(u_1) \succ u_1 \succ \cdots \succ D(u_n) \succ u_n \succ D(p) \succ p \succ$$
$$D(v_1) \succ v_1 \succ \cdots \succ D(v_n) \succ v_n \succ D(d) \succ d.$$

   The crucial feature of π is that relevant candidates are separated from each other with at least $3k+1$ dummy ones; their particular order is not important. For each relevant candidate $c \in R$, let $\text{inc}(c)$ be a pair of preference orders where one is equal to reversed π, and the other is identical to π except that c is shifted one position forward. By adding a pair of voters with preference orders $\text{inc}(c)$ to the election, we increase the score of c by $|C|$, the scores of the other relevant candidates by $|C|-1$, and the scores of the dummy candidates by at most $|C|-1$. For each relevant candidate $c \in R$, we add polynomially many such pairs of voters—the polynomial is with respect to n and k—so that the scores of the candidates and the value of X implied by this process satisfy conditions (1)–(5). Here is the direct algorithm: First, for each relevant candidate $c \in R$ we add the smallest number of pairs of voters $\text{inc}(c)$ so that, taken together with the edge voters, all relevant candidates have identical scores and each relevant candidate has a higher score than each dummy candidate. Then, for each relevant candidate $c \in R$, we add $3k+1$ pairs of voters $\text{inc}(c)$ (this ensures that, in total, each relevant candidate has at least $3k$ points more than each dummy candidate). Finally, we add $4k$ pairs of voters $\text{inc}(d)$, one pair of voters $\text{inc}(p)$, and for each $i \in [n]$, we add $k-1$ pairs of voters $\text{inc}(u_i)$ and $k-1$ pairs of voters $\text{inc}(v_i)$ (this ensures that the scores of all relevant candidates are as promised).

Next, we form an election F identical to E, except that one of the edge voters ranks p one position lower (so that p's score in F is $X - k$). Let I_E and I_F be instances of Borda #Destructive Shift-Bribery with designated candidate d, shift radius $3k$, unit prices, and elections E and F, respectively. Our reduction queries the oracle for the numbers of solutions for I_E and I_F, subtracts the latter from the former, and outputs this value. We claim that it is exactly the number of size-k matchings in G.

To see why this is the case, consider some solution for I_E. There are two possibilities: Either d passes some member of $U(G) \uplus V(G)$ twice (in which case this candidate gets at least $X+1$ points, whereas d always gets exactly X points), or d passes each member of $U(G) \uplus V(G)$ at most once. In the latter case, only p can defeat d (all other candidates have at most X points). However, for this to happen, d must pass p exactly k times (with the shift radius of $3k$, d cannot pass p more times). Further, since we assumed that d never passes a member of $U(G) \uplus V(G)$ more than once, the votes where d passes p must correspond to a size-k matching in G. We refer to such solutions as *matching solutions*.

The set of solutions for I_F contains all solutions for I_E except for the matching ones (because in I_F, p ends up with at most X points and not $X + 1$). So, by subtracting the number of solutions for I_F from the number of solutions for I_E, we get exactly the number of size-k matchings in G. □

B Additional Material for Section 4

In this section, we provide further details concerning our experiments on how #Swap-Bribery can be used to measure the robustness of election winners. We start with describing the dataset we used in Appendix B.1. Subsequently, in Appendix B.2, we explain how we used sampling to measure the winning probability of candidates in case a certain number of random swaps is performed.

B.1 Dataset

Here we briefly recall the four statistical cultures that we mention in our experimental studies:

Impartial Culture. In the impartial culture model (IC), each election consists of preference orders chosen uniformly at random.

The Urn Model. In the urn model, with parameter $\alpha \in \mathbb{R}_+$, to generate an election (with m candidates), we start with an urn containing all $m!$ preference orders and generate the votes one by one, each time drawing the vote from the urn and then returning it there with $\alpha m!$ copies.

The Mallows Model. In the Mallows model, with parameter $\phi \in [0, 1]$, each election has a central preference order v^* (chosen uniformly at random) and the votes are sampled from a distribution where the probability of obtaining vote v is proportional to $\phi^{d_{\mathrm{swap}}(v^*,v)}$.

Euclidean Models. In the tD-Cube and tD-Sphere models, the candidates and voters are points sampled uniformly at random from a t-dimensional hypercube/sphere, and the voters rank the candidates with respect to their distance (so a voter ranks the candidate whose point is closest to that of the voter first, then the next closest candidate, and so on).

Single-Peaked Elections. Let $C = \{c_1, \dots, c_m\}$ be a set of candidates and let $\lhd$ be a linear order over C. We will refer to $\lhd$ as the societal axis. We say that a voter v's preference order is single-peaked with respect to the axis if for every $t \in [m]$ it holds that v's t top-ranked candidates form an interval within $\lhd$. An election $E = (C, V)$ is single-peaked with respect to a given axis if each voter's preference order is single-peaked with respect to this axis; an election is single-peaked if it is single-peaked with respect to *some* axis.

The notion of single-peakedness is due to Black [5] and is intuitively understood as follows: The societal axis orders the candidates with respect to positions on some one-dimensional issue (e.g., it may be the level of taxation that the candidate support, or a position on the political left-to-right spectrum). Each voter cares only about the issue represented on the axis. So, each voter chooses his or her top-ranked candidate freely, but then the voter chooses the second-best one among the two candidates next to the favorite one on the axis, and so on.

Szufa et al. [30] use the following two models for generating single-peaked elections (in both models, the societal axis is chosen uniformly at random and the votes are generated one-by-one, until a required number is produced):

Single-Peaked (Conitzer). In the Conitzer model, we generate a vote as follows. First, we choose the top-ranked candidate uniformly at random. Then, we perform $m-1$ iterations, extending the vote with one candidate in each iteration: With probability $1/2$ we extend the vote with the candidate "to the left" of the so-far ranked ones, and with probability $1/2$ we extend it with the one "to the right" of the so-far ranked ones (if we ran out of the candidates on either side, then, naturally, we always choose the candidate from the other one). This model was popularized by Conitzer [12] and, hence, its name.

Single-Peaked (Walsh). In the Walsh model, we generate votes by choosing them uniformly at random from the set of all preference orders single-peaked with respect to a given axis. This model was popularized by Walsh [32], who also provided a sampling algorithm.

Szufa et al. [30] give a detailed analysis explaining why these two models produce quite different elections (they also point out that 1D-Interval elections tend to be very similar to single-peaked elections from the Conitzer model).

Elecitons Single-Peaked on a Circle (SPOC). Peters and Lackner [28] extended the notion of single-peaked elections to single-peakedness on a circle. The model is very similar to the classic notion of single-peakedness, except that the axis is cyclic. Let C be a set of m candidates. Voter v has a preference order that is single-peaked on a circle with respect to the axis $\lhd$ if for every $t \in [m]$ it holds that the set of t top-ranked candidates according to v either forms an interval with respect to $\lhd$ or a complement of an interval.

Szufa et al. [30] generate SPOC elections in the same way as single-peaked elections in Conitzer's model, except that the axis is cyclic (so one never "runs out of candidates" on one side). Such SPOC elections are quite similar to 2D-Hypersphere ones (and, as indicated by Szufa et al., also to IC elections).

Single-Crossing Elections. Intuitively, an election is single-crossing if it is possible to order the voters so that for each two candidates a and b, as we consider the voters in this order, the relative ranking of a and b changes at most once. Formally, single-crossing elections are defined as follows.

Let $E = (C, V)$ be an election, where $C = \{c_1, \dots, c_m\}$ and $V = (v_1, \dots, v_n)$. This election is single-crossing with respect to its natural voter order if for each two candidates c_i, c_j there is an integer $t_{i,j}$ such that the set $\{\ell \mid v_\ell \text{ ranks } c_i \text{ above } c_j\}$ is either $\{1, 2, \dots, t_{i,j}\}$ or $\{t_{i,j}, \dots, n-1, n\}$. An election is single-crossing if it is possible to reorder its voters so that it becomes single-crossing with respect to the natural voter order.

It is not clear how to generate single-crossing elections uniformly at random, or what a good procedure for generating single-crossing elections should be. Szufa et al. [30] propose one procedure and we point the reader to their paper for the details.

B.2 Sampling Elections

In our experiments, to calculate $P_{E',c}(r)$ for different candidates c, elections E', and swap distances r, we sampled elections at swap distance r from E' uniformly at random. Unfortunately, to sample elections at some given swap distance uniformly at random, it is not enough to simply perform r swaps in some of the votes in E', as this procedure does not necessarily produce an election at distance r and cannot be easily adapted to result in a uniform distribution. Thus, we use a sampling procedure that relies on counting the number of elections at some swap distance r.

To compute this value, we employ dynamic programming using a table T^m_E: Each entry $T^m_E[n, r]$ contains the number of elections at swap distance r from a given fixed election with

n voters and m candidates (note that it is irrelevant what this election is, so we can simply assume an election with n identical votes). In the following, let $u(m) = \frac{m(m-1)}{2}$ denote the maximal number of swaps that can be performed in a vote.

To compute T_E^m, we start by computing $T_E^m[1, r]$, that is, the number of votes at swap distance r from a given single vote over m candidates. We compute this value using dynamic programming. As we will use this quantity separately in the following sampling algorithm, we create a separate table T_V for it, where $T_V[m, r]$ contains the number of votes at swap distance r from a given vote over m candidates. Note that $T_V[m, r]$ is simply the number of permutations over m elements with r inversions. Computing this value is a well-studied problem and we use the following procedure [27]: We initialize the table with $T_V[m, 0] = 1$. We update the table by increasing m and for each m starting from $r = 0$ and going to $r = u(m)$ using the following recursive relation:

$$T_V[m, r] = T_V[m, r-1] + T_V[m-1, r] - T_V[m-1, r-m].$$

Unfortunately, no closed form expression for this value seems to be known [27].

Using T_V, we are now ready to compute T_E^m. We start by setting $T_E^m[1, r] = T_V[m, r]$. Subsequently, we fill T_E^m by increasing n and for each n starting from $r = 0$ and going to $r = u(m) \cdot n$ using the following recursive relation:

$$T_E^m[n, r] = \sum_{i=\max(r-u(m)\cdot(n-1),0)}^{\min(r,u(m))} T_V[m, i] \cdot T_E^m[n-1, r-i].$$

The reasoning behind this formula is that, in the nth vote, between $\max(r - u(m) \cdot (n-1), 0)$ and $\min(r, u(m))$ swaps can be performed. We iterate over all these possibilities and count, for each i in this interval, the number of elections where i swaps in the nth vote and $r - i$ swaps in the remaining $n - 1$ votes are performed.

Using T_E and T_V, we split the process of sampling elections at swap distance r into two steps. First, we sample the distribution of swaps to votes proceeding recursively vote by vote. For the first vote, the probability that i swaps are performed is proportional to the number of possibilities to perform i swaps in the first vote times the number of elections at swap distance $r - i$ from the remaining $n - 1$-voter election. This results in performing $i \in [\max(r - u(m) \cdot (n-1), 0), \min(r, u(m))]$ swaps in the first vote with probability

$$\frac{T_V[m, i] \cdot T_E^m[n-1, r-i]}{T_E^m[n, r]}.$$

We then delete this vote and solve the problem for the remaining $n-1$ votes and $r-i$ swaps recursively.

Second, for each vote, we generate a vote at the assigned swap distance r. To do so, we generate a permutation with r inversions uniformly at random and sort the vote according to the permutation. A permutation σ over m elements is fully characterized by the tuple $t(\sigma) = (t_1(\sigma), \ldots, t_m(\sigma))$ where, for each $i \in [m]$, $t_i(\sigma)$ denotes the number of entries smaller than i appearing after i in σ. Note that, for all $i \in [m]$, it needs to hold that $0 \leq t_i(\sigma) \leq m - i$ and that the number of inversions in a permutation σ corresponds to the sum of the entries of $t(\sigma)$. Thus, the problem of uniformly sampling a permutation over m elements with r inversions is equivalent to uniformly distributing r indistinguishable balls into $m-1$ distinguishable bins with each bin $i \in [m-1]$ having capacity i. We again proceed bin by bin: We put $i \in [0, m-1]$ balls into the first bin with probability $\frac{T_V[m-1,r-i]}{T_V[m,r]}$ and then solve the problem for the remaining $m - 2$ bins and $r - i$ balls recursively.

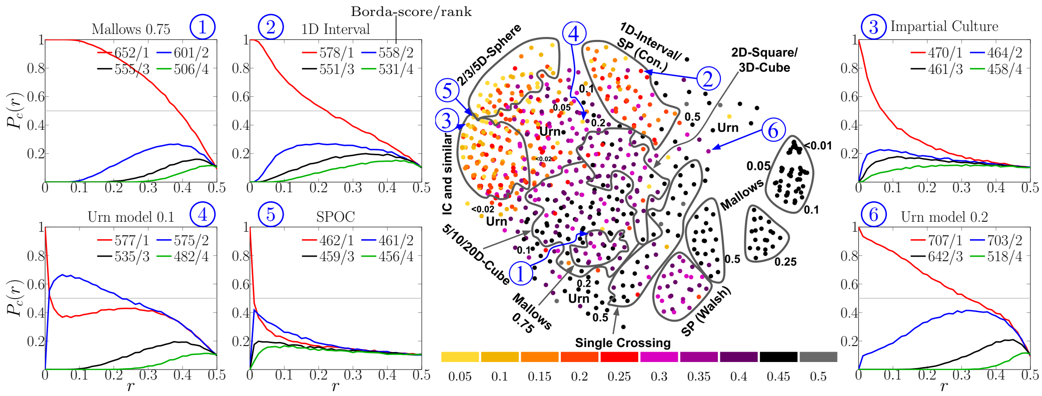


Figure 5: Map of elections visualizing the 50%-winner threshold for Borda and six plots showing $P_{E,c}(r)$ as a function of r, for six selected elections and the four most successful candidates in each.

B.3 Results for Borda

Let us now discuss the results regarding Borda elections in more detail. In Figure 5, we display the results of our experiments for Borda on the 800 synthetic elections with 10 candidates and 100 voters. As in Figure 3, the "map of elections" shows the 50%-winner threshold and the six plots show the probability of victory of four candidates in six selected elections (the interpretation of these plots is the same as in the case of Plurality and Figure 3).

Comparing the map for Plurality (from Figure 3) and Borda (from Figure 5), we see that Finding 2 still holds: Borda winners tend to be more robust than winners under Plurality, as witnessed by the fact that for most elections the 50%-winner threshold for Borda is higher than for Plurality. This is particularly true for tD-Cube elections and Urn elections with high values of the contagion parameter. Most of the patterns described for Plurality can also be found in the map for Borda; some of them are even more pronounced, as, e.g., the difference between 1D-Interval and multidimensional hypercube elections.

Considering the robustness of election winners, as indicated in Figure 2b, the score difference between the winner and its runner-up has a higher predictive value for Borda than for Plurality.

Finding 6. *Candidates who win by* 50 *points or more are very robust winners. The winning probability of candidates who win by around* 20 *points might decrease noticeably already at small swap distance. Nevertheless, these candidates typically stay the most probable winner for a long time.*

Let us consider the six elections from Figure 5. Among all elections from our dataset, Election 1 has the largest score difference between the winner and the runner-up where at swap distance 0.1 the blue candidate has some non-negligible probability of victory. Nevertheless, the red candidate is the clear winner of this elections, even after many random swaps. Note that this finding may be intuitively surprising, as 50 points seems relatively little as compared to the totally awarded 4500 points (yet, in a Borda election with 10 candidates and 100 voters, the highest possible score is 900 and compared to this value, 50 is not completely negligible).

In contrast to this, in Election 2, where the score difference between the winner and the runner-up is around 20, the blue candidate starts to have a non-negligible chance of winning even at swap distance 0.025. Nevertheless, the red candidate stays the most probable winner until swap distance 0.5 and wins with more than 50% probability until swap distance 0.2. Thus, there is little doubt that the red candidate should be the winner of the election.

Finding 7. *Candidates winning by less than* 10 *points might be both quite robust or very sensitive to random swaps.*

As soon as the score difference between the first and second candidate drops to around 10 points, elections with a similar distribution of scores start to show fundamentally different behavior. Only examining the scores, Elections 3 to 6 all seem close, as the maximum gap between the red and blue candidates is at most six points in these elections. However, looking at the plots, there are significant differences. For Elections 4 and 5, the initial winner stops to be the most probable winner already at swap distance 0.0125, which is the smallest swap distance we examined. In Election 5, all candidates lie close to each other. In contrast to this, in Election 4, the blue candidate dominates the red candidate at swap distance 0.0125 and above to an extent that one might wonder whether the blue candidate is not the "true" winner of the election (even though the election was generated from the urn model and was not modified in any way).

Elections 4 and 5 stand in sharp contrast to the other two "close" elections, i.e., Elections 3 and 6. For example, Elections 3 and 5, which both have a similar distribution of scores and lie very close on the map, exhibit quite a different behavior. In Election 3, the red candidate remains the most probable winner until swap distance 0.4 and in Election 5 the red candidate stops being the most probably winner very early on. In Election 6, the red candidate stays a winner with probability greater than 50% even until swap distance 0.3. To sum up, despite all seeming quite close, while for Elections 4 and 5 it is recommendable to reexamine the election issue, for Election 3 and, in particular, for Election 6, the selected red candidate is quite a robust winner.

B.4 Results for Copeland

To illustrate that our approach to assess the robustness of an election winner is also useful for non-scoring rules, in the following, we briefly look at a variant of the Condorcet-consistent Copeland rule. We start by examining the synthetic dataset and in the end briefly look at the cycling data. Under Copeland, the score of a candidate c is the number of other candidates $d \neq c$ for which a strict majority of voters prefer c over d minus the number of other candidates $d \neq c$ which a strict majority of voters prefers over c. Thus, the Copeland score of a candidate lies between $m-1$ and $1-m$. For elections with a higher number of voters than candidates, the range of possible scores might therefore often be not rich enough to differentiate the strength of candidates adequately.

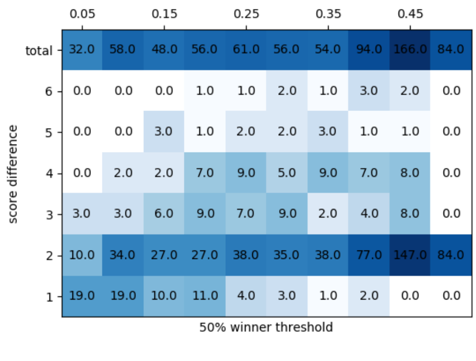


Figure 6: Number of elections with a certain 50%-winner threshold (x-axis) and a certain score difference (y-axis) under Copeland.

This intuition is confirmed in Figure 6, where we display the relationship of the score difference between the winner and runner-up and the 50%-winner threshold for the elections from our synthetic dataset. In particular, a score difference of two has no real predictive value for the 50%-winner threshold, which is quite intuitive: In an election where all voters rank the candidates in the same order, the winner has a score of $m-1$ and the second candidate has a score of $m-3$. In contrast, in an

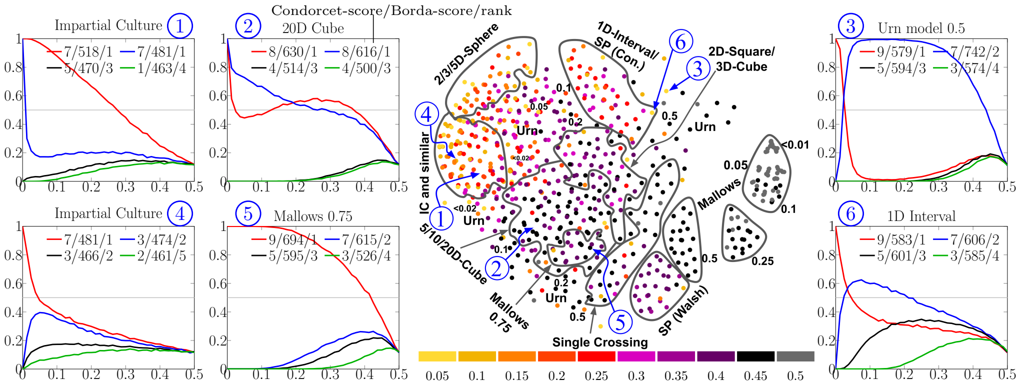


Figure 7: Map of elections visualizing the 50%-winner threshold for Copeland and six plots showing $P_{E,c}(r)$ as a function of r, for six selected elections and the four most successful candidates in each.

election consisting of $2x+1$ voters where $x+1$ of them rank a candidate c in first place and a candidate d in second place and the others rank d in first and c in last place, c has a score of $m-1$ and d of $m-3$.

We depict the map of elections visualizing the 50%-winner threshold for Copeland and six selected elections in Figure 7. The map for Copeland is remarkably similar to the one for Borda displayed in Figure 5. The only major differences are that quite some elections that are close for Borda are tied for Copeland and that for some other elections the robustness of the winner is a little higher for Copeland than for Borda.

Finding 8. *A small score difference has no predictive value for the robustness of the election winner under Copeland. Candidates winning by a larger score difference are typically relatively robust winners.*

We have already seen in Figure 6 that a score difference of two may result in elections with very robust and very sensitive winners. This is also visible in Elections 3, 5, and 6 in Figure 7, which all have a score difference of two. While both Election 6 and, in particular, Election 3 have a non-robust winner, the red candidate in Election 5 is a very robust winner. Notably, as in most elections with a score difference of two from our dataset, in all three elections, the red candidate is a Condorcet winner. Thus, as an alternative measure for the robustness of the winning candidate in these three elections one might consider the difference between the number of voters preferring the winning candidate to the runner-up and the number of voters preferring the runner-up to the winning candidate. This value is 10 in Election 3, 20 in Election 5, and 4 in Election 6. While Election 5 with the highest value is also the one with the most robust winner, the extend of the robustness can typically not be directly concluded from these values.

With increasing score difference, winners become more and more robust (notably, only very few elections have a score difference higher than four tough). However, there also exist several elections with a "higher" score difference of four with a non-robust winner. An example of this is Election 4, where even the green candidate, which is five points behind the winner already wins in around 1.7% of elections at swap distance 0.0125. To confirm that a high score difference leads to robust winners, experiments with a higher number of candidates and therefore also a higher range of scores are needed.

Finding 9. *In tied elections, the winning probability of the original winners may be quite different (even at a small swap distance).*

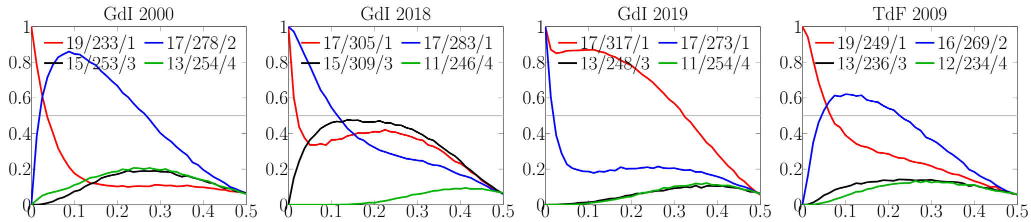


Figure 8: We display for four elections selected from our cycling dataset under Copeland's voting rule, $P_{E,c}(r)$ as a function of r for the four most successful candidates. In the legend, for each candidate, the first entry contains its Copeland score, the second entry its Borda score and the last entry its rank in the original election.

Out of the 800 elections from the synthetic dataset, 94 have a tied winner under Copeland, which are much more than for Plurality and Borda. While for around half of these elections, the winning probabilities of the initially winning candidates are quite similar at (small) swap distances, there also exist a significant number of elections where this is not the case. An example is Election 1, where the red candidate wins significantly more elections than the blue candidate already if only 1.25% of possible swaps are performed. This might justify to select the red candidate if a single candidate needs to be selected as the winner. One might argue that using the Borda score of the candidates as a tie-breaker would have a similar effect here. However, as exhibited in the tied Election 2, this might not always be the case. In Election 2 the blue candidate has a lower Borda score than the red candidate. However, at swap distance 0.15 or below, which can be considered as the most important region, the blue candidate has a significantly higher chance to win than the red candidate.

Lastly, as in the main body for Borda and Plurality, we display in Figure 8 four elections selected from our cycling dataset. Both GdI 2018 and 2019 have a tied Copeland winner with 17 points, where both times the red candidate has a higher Borda score. Considering the development of the winning probabilities of the initially winning candidates at increasing swap distance, in GdI 2019, the red candidate is clearly stronger than the blue candidate. In contrast to this, in GdI 2018, both initial winners are not very robust with a slight advantage for the blue candidate. However, already at swap distance 0.1375 the black candidate surpasses the blue candidate as the most probable winner.

For many of the non-tied elections from the dataset, the initial winner is quite robust. However, this is not the case for GdI 2000 and TdF 2009. In GdI 2000, the red candidate is the Condorcet winner. Yet, already at swap distance 0.025, the blue candidate becomes the most probable winner and subsequently for increasing swap distance dominates the red candidate to a significant extend. In TdF 2009, the score difference is three. Nevertheless, a similar, slightly less strong behavior is visible.