Multivariate Complexity Analysis of Swap Bribery

Britta Dorn[1] and Ildikó Schlotter[2]*

[1] Fakultät für Mathematik und Wirtschaftswissenschaften, Universität Ulm
Helmholtzstr. 18, D-89081 Ulm, Germany
britta.dorn@uni-ulm.de
[2] Budapest University of Technology and Economics
H-1521 Budapest, Hungary
ildi@cs.bme.hu

Abstract. We consider the computational complexity of a problem modeling *bribery* in the context of voting systems. In the scenario of Swap Bribery, each voter assigns a certain price for swapping the positions of two consecutive candidates in his preference ranking. The question is whether it is possible, without exceeding a given budget, to bribe the voters in a way that the preferred candidate wins in the election. We initiate a parameterized and multivariate complexity analysis of Swap Bribery, focusing on the case of k-approval. We investigate how different cost functions affect the computational complexity of the problem. We identify a special case of k-approval for which the problem can be solved in polynomial time, whereas we prove NP-hardness for a slightly more general scenario. We obtain fixed-parameter tractability as well as W[1]-hardness results for certain natural parameters.

1 Introduction

In the context of voting systems, the question of how to manipulate the votes in some way in order to make a preferred candidate win the election is a very interesting question. One possibility is *bribery*, which can be described as spending money on changing the voters' preferences over the candidates in such a way that a preferred candidate wins, while respecting a given budget. There are various situations that fit into this scenario: The act of bribing the voters in order to make them change their preferences, or paying money in order to get into the position of being able to change the submitted votes, but also the setting of systematically spending money in an election campaign in order to convince the voters to change their opinion on the ranking of candidates.

The study of bribery in the context of voting systems was initiated by Faliszewski, Hemaspaandra, and Hemaspaandra in 2006 [15]. Since then, various models have been analyzed. In the original version, each voter may have a different but fixed price which is independent of the changes made to the bribed vote. The scenario of nonuniform bribery introduced by Faliszewski [14] and the case of microbribery studied by Faliszewski, Hemaspaandra, Hemaspaandra, and Rothe in [17] allow for prices that depend on the amount of change the voter is asked for by the briber.

In addition, the Swap Bribery problem as introduced by Elkind, Faliszewski, and Slinko [13] takes into consideration the ranking aspect of the votes: In this model, each voter may assign different prices for swapping two consecutive candidates in his preference ordering. This approach is natural, since it captures the notion of small changes and comprises the preferences of the voters. Elkind et al. [13] prove complexity results for this problem for several election systems such as Borda, Copeland, Maximin, and approval voting. In particular, they provide a detailed case study for k-approval. In this voting system,

* Supported by the Hungarian National Research Fund (OTKA 67651).

	Result	Reference
$k = 1$ or $k = m - 1$	P	[13]
$k = 2$	NP-complete	[3, 13]
$k \geq 3$ constant, costs in $\{0, 1, 2\}$	NP-complete	[13]
$k \geq 2$ constant, costs in $\{0, 1\}$ and $\beta = 0$	**NP-complete**	[3], Prop. 2
$k \geq 2$ constant, costs in $\{1, 1 + \varepsilon\}$, $\varepsilon > 0$	**NP-complete**, **W[1]-hard (β)**	Thm. 3
k constant, m or n constant	P	[13]
k part of the input, $n = 1$	NP-complete	[13]
$2 \leq k \leq m - 2$ part of the input, costs in $\{0, 1\}$ and $\beta = 0$, n constant	**NP-complete**	[4], Prop. 2
k part of the input, $n = 1$	**W[1]-hard (k)**	Thm. 7
k part of the input, all costs $= 1$	**P**	Thm. 1
k part of the input	**FPT (m)**	Thm. 5
k part of the input	**FPT (n)**	Thm. 6
k part of the input	**FPT (β, n) by kernelization**	Thm. 8

Table 1. Overview of known and new results for Swap Bribery for k-approval. The results obtained in this paper are printed in bold. Here, m and n denote the number of candidates and votes, respectively, and β is the budget. For the parameterized complexity results, the parameters are indicated in parentheses.

every voter can specify a group of k preferred candidates which are assigned one point each, whereas the remaining candidates obtain no points. The candidates which obtain the highest sum of points over all votes are the winners of the election. Two prominent special cases of k-approval are plurality, (where $k = 1$, i.e., every voter can vote for exactly one candidate) and veto (where $k = m - 1$ for m candidates, i.e., every voter assigns one point to all but one disliked candidate). Table 1 shows a summary of research considering Swap Bribery for k-approval, including both previously known and newly achieved results.

This paper contributes to the further investigation of the case study of k-approval that was initiated in [13], this time from a parameterized point of view. The main goal of this approach is to find *fixed-parameter tractable* (FPT) algorithms confining the combinatorial explosion which is inherent in NP-hard problems to certain problem-specific parameters, or to prove that their existence is implausible. This line of research has been pioneered by Downey and Fellows [11], see also [19, 26] for two more recent monographs, and naturally expands into the field of multivariate algorithmics, where the influence of "combined" parameters is studied, see the recent survey by Niedermeier [27]. These approaches seem to be appealing in the context of voting systems, where NP-hardness is a desired property for various problems, like Manipulation (where certain voters, the manipulators, know the preferences of the remaining voters and try to adjust their own preferences in such a way that a preferred candidate wins), Lobbying (here, a lobby affects certain voters on their decision for several issues in an election), Control (where the chair of the election tries to make a certain candidate win (or lose) by deleting or adding either candidates or votes), or, as in our case, Swap Bribery. However, NP-hardness does not necessarily constitute a guarantee against such dishonest behavior. As Conitzer et al. [9] point out for the Manipulation problem, an NP-hardness result in these settings would lose relevance if an efficient fixed-parameter algorithm with respect to an appropriate parameter was found. Parameterized complexity can hence provide a more robust notion of hardness. The investigation of problems from voting theory under this aspect has started, see for example [2, 4, 5, 8, 25].

We contribute to the research concentrating on Swap Bribery for k-approval by first examining how the computational complexity of this problem depends on certain restrictions on the cost function. We show that a very restricted version of this problem where there are only two types of costs and k is a fixed value is already NP-hard, as well as fixed-parameter intractable if the parameter is the budget. By contrast, we show that the natural special case where all costs are identical can be solved in polynomial time. Shifting our attention the general cost functions, we obtain fixed-parameter tractability with respect to the parameter 'number of candidates' for k-approval and a large class of other voting systems. We also investigate the parameter 'number of votes', and consider k-approval where k is part of the input. In this case, the problem is known to be NP-complete already for only one vote ($n = 1$); we strengthen this result by proving W[1]-hardness with respect to the parameter k. Complementing this contribution, we present a fixed-parameter tractable algorithm with respect to n for the case where k is a constant, using the technique of color-coding. We also provide a polynomial kernel for certain combined parameters.

The paper is organized as follows. After introducing notation in Section 2, we investigate the complexity of Swap Bribery depending on the cost function in Section 3, where we show the connection to the Possible Winner problem, identify a polynomial-time solvable case of k-approval and prove a hardness result. In Section 4, we consider the parameter 'number of candidates' and obtain an FPT result for Swap Bribery for a large class of voting systems. Section 5 investigates the influence of the parameter 'number of votes', providing both W[1]-hardness and fixed-parameter tractability results and considering combinations of different parameters. We conclude with a discussion of open problems and further directions that might be interesting for future investigations.

2 Preliminaries

Elections. An $\mathcal{E}$-*election* is a pair $E = (C, V)$, where $C = \{c_1, \ldots, c_m\}$ denotes the set of *candidates*, $V = \{v_1, \ldots, v_n\}$ is the set of *votes* or *voters*, and $\mathcal{E}$ is the *election system* which is a function mapping (C, V) to a set $W \subseteq C$ called the *winners* of the election. We will express our results for the *winner case* where several winners are possible, but our results can be adapted to the *unique winner case* where W consists of a single candidate only.

In our context, each vote is a strict linear order over the set C, and we denote by $\mathrm{rank}(c, v)$ the position of candidate $c \in C$ in a vote $v \in V$. By contrast, the concept of partial votes, mentioned only occasionally in this paper, can be used to describe partial orders over the candidates.

For an overview of different election systems, we refer to [7]. We will mainly focus on election systems that are characterized by a given *scoring rule*, expressed as a vector $(s_1, s_2, \ldots, s_m)$. Given such a scoring rule, the *score* of a candidate c in a vote v, denoted by $\mathrm{score}(c, v)$, is $s_{\mathrm{rank}(c,v)}$. The score of a candidate c in a set of votes V is $\mathrm{score}(c, V) = \sum_{v \in V} \mathrm{score}(c, v)$, and the winners of the election are the candidates that receive the highest score in the given votes.

The election system we are particularly interested in is k-approval, which is defined by the scoring vector $(1, \ldots, 1, 0, \ldots, 0)$, starting with k ones. In the case of $k = 1$, this is the *plurality* rule, whereas $(m-1)$-approval is also known as *veto*. Given a vote v, we will say that a candidate c with $1 \leq \mathrm{rank}(c, v) \leq k$ takes a *one-position* in v, whereas a candidate c' with $k + 1 \leq \mathrm{rank}(c', v) \leq m$ takes a *zero-position* in v.

Swap Bribery, Possible Winner, Manipulation. Given V and C, a *swap* in some vote $v \in V$ is a triple (v, c_1, c_2) where $\{c_1, c_2\} \subseteq C, c_1 \neq c_2$. Given a vote v, we say that a swap $\gamma = (v, c_1, c_2)$ is *admissible in* v if $\mathrm{rank}(c_1, v) = \mathrm{rank}(c_2, v) - 1$. Applying this swap means exchanging the positions of c_1 and c_2 in the vote v, we denote by v^γ the vote obtained

this way. Given a vote v, a set Γ of swaps is *admissible in* v if the swaps in Γ can be applied in v in a sequential manner, one after the other, in some order. Note that the obtained vote, denoted by v^Γ, is independent of the order in which the swaps of Γ are applied. We also extend this notation for applying swaps in several votes, in the straightforward way.

In a Swap Bribery instance, we are given V, C, and $\mathcal{E}$ forming an election, a preferred candidate $p \in C$, a cost function $c : C \times C \times V \to \mathbb{N}$ mapping for every vote each possible swap to a non-negative integer, and a budget $\beta \in \mathbb{N}$. The task is to determine a set of admissible swaps Γ whose total cost is at most β, such that p is a winner in the $\mathcal{E}$-election (C, V^Γ). Such a set of swaps is called a *solution* of the Swap Bribery instance. The underlying decision problem is the following.

Swap Bribery
Given: An $\mathcal{E}$-election $E = (C, V)$, a preferred candidate $p \in C$, a cost function c mapping each possible swap to a non-negative integer, and a budget $\beta \in \mathbb{N}$.
Question: Is there a set of swaps Γ whose total cost is at most β such that p is a winner in the $\mathcal{E}$-election (C, V^Γ)?

We will also show the connection between Swap Bribery and the Possible Winner problem. In this setting, we have an election where some of the votes may be *partial* orders over C instead of complete linear ones. The question is whether it is possible to extend the partial votes to complete linear orders in such a way that a preferred candidate wins the election. For a more formal definition, we refer to the article by Konczak and Lang [23] who introduced this problem. The corresponding decision problem is defined as follows.

Possible Winner
Given: A set of candidates C, a set of partial votes $V' = (v'_1, \ldots, v'_n)$ over C, an election system $\mathcal{E}$, and a preferred candidate $p \in C$.
Question: Is there an extension $V = (v_1, \ldots, v_n)$ of V' such that each v_i extends v'_i to a complete linear order, and p is a winner in the $\mathcal{E}$-election (C, V)?

A special case of Possible Winner is Manipulation (see e.g. [9, 22, 13]). Here, the given set of partial orders consists of two subsets; one subset contains complete preference orders and the other one completely unspecified votes.

Parameterized complexity, multivariate complexity.

Parameterized complexity is a two-dimensional framework for studying the computational complexity of problems [11, 19, 26]. One dimension is the size of the input I (as in classical complexity theory) and the other dimension is the parameter k (usually a positive integer). A problem is called *fixed-parameter tractable* (FPT) with respect to a parameter k if it can be solved in $f(k) \cdot |I|^{O(1)}$ time, where f is an arbitrary computable function [11, 19, 26]. Multivariate complexity is the natural sequel of the parameterized approach when expanding to multidimensional parameter spaces, see [27]. For example, if we consider a parameterization where the parameter is a pair $k = (k_1, k_2)$, then we refer to this as a *combined parameterization*, or we simply say that both k_1 and k_2 are parameters. In such a case, the desired FPT algorithm should run in time $f(k_1, k_2) \cdot |I|^{O(1)}$ for some f.

The first level of (presumable) parameterized intractability is captured by the complexity class W[1]. A *parameterized reduction* reduces a problem instance (I, k) in $f(k) \cdot |I|^{O(1)}$ time (for some computable function f) to an instance (I', k') such that (I, k) is a yes-instance if and only if (I', k') is a yes-instance, and k' only depends on k but not on $|I|$. To prove W[1]-hardness of a given parameterized problem Q, one needs to present a parameterized reduction from some already known W[1]-hard problem to Q.

We will use the following W[1]-hard problems [10, 18] for the hardness reductions in this work:

Clique
Given: An undirected graph $G = (V, E)$ and $k \in \mathbb{N}$.
Question: Is there a complete subgraph (clique) of G of size k?

Multicolored Clique
Given: An undirected graph $G = (V_1 \cup V_2 \cup \cdots \cup V_k, E)$ with $V_i \cap V_j = \emptyset$ for $1 \leq i < j \leq k$ where the vertices of V_i induce an independent set for $1 \leq i \leq k$.
Question: Is there a complete subgraph (clique) of G of size k?

We will also make use of a *kernelization* algorithm in this work, which is a standard technique for obtaining fixed-parameter results, see [6, 21, 26]. The idea is to transform the input instance (I, k) in a polynomial-time preprocessing step via *data reduction rules* into a "reduced" instance (I', k') such that two conditions hold: First, (I, k) is a yes-instance if and only if (I', k') is a yes-instance, and second, the size of the reduced instance depends on the parameter only, i.e. $|I'| + |k'| \leq g(k)$ for some arbitrary computable function g. The reduced instance (I', k') is then referred to as the *problem kernel*. If in addition g is a polynomial function, we say that the problem admits a *polynomial kernel*. The existence of a problem kernel is equivalent to fixed-parameter tractability of the corresponding problem with respect to the particular parameter [26].

3 Complexity depending on the cost function

In this section, we focus our attention on Swap Bribery for k-approval. We start with the case where all costs are equal to 1, for which we obtain polynomial-time solvability. Below we provide an algorithm which for every possible s checks if there is a solution in which the preferred candidate wins with score s. This can be carried out by solving a minimum cost maximum flow problem.

Theorem 1. Swap Bribery *for k-approval is polynomial-time solvable, if all costs are* 1.

Proof. Let V be the set of votes and C be the set of candidates. The score of any candidate is an integer between 0 and $|V|$. Our algorithm finds out for each possible s^* with $1 \leq s^* \leq |V|$ whether there is a solution in which the preferred candidate p wins with score s^*.

Given a value s^*, we answer the above question by solving a corresponding minimum cost maximum flow problem. We will define a network $N = (G, s, t, g, w)$ on a directed graph $G = (D, E)$ with a source vertex s and a target vertex t, where g denotes the capacity function and w the cost function defined on E. See Figure 1 for an illustration of the network. First, we introduce the vertex sets

$A = \{a_{v,c} \mid v \in V, c \in C, \text{rank}(c, v) \leq k\}$,
$A' = \{a'_{v,c} \mid v \in V, c \in C\}$, and
$B = \{b_c \mid c \in C\}$,

and we set $D = \{s, t, x\} \cup A \cup A' \cup B$. We define the arcs E as the union of the sets

$E_S = \{sa \mid a \in A\}$,
$E_A = \{a_{v,c}a'_{v,c} \mid \text{rank}(c, v) \leq k\}$,
$E_{A'} = \{a_{v,c}a'_{v,c'} \mid \text{rank}(c, v) \leq k, \text{rank}(c', v) > k\}$,
$E_B = \{a'_{v,c}b_c \mid v \in V, c \in C\}$,
$E_X = \{b_c x \mid c \in C, c \neq p\}$,

plus the arcs $b_p t$ and xt. We set the cost function w to be 0 on each arc except for the arcs of $E_{A'}$, and we set $w(a_{v,c}a'_{v,c'}) = \text{rank}(c', v) - \text{rank}(c, v)$. We let the capacity g be 1 on the arcs of $E_S \cup E_A \cup E_{A'} \cup E_B$, we set it to be s^* on the arcs of $E_X \cup \{b_p t\}$, and we set $g(xt) = |V|k - s^*$.

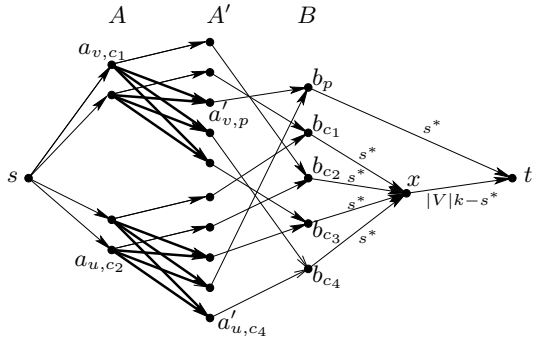

Fig. 1. The network for a small instance with $k = 2$, having 5 candidates and 2 voters, u and v. The votes of the illustrated instance are $v : c_1 \succ c_2 \succ p \succ c_4 \succ c_3$ and $u : c_1 \succ c_2 \succ c_3 \succ p \succ c_4$. Each unlabeled edge has capacity 1; otherwise labels correspond to capacities. Bold edges correspond to bribing voters in order to move a point from one candidate to another candidate. (Moving a point from a candidate c to candidate c' in some vote means swapping c all the way down to the k-th position, swapping c' all the way up to the $(k+1)$-th position, and then swapping c with c'.) The costs of such edges depend on the given votes, e.g. $w(a_{v,c_1}a'_{v,p}) = 1$ and $w(a_{u,c_2}a'_{u,c_4}) = 3$. All other edges have cost 0.

The soundness of the algorithm and hence the theorem itself follows from the following observation: there is a flow of value $|V|k$ on N having total cost at most β if and only if there exists a set Γ of swaps with total cost at most β such that $\text{score}(p, V^\Gamma) = s^*$ and $\text{score}(c, V^\Gamma) \leq s^*$ for any $c \in C, c \neq p$.

First, suppose that such a flow f exists. Since all capacities and costs are integrals, we know that f is integral as well [20]. For each vote $v \in V$, we define a set of swaps on v as follows. We define two sets $X^{\rightarrow}(v)$ and $X^{\leftarrow}(v)$ in a way that if $f(a_{v,c}a'_{v,c'}) = 1$ holds for some c and c' with $c \neq c'$, then we put c into $X^{\rightarrow}(v)$ and we put c' into $X^{\leftarrow}(v)$. Clearly, $|X^{\rightarrow}(v)| = |X^{\leftarrow}(v)|$, by the given capacities. Observe that moving the candidates in $X^{\rightarrow}(v)$ to the positions $k+1, k+2, \ldots, k+h$ and also the candidates in $X^{\leftarrow}(v)$ to the positions $k, k-1, \ldots, k-h+1$ for $h = |X^{\rightarrow}(v)|$ has total cost $\sum_{c' \in X^{\leftarrow}(v)} \text{rank}(c', v) - \sum_{c \in X^{\rightarrow}(v)} \text{rank}(c, v)$. Thus, by letting $\Gamma(v)$ contain these swaps for some v, we know that the cost of the bribery $\Gamma = \{\Gamma(v) \mid v \in V\}$ is exactly the cost of the flow f which is not more than β. Observe that as a result of these swaps, each candidate c other than p will receive at most s^* scores in V^Γ because of the capacity $g(b_c x) \leq s^*$. On the other hand, by $g(xt) = |V|k - s^*$ we get $f(b_p t) = s^*$, which yields that p will receive exactly s^* scores in V^Γ. Thus, Γ has the properties claimed.

For the converse direction, let Γ be a set of swaps with total cost at most β such that $\text{score}(p, V^\Gamma) = s^*$ and $\text{score}(c, V^\Gamma) \leq s^*$ for any $c \in C, c \neq p$. For some $v \in V$, let $X^{\rightarrow}(v)$ denote those candidates c for which $\text{score}(c, v) > \text{score}(c, v^\Gamma)$, and let $X^{\leftarrow}(v)$ denote those candidates c' for which $\text{score}(c', v) < \text{score}(c', v^\Gamma)$. It is easy to see that the swaps applied in v by Γ have total cost at least $\sum_{c \in X^{\leftarrow}(v)} \text{rank}(c', v) - \sum_{c \in X^{\rightarrow}(v)} \text{rank}(c, v)$. Therefore, a flow can be easily constructed having cost at most β in the following way: for each v and c where $\text{score}(c, v) = \text{score}(c, v^\Gamma) = 1$ we let $f(a_{v,c}a'_{v,c}) = 1$, and for each v and c where c is the i-th candidate in $X^{\rightarrow}(v)$ we set $f(a_{v,c}a'_{v,c'}) = 1$ for the i-th candidate C' of $X^{\leftarrow}(v)$ according to some fixed ordering. It is not hard to verify that this indeed determines a flow for N, with value $|V|k$ and cost at most β. □

Note that Theorem 1 implies a polynomial-time approximation algorithm for Swap Bribery for k-approval with approximation ratio δ, if all costs lie within the range $[1, \delta]$ for some $\delta \geq 1$.

Proposition 2 shows the connection between Swap Bribery and Possible Winner. This result is an easy consequence of a reduction given by Elkind et al. [13].

Proposition 2. *The special case of* Swap Bribery *where the costs are in* $\{0, \delta\}$ *for some* $\delta > 0$ *and the budget is zero is equivalent to the* Possible Winner *problem.*

Proof. It has already been proved by Elkind et al. [13] that Possible Winner reduces to Swap Bribery if the possible costs include 0 and 1, and the budget is zero. Clearly, the result also holds if we assume that the costs include 0 and δ for some $\delta > 0$.

For the other direction, it is easy to see that a Swap Bribery instance with costs in $\{0, \delta\}$, $\delta > 0$ and budget zero is equivalent to the Possible Winner instance with the same candidates where each vote v is replaced by the transitive closure of the relation $\succ_v$ for which $a \succ_v b$ holds if and only if a precedes b in the vote v and the cost of swapping a with b in v is non-zero. □

As a corollary, Swap Bribery with costs in $\{0, \delta\}$, $\delta > 0$ and budget zero is NP-complete for almost all election systems based on scoring vectors [3], and also for the voting rules Copeland [28] and Maximin [28]. For many voting systems such as k-approval, Borda, and Bucklin, it is NP-complete even for a fixed number of votes [4]. A further consequence of Proposition 2, contrasting the polynomial-time approximation algorithm implied by Theorem 1, is the fact that approximating Swap Bribery with an arbitrary factor in a setting where zero costs are allowed is NP-hard for all voting rules where the Possible Winner problem is NP-hard. This has been observed by Elkind and Faliszewski [12] as well.

We now turn our attention to the simplest case among those where the cost function is not a constant, i.e. where only two different positive costs are possible. Theorem 3 shows that the corresponding problem is hard already for 2-approval.

Theorem 3. *Suppose that* $\varepsilon > 0$.
(1) Swap Bribery *for 2-approval with costs in* $\{1, 1+\varepsilon\}$ *is NP-complete.*
(2) Swap Bribery *for 2-approval with costs in* $\{1, 1+\varepsilon\}$ *is W[1]-hard, if the parameter is the budget* β*, or equivalently, the maximum number of swaps allowed.*

Proof. We present a reduction from the Multicolored Clique problem. Let $\mathcal{G} = (V, E)$ with the k-partition $V = V_1 \cup V_2 \cup \cdots \cup V_k$ be the given instance of Multicolored Clique. Here and later, we write $[k]$ for $\{1, 2, \ldots, k\}$. For each $i \in [k]$, $x \in V_i$, and $j \in [k] \setminus \{i\}$ we let $E_x^j = \{xy \mid y \in V_j, xy \in E\}$. We construct an instance $I_{\mathcal{G}}$ of Swap Bribery as follows.

The set $\mathcal{C}$ of candidates will be the union of the sets $A, B, C, \widetilde{C}, F, H, \widetilde{H}, M, \widetilde{M}, D, G, T$, and $\{p, r\}$. Table 2 shows the exact definitions of these sets. Our preferred candidate is p. The sets D, G, and T will contain *dummies*, *guards*, and *transporters*, respectively. Our budget will be $\beta = k^3 + 10k^2$. Regarding the indices i and j, we suppose $i, j \in [k]$ if not stated otherwise.

The set of votes will be $W = W_G \cup W_I \cup W_S \cup W_C$. Votes in W_G will define guards (explained later), votes in W_I will set the initial scores, votes in W_S will represent the selection of k vertices, and finally, votes in W_C will be responsible for checking that the selected vertices are pairwise neighboring. We construct W such that the following will hold for some (even) integer K, determined later:

$\text{score}(r, W) = 0$,
$\text{score}(a, W) = K + 1$ for each $a \in A$,
$\text{score}(q, W) = K$ for each $q \in \mathcal{C} \setminus (A \cup D \cup \{r\})$,
$\text{score}(d, W) \leq 1$ for each $d \in D$.

We define the cost function such that each swap has cost 1 or $1 + \varepsilon$. We will define each cost to be 1 if not explicitly stated otherwise. Since each cost is at least 1, none of

candidate set	cardinality
$A = \{a^{i,j} \mid i, j \in [k]\}$	$\lvert A\rvert = k^2$
$B = \{b_v^j \mid j \in [k], v \in V\}$	$\lvert B\rvert = k\lvert V\rvert$
$C = \{c_v^j \mid j \in [k], v \in V\}$	$\lvert C\rvert = k\lvert V\rvert$
$\widetilde{C} = \{\widetilde{c}_v^j \mid j \in [k], v \in V\}$	$\lvert\widetilde{C}\rvert = k\lvert V\rvert$
$F = \{f_v^j \mid j \in [k], v \in V\}$	$\lvert F\rvert = k\lvert V\rvert$
$H = \{h_v^j \mid j \in [k], v \in V\}$	$\lvert V\rvert = k\lvert V\rvert$
$\widetilde{H} = \{h_v^j \mid j \in [k], v \in \bigcup_{i<j} V_i\}$	$\lvert\widetilde{H}\rvert = \sum_{j=1}^{k}(k-j)\lvert V_j\rvert < k\lvert V\rvert$
$M = \{m^{i,j} \mid 1 \le i \le j \le k\}$	$\lvert M\rvert = \binom{k}{2} + k$
$\widetilde{M} = \{\widetilde{m}^{i,j} \mid 1 \le i < j \le k\}$	$\lvert\widetilde{M}\rvert = \binom{k}{2}$
$D = \{d_1, d_2, \dots\}$	$\lvert D\rvert \le \lvert W\rvert = O(k^3\lvert V\rvert^2)$
$G = \{g_1, g_2, \dots, g_{\beta+2}\}$	$\lvert G\rvert = \beta + 2 = k^3 + 10k^2$
$T = \{t_1, t_2, \dots\}$	$\lvert T\rvert = O(k^2\lvert V\rvert)$

Table 2. The candidate sets constructed in the proof of Theorem 3.

the candidates ranked after the position $\beta + 2$ in a vote v can receive non-zero points in v without violating the budget. Thus, we can represent votes by listing only their first $\beta + 2$ positions. A candidate does not *appear* in some vote, if he is not contained in these positions.

Dummies, guards, truncation, and transporters. First, let us clarify the concept of dummy candidates: we will ensure that no dummy can receive more than one point in total, by letting each $d \in D$ appear in exactly one vote. Since we will use at most one dummy in each vote, this can be ensured easily by using at most $|W|$ dummies in total. We will use the sign $*$ to denote dummies in votes.

Now, we define $\beta + 2$ guards using the votes W_G. We let W_G contain votes of the form $w_G(h)$ for each $h \in [\beta + 2]$, each such vote having multiplicity $K/2$ in W_G. We let $w_G(h) = (g_h, g_{h+1}, g_{h+2}, \dots, g_{\beta+2}, g_1, g_2, \dots g_{h-1})$.[3] Note that $\mathrm{score}(g, W_G) = K$ for each $g \in G$, and the total score obtained by the guards in W_G cannot decrease. As we will make sure that p cannot receive more than K points without exceeding the budget, this yields that in any possible solution, each guard must have score exactly K.

Using guards, we can *truncate* votes at any position $h > 2$ by putting arbitrarily chosen guards at the positions $h, h + 1, \dots, \beta + 2$. This way we ensure that only candidates on the first $h - 1$ positions can receive a point in this vote. We will denote truncation at position h by using a sign † at that position.

Sometimes we will need votes which ensure that some candidate q_1 can "transfer" one point to some candidate q_2 using a cost of b from the budget ($b \in \mathbb{N}^+$, $q_1, q_2 \in \mathcal{C} \setminus (D \cup G \cup T)$). In such cases, we construct b votes by using exactly $b - 1$ *transporter* candidates, say $t_1, t_2, \dots, t_{b-1}$. The constructed votes are as follows: for each $h \in [b - 2]$ we add a vote $(*, t_h, t_{h+1}, \dagger)$, and we also add the votes $(*, q_1, t_1, \dagger)$ and $(*, t_{b-1}, q_2, \dagger)$. We let the cost of any swap here be 1, and we denote the obtained set of votes by $q_1 \rightsquigarrow^b q_2$. (Note that $q_1 \rightsquigarrow^1 q_2$ only consists of the vote $(*, q_1, q_2, \dagger)$.)

The idea behind the construction of these transfer votes is the following. We will set the initial score of each transporter candidate in W to K, in order to make sure that none of them can gain additional points in any solution. Also, the candidates $t_1, \dots, t_{b-1}$ will not appear in any other vote than in the votes of $q_1 \rightsquigarrow^b q_2$ (and the votes of W_I adjusting their initial scores). Hence, any bribery swapping q_1 and t_1 in the vote $(*, q_1, t_1, \dagger)$ must also swap t_1 and t_2 in $(*, t_1, t_2, \dagger)$ to prevent t_1 from gaining an extra point. Following this argument for the candidates $t_2, \dots, t_{b-1}$, we can see that the only way q_1 can lose a point in these

[3] For convenience, here we use the vector-style representation of the linear orders given by the voters.

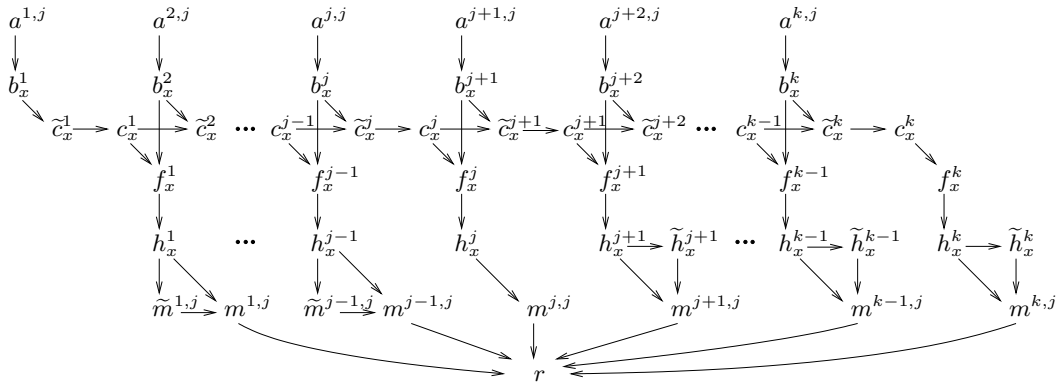

Fig. 2. Part of the instance $I_{\mathcal{G}}$ in the proof of Theorem 3, assuming $x \in V_j$ in the figure. An arc goes from q_1 to q_2 if q_1 can transfer a point to q_2 using one or several swaps.

votes in a successful bribery is to swap the second and the third candidates in each vote of $q_1 \rightsquigarrow^b q_2$. Clearly, this has cost b, and results in q_2 obtaining an additional point.

Setting initial scores. Using dummies and guards, we define W_I to adjust the initial scores of the relevant candidates as follows. We put the following votes into W_I:

$(p, *, \dagger)$ with multiplicity K,	
$(a^{i,j}, *, \dagger)$ with multiplicity $K+1-\lvert V_j\rvert$	for each $i, j \in [k]$,
$(h_x^i, *, \dagger)$ with multiplicity $K - \lvert E_x^i\rvert$	for each $i \in [k], x \in \bigcup_{i<j} V_j$,
$(\widetilde{h}_x^i, *, \dagger)$ with multiplicity $K - \lvert E_x^i\rvert$	for each $i \in [k], x \in \bigcup_{i>j} V_j$,
$(m^{i,j}, *, \dagger)$ with multiplicity $K-2$	for each $i < j$,
$(q, *, \dagger)$ with multiplicity $K-1$	for each remaining $q \notin D \cup G \cup \{r\}$.

The preferred candidate p will not appear in any other vote, implying $\text{score}(p, W) = K$.

Selecting vertices. The set W_S consists of the following votes:

$a^{i,j} \rightsquigarrow^1 b_x^i$	for each $i, j \in [k]$ and $x \in V_j$,
$b_x^1 \rightsquigarrow^2 \widetilde{c}_x^1$	for each $x \in V$,
$w_S(i,x) = (b_x^i, c_x^{i-1}, \widetilde{c}_x^i, f_x^{i-1}, \dagger)$	for each $2 \le i \le k$, $x \in V$,
$c_x^k \rightsquigarrow^2 f_x^k$	for each $x \in V$,
$\widetilde{c}_x^i \rightsquigarrow^1 c_x^i$	for each $i \in [k], x \in V$, and
$f_x^i \rightsquigarrow^{2(k-i)+1} h_x^i$	for each $i \in [k], x \in V$.

Swapping candidate b_x^i with c_x^{i-1}, and swapping candidate $\widetilde{c}_x^i$ with f_x^{i-1} in $w_S(i,x)$ for some $2 \le i \le k$, $x \in V$ will have cost $1+\varepsilon$.

Checking incidency. The set W_C will contain the votes

$h_x^i \rightsquigarrow^1 \widetilde{h}_x^i$	for each $i \in [k]$, $x \in \bigcup_{i>j} V_j$,
$w_C(i,j,y,x) = (h_x^i, \widetilde{h}_y^j, \widetilde{m}^{i,j}, m^{i,j}, \dagger)$	for each $i < j$, $x \in V_j$, $y \in V_i$, $xy \in E$,
$\widetilde{m}^{i,j} \rightsquigarrow^1 m^{i,j}$	for each $i < j$,
$h_x^i \rightsquigarrow^3 m^{i,i}$	for each $i \in [k]$, $x \in V_i$, and
$m^{i,j} \rightsquigarrow^1 r$ with multiplicity 2	for each $i < j$,
$m^{i,i} \rightsquigarrow^1 r$	for each $i \in [k]$.

Again, swapping candidate h_x^i with $\widetilde{h}_y^j$, and also candidate $\widetilde{m}^{i,j}$ with $m^{i,j}$ in a vote of the form $w_C(i,j,y,x)$ will have cost $1+\varepsilon$.

It remains to define K properly. To this end, we let $K \ge 2$ be the minimum even integer not smaller than the integers in the set $\{\lvert E_x^j\rvert \mid j \in [k], x \notin V_j\} \cup \{\lvert V_i\rvert \mid i \in [k]\} \cup \{k^2\}$. This finishes the construction. It is straightforward to verify that the initial scores of the candidates are as claimed above. The constructed instance is illustrated in Figure 2.

Construction time. Note $|W_G| = (\beta+2)K/2$, $|W_I| = O(Kk^2 + Kk|V|) = O(Kk|V|)$, $|W_S| = O(k^2|V|)$, and $|W_C| = O(k|V|+|E|)$. Hence, the number of votes is polynomial in the size of the input graph $\mathcal{G}$. This also implies that the number of candidates is polynomial as well, and the whole construction takes polynomial time. Note also that β is only a function of k, hence this yields a parameterized reduction as well.

If for some vote v, exactly one candidate q_1 gains a point and exactly one candidate q_2 loses a point as a result of the swaps in Γ, then we say that q_2 *sends* one point to q_1, or equivalently, q_1 *receives* one point from q_2 in v according to Γ. Also, if Γ consists of swaps that transform a vote $(a, b, c, d, \dagger)$ into a vote $(c, d, a, b, \dagger)$, then we say that a sends one point to c, and b sends one point to d. A point is *transferred* from q_1 to q_2 in Γ, if it is sent from q_1 to q_2 possibly through some other candidates.

Our aim is to show the following: $\mathcal{G}$ has a k-clique if and only if the constructed instance $I_\mathcal{G}$ is a yes-instance of Swap Bribery. This will prove both (1) and (2).

Main ideas of the proof. In any successful bribery, all of the k^2 candidates in A must lose a point so that they do not beat p. However, as all other candidates except for r (and the dummies) have score equal to p's score, the construction forces these k^2 points to be transferred to candidate r. The candidates through whom these points are transferred to r will encode the selection of vertices, and our constraint on the budget will ensure that the selected vertices form a clique.

In particular, selecting some vertex x as the j-th vertex of the clique corresponds to the choice of the briber to transfer one point from candidate $a^{i,j} \in A$ to candidate $b^i_x \in B$, for each $1 \le i \le k$. Then this point received by b^i_x is further transferred to $h^i_x \in H$ through candidates of $C \cup \widetilde{C} \cup F$. The main role of these additional candidates is to make sure that in an optimal bribery all of the k candidates $a^{1,j}, \ldots, a^{k,j}$ behave "consistently", selecting one unique vertex x as the j-th member of the clique. The argument here relies heavily on our bound on the cost: to obtain optimality, the briber needs certain swaps to be performed in a "parallel" fashion (i.e., within the same vote) to avoid expensive swaps of cost $1+\varepsilon$.

The incidency of the selected vertices is forced by the swaps that an optimal bribery must apply in the votes W_C. In these swaps, for each i and each selected vertex x, the extra point received by h^i_x is transferred to r through candidates of $\widetilde{H} \cup M \cup \widetilde{M}$. Here we follow arguments similar to those used for showing the consistency of the vertex selection: if x and y are the j-th and the i-th vertex selected as a member of the clique, respectively, then the swaps transferring a point from h^i_x to r can be parallelized with those transferring a point from h^j_y to r if and only if x and y are incident. This will ensure the correctness of our reduction.

Direction $\Longrightarrow$. Suppose that $\mathcal{G}$ has a clique consisting of the vertices $x_1, x_2, \ldots, x_k$ with $x_i \in V_i$. We are going to define a set Γ of swaps transforming W into W^Γ with total cost β such that p wins in W^Γ according to 2-approval.

First, we define the swaps applied by Γ in W_S:

- Swap $a^{i,j}$ with $b^i_{x_j}$ for each i and j in $a^{i,j} \rightsquigarrow^1 b^i_{x_j}$. Cost: k^2.
- Transfer one point from $b^1_{x_j}$ to $\widetilde{c}^1_{x_j}$ for each $j \in [k]$ in $b^1_{x_j} \rightsquigarrow^2 \widetilde{c}^1_{x_j}$. Cost: $2k$.
- Apply four swaps in each vote $w_S(i, x_j) = (b^i_{x_j}, c^{i-1}_{x_j}, \widetilde{c}^i_{x_j}, f^{i-1}_{x_j}, \dagger)$ transforming it to $(\widetilde{c}^i_{x_j}, f^{i-1}_{x_j}, b^i_{x_j}, c^{i-1}_{x_j}, \dagger)$, sending one point from $b^i_{x_j}$ to $\widetilde{c}^i_{x_j}$ and simultaneously, also one point from $c^{i-1}_{x_j}$ to $f^{i-1}_{x_j}$. Cost: $4k(k-1)$.
- Swap $\widetilde{c}^i_{x_j}$ with $c^i_{x_j}$ for each $i, j \in [k]$ in $\widetilde{c}^i_{x_j} \rightsquigarrow^1 c^i_{x_j}$. Cost: k^2.
- Transfer one point from $c^k_{x_j}$ to $f^k_{x_j}$ for each $j \in [k]$ in $c^k_{x_j} \rightsquigarrow^2 f^k_{x_j}$. Cost: $2k$.
- Transfer one point from $f^i_{x_j}$ to $h^i_{x_j}$ for each $i, j \in [k]$ in $f^i_{x_j} \rightsquigarrow^{2(k-i)+1} h^i_{x_j}$. Cost: k^3.

The above swaps transfer one point from $a^{i,j}$ to $h^i_{x_j}$ via the candidates $b^i_{x_j}$, $\widetilde{c}^i_{x_j}$, $c^i_{x_j}$, and $f^i_{x_j}$ for each i and j. These swaps of Γ, applied in the votes W_S, have total cost $k^3 + 6k^2$.

Now, we define the swaps applied by Γ in the votes W_C.

- Swap $h^i_{x_j}$ with $\widetilde{h}^i_{x_j}$ for each $j < i$ in $h^i_{x_j} \rightsquigarrow^1 \widetilde{h}^i_{x_j}$. Cost: $k(k-1)/2$.
- Apply four swaps in each vote $w_C(i, j, x_i, x_j) = (h^i_{x_j}, \widetilde{h}^j_{x_i}, \widetilde{m}^{i,j}, m^{i,j}, \dagger)$ transforming it to $(\widetilde{m}^{i,j}, m^{i,j}, h^i_{x_j}, \widetilde{h}^j_{x_i}, \dagger)$, sending one point from $h^i_{x_j}$ to $\widetilde{m}^{i,j}$ and, simultaneously, also one point from $\widetilde{h}^j_{x_i}$ to $m^{i,j}$. Note that $w_C(i, j, x_i, x_j)$ is indeed defined for each i and j, since x_i and x_j are neighboring. Cost: $2k(k-1)$.
- Swap $\widetilde{m}^{i,j}$ with $m^{i,j}$ for each $i < j$ in $\widetilde{m}^{i,j} \rightsquigarrow^1 m^{i,j}$. Cost: $k(k-1)/2$.
- Transfer one point from $h^i_{x_i}$ to $m^{i,i}$ for each $i \in [k]$ in $h^i_{x_i} \rightsquigarrow^3 m^{i,i}$. Cost: $3k$.
- Swap $m^{i,j}$ with r in both of the votes $m^{i,j} \rightsquigarrow^1 r$ for each $i < j$. Cost: $k(k-1)$.
- Swap $m^{i,i}$ with r for each $i \in [k]$ in $m^{i,i} \rightsquigarrow^1 r$. Cost: k.

Candidate r receives k^2 points after all these swaps in Γ. Easy computations show that the above swaps have cost $4k^2$, so the total cost of Γ is $\beta = k^3 + 10k^2$. Clearly,

$\text{score}(p, W^\Gamma) = K$,
$\text{score}(r, W^\Gamma) = k^2 \leq K$,
$\text{score}(a, W^\Gamma) = K$ for each $a \in A$, and
$\text{score}(q, W^\Gamma) = \text{score}(q, W) \leq K$ for all the remaining candidates q.

This means that p is a winner in W^Γ according to 2-approval. Hence, Γ is indeed a solution for $I_\mathcal{G}$, proving the first direction of the reduction.

Direction $\Longleftarrow$. Suppose that $I_\mathcal{G}$ is solvable, and there is a set Γ of swaps transforming W into W^Γ with total cost at most β such that p wins in W^Γ according to 2-approval. We also assume w.l.o.g. that Γ is a solution having minimum cost.

As argued above, $\text{score}(p, W^\Gamma) \leq K$ and $\text{score}(g, W^\Gamma) \geq K$ for each $g \in G$ follow directly from the construction. Thus, only $\text{score}(p, W^\Gamma) = \text{score}(g, W^\Gamma) = K$ for each $g \in G$ is possible. Hence, for any $i, j \in [k]$, by $\text{score}(a^{i,j}, W) = K + 1$ we get that $a^{i,j}$ must lose at least one point during the swaps in Γ. As no dummy can have more points in W^Γ than in W (by their positions), and each candidate in $\mathcal{C} \setminus (A \cup D \cup \{r\})$ has K points in W, the k^2 points lost by the candidates in A can only be transferred by Γ to the candidate r.

By the optimality of Γ, this means that $a^{i,j}$ sends a point to b^i_x in Γ for some unique $x \in V_j$; we define $\sigma(i, j) = x$ in this case. First, we show $\sigma(1, j) = \sigma(2, j) = \cdots = \sigma(k, j)$ for each $j \in [k]$, and then we prove that the vertices $\sigma(1, 1), \ldots, \sigma(k, k)$ form a k-clique in $\mathcal{G}$.

Let B^* be the set of candidates in B that receive a point from some candidate in A according to Γ; $|B^*| = k^2$ follows from the minimality of Γ. Observing the votes in $W_S \cup W_C$, we can see that some $b^i_x \in B^*$ can only transfer one point to r by transferring it to $h^{i'}_x$ via $f^{i'}_x$ for some i' using swaps in the votes W_S, and then transferring the point from $h^{i'}_x$ to r using swaps in the votes W_C. Basically, there are three ways to transfer a point from b^i_x to $h^{i'}_x$:

(A) b^i_x sends one point to f^{i-1}_x in $w_S(i, x)$ at a cost of $3 + 2\varepsilon$, and then f^{i-1}_x transfers one point to h^{i-1}_x. This can be carried out applying exactly $3 + 2(k - i + 1) + 1 = 6 + 2(k - i)$ swaps, having total costs $6 + 2(k - i) + 2\varepsilon$.
(B) b^i_x sends one point to $\widetilde{c}^i_x$ in $w_S(i, x)$, $\widetilde{c}^i_x$ sends one point to c^i_x, c^i_x sends one point to f^i_x, and then the point gets transferred to h^i_x. Again, the number of used swaps is exactly $5 + 2(k - i) + 1 = 6 + 2(k - i)$, and the total cost is at least $6 + 2(k - i)$.
(C) b^i_x sends one point to $\widetilde{c}^i_x$ in $w_S(i, x)$, and then the point is transferred to a candidate $f^{i'}_x$ for some $i' > i$ via the candidates $c^i_x, \widetilde{c}^{i+1}_x, c^{i+1}_x, \ldots, c^{i'}_x$. Again, the number of used swaps is exactly $5 + 2(k - i) + 1 = 6 + 2(k - i)$, and the total cost is at least $6 + 2(k - i)$.

Summing up these costs for each $b^i_x \in B^*$, and taking into account the cost of sending the k^2 points from the candidates of A to B^*, we get that the swaps of Γ applied in the votes W_S must have total cost at least $k^2 + k \left(\sum_{j=1}^{k} 6 + 2(k - i) \right) = k^3 + 6k^2$. Equality can only hold if each $b^i_x \in B^*$ transfers one point to $h^{i'}_x$ for some $i' \geq i$, i.e. either case B or C happens.

Let H^* be the set of those k^2 candidates in H that receive a point transferred from a candidate in B^*, and let us consider now the swaps of Γ applied in the votes W_C that transfer one point from a candidate $h_x^i \in H^*$ to r. Let j be the index such that $x \in V_j$. First, note that h_x^i must transfer one point to $m^{i,j}$ (if $i \leq j$) or to $m^{j,i}$ (if $i > j$). Moreover, independently of whether $i < j$, $i = j$, or $i > j$ holds, this can only be done using exactly 3 swaps, thanks to the role of the candidates in $\widetilde{H}$ and in $\widetilde{M}$. To see this, note that only the below possibilities are possible:

- If $i < j$, then h_x^i sends one point in $w_C(i, j, y, x)$ for some $y \in V_i$ either to $\widetilde{m}^{i,j}$ via two swaps, or to $m^{i,j}$ via three swaps. In the former case, $\widetilde{m}^{i,j}$ must further transfer the point to $m^{i,j}$, which is the third swap needed.
- If $i > j$, then h_x^i first sends one point to $\widetilde{h}_x^i$, and then $\widetilde{h}_x^i$ sends this point either to $\widetilde{m}^{j,i}$ via one swap, or to $m^{j,i}$ via two swaps applied in the vote $w_C(j, i, x, y)$ for some $y \in V_i$. In the former case, $\widetilde{m}^{j,i}$ transfers the point to $m^{j,i}$ via an additional swap. Note that in any of these cases, Γ applies 3 swaps (maybe having cost $3 + \varepsilon$ or $3 + 2\varepsilon$).
- If $i = j$, then h_x^i sends one point to $m^{i,i}$ through 3 swaps.

Thus, transferring a point from h_x^i to r needs 4 swaps in total, and hence the number of swaps applied by Γ in the votes W_C is at least $4k^2$. Now, by $\beta = k^3 + 10k^2$ we know that equality must hold everywhere in the previous reasonings. Therefore, as argued above, each b_x^i must transfer a point to $h_x^{i'}$ for some $i' \geq i$, i.e., only cases B and C might happen from the above listed possibilities. Now, we are going to argue that only case B can occur.

Let us consider the multiset I_B containing k^2 pairs of indices, obtained by putting (i, j) into I_B for each $b_x^i \in B^*$ with $x \in V_j$. It is easy to see that $I_B = \{(i, j) \mid 1 \leq i, j \leq k\}$. Similarly, we also define the multiset I_H containing k^2 pairs of indices, obtained by putting (i, j) into I_H for each $h_x^i \in H^*$ with $x \in V_j$. By the previous paragraph, I_H can be obtained from I_B by the following operation, possibly applied several times: we take some pair (i, j) from I_B and increase its first member, i.e. we replace (i, j) with (i', j) for some $i' > i$. Let the *measure* of a multiset of pairs I be $\mu(I) = \sum_{(i,j)\in I} i + j$. Then, $\mu(I_H) \geq \mu(I_B) = k^2(k + 1)$.

By the above arguments, if for some $i < j$ the pair (i, j) is contained with multiplicity m_1 in I_H, and (j, i) is contained with multiplicity m_2 in I_H, then the candidate $m^{i,j}$ has to send $m_1 + m_2$ points to r. Similarly, if (i, i) is contained in I_H with multiplicity m, then $m^{i,i}$ has to send m points to r. Thus, $\mu(I_H)$ equals the value obtained by summing up $i + j$ for each $m^{i,j}$ and for each point transferred from $m^{i,j}$ to r. However, each $m^{i,j}$ (where $i < j$) can only send two points to r, and each $m^{i,i}$ can only send one point to r, implying $\mu(I_H) \leq \sum_{i\in[k]}(i + i) + 2\sum_{1\leq i<j\leq k}(i + j) = k^2(k + 1) = \mu(I_B)$. Hence, the measures of I_B and I_H must be equal, from which $I_H = I_B$ follows. Thus, only case B can happen.

Therefore, Γ must send one point from b_x^i to $\widetilde{c}_x^i$ at a cost of 2, and apply three more swaps of cost 3 to transfer one point from $\widetilde{c}_x^i$ to f_x^i. But in the case $i \geq 2$, this can only be done avoiding any swap of cost $1 + \varepsilon$ in the vote $w_S(i, x)$, if f_x^{i-1} simultaneously receives one point from c_x^{i-1} in $w_S(i, x)$ as well, which implies $b_x^{i-1} \in B^*$. Applying this argument iteratively, this shows that $b_x^i \in B^*$ implies $\{b_x^h \mid h < i\} \subseteq B^*$. Hence, B^* is the union of k sets of the form $\{h_x^1, h_x^2, \ldots, h_x^k\}$, implying $\sigma(1, j) = \sigma(2, j) = \cdots = \sigma(k, j)$ for each $j \in [k]$.

Finally, consider the swaps that transfer one point from $h_x^i \in H^*$ to $m^{i,j}$ in W_C where $x \in V_j$ and $i < j$. We know that if $x \in V_j$, then this must be done by applying some swaps in the vote $w_C(i, j, y, x)$ for some $y \in V_i$ such that $xy \in E$. But because of our budget, each such swap must have cost 1 and not $1 + \varepsilon$, which can only happen if Γ transforms $w_C(i, j, y, x) = (h_x^i, \widetilde{h}_y^j, \widetilde{m}^{i,j}, m^{i,j}, \dagger)$ into $(\widetilde{m}^{i,j}, m^{i,j}, h_x^i, \widetilde{h}_y^j, \dagger)$. But this implies that h_y^j must also be in B^*, implying $y = \sigma(j, i)$. Therefore we obtain that $\sigma(i, j)$ and $\sigma(j, i)$ must be vertices connected by an edge in $\mathcal{G}$. This proves the existence of a k-clique in $\mathcal{G}$, proving the theorem. □

Looking into the proof of Theorem 3, we can see that the results hold even in the following restricted case:

- the costs are uniform in the sense that swapping two given candidates has the same price in any vote, and
- the maximum number of swaps allowed in a vote is four.

By applying minor modifications to the given reduction, Theorem 3 can be generalized to hold for the following modified versions as well.

- If we want p to be the unique winner: we only have to set $\text{score}(p, W) = K + 1$.
- If we use k-approval for any fixed k with $k \geq 3$ instead of 2-approval: it suffices to insert $k - 2$ dummies into the first $k - 2$ positions of each vote.[4]

We can summarize these generalizations of Theorem 3 in the following theorem, which follows directly from the discussion above.

Theorem 4 (Generalization of Theorem 3). *For any constant $k \geq 2$,* Swap Bribery *for k-approval is W[1]-hard when parameterized by the value of the budget, assuming that the minimum cost of a swap is 1; this holds even if the following restrictions apply:*

- *there are only two different positive costs possible for a swap, i.e. each swap has a cost in $\{1, 1 + \epsilon\}$ for some $\epsilon > 0$,*
- *the cost of swapping two given candidates is the same in each vote, and*
- *the maximum number of swaps allowed in a vote is 4.*

4 Parameterizing with the number of candidates

In this section, we will consider the parameter 'number of candidates'. For this case, the following definition is helpful.

Let $S_m = \{\pi_1, \pi_2, \ldots, \pi_{m!}\}$ be the set of all permutations on m elements. We say that an election system is *described by linear inequalities*, if for a given set $C = \{c_1, c_2, \ldots, c_m\}$ of candidates it can be characterized by $f(m)$ sets $A_1, A_2, \ldots A_{f(m)}$ (for some computable function f) of linear inequalities over $m!$ variables $x_1, x_2, \ldots, x_{m!}$ in the following sense: if n_i denotes the number of those votes in a given election E that order C according to π_i, then the first candidate c_1 is a winner of the election if and only if for at least one index i, the setting $x_j = n_j$ for each j satisfies all inequalities in A_i. Let us remark that Faliszewski et al. independently defined a very similar notion in the context of multimode control problems [16].

It is easy to see that many election systems can be described by linear inequalities: any system based on scoring rules, Copeland$^\alpha$ ($0 \leq \alpha \leq 1$), Maximin, Bucklin, and Ranked pairs. For example, k-approval is described by the following set A_1 of linear inequalities:

$$A_1 : \quad \sum_{i:\text{rank}(c_1,\pi_i)\leq k} x_i \geq \sum_{i:\text{rank}(c_j,\pi_i)\leq k} x_i \quad \text{for each } 2 \leq j \leq m,$$

where $\text{rank}(c_j, \pi_i)$ denotes the position of candidate c_j in the linear order corresponding to the permutation π_i.

To see an example where we need more than one set of linear inequalities, consider the Bucklin rule. The *Bucklin winning round* in an election is the smallest number b such that

[4] Note that the number of candidates in the constructed instance will depend on the value of k, so in particular, the result does not hold for voting rules such as veto or $(m - 2)$-approval.

there exists a candidate that is ranked in the first b positions in at least $\lfloor \frac{n}{2} \rfloor + 1$ votes (where n is the number of votes in the election). According to Bucklin, the winners of an election with Bucklin winning round b are those candidates that have maximal b-approval score, i.e. that are ranked in the first b positions by the maximum number of votes. Note that the b-approval score of each winner must be at least $\lfloor \frac{n}{2} \rfloor + 1$. This voting system can be described by the following sets of linear inequalities $A_1, A_2, \ldots, A_m$ where A_b corresponds to the case where the Bucklin winning round is exactly b.

$$
\begin{aligned}
A_b: \quad & \sum_{i:\mathrm{rank}(c_j,\pi_i)\leq b-1} x_i \leq \left\lfloor \frac{n}{2} \right\rfloor && \text{for each } 1 \leq j \leq m, \\
& \sum_{i:\mathrm{rank}(c_1,\pi_i)\leq b} x_i \geq \left\lfloor \frac{n}{2} \right\rfloor + 1, && \\
& \sum_{i:\mathrm{rank}(c_1,\pi_i)\leq b} x_i \geq \sum_{i:\mathrm{rank}(c_j,\pi_i)\leq b} x_i && \text{for each } 2 \leq j \leq m.
\end{aligned}
$$

In the above description, the linear inequalities in the first line mean that the Bucklin winning round is at least b. The second line implies that c_1 has b-approval score at least $\lfloor \frac{n}{2} \rfloor + 1$, and the third set of inequalities requires that no candidate has greater b-approval score than c_1. Clearly, c_1 is a winner according to Bucklin if and only if each linear inequality of A_b is satisfied by setting $x_j = n_j$ $(1 \leq j \leq m)$ for at least one set A_b among the sets $A_1, A_2, \ldots, A_m$.

Theorem 5. Swap Bribery *is FPT if the parameter is the number of candidates, for any election system described by linear inequalities.*

Proof. Let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of candidates, where c_1 is the preferred one, and let $A_1, A_2, \ldots A_{f(m)}$ be the sets of linear inequalities over variables $x_1, \ldots, x_{m!}$ describing the given election system $\mathcal{E}$. For some $\pi_i \in S_m$, let v_i denote the vote that ranks C according to π_i. We describe the set V of votes by writing n_i for the multiplicity of the vote v_i in V.

Our algorithm solves $f(m)$ integer linear programs with variables $T = \{t_{i,j} \mid i \neq j, 1 \leq i, j \leq m!\}$. We will use $t_{i,j}$ to denote the number of votes v_i that we transform into votes v_j; we will require $t_{i,j} \geq 0$ for each $i \neq j$. Let V^T denote the set of votes obtained by transforming the votes in V according to the variables $t_{i,j}$ for each $i \neq j$. Such a transformation from V is feasible if $\sum_{j \neq i} t_{i,j} \leq n_i$ holds for each $i \in [m!]$ (inequality $\mathcal{A}$). By [13], we can compute the price $c_{i,j}$ of transforming the vote v_i into v_j in $O(m^3)$ time. Transforming V into V^T can be done with total cost at most β, if $\sum_{i,j \in [m!]} t_{i,j} c_{i,j} \leq \beta$ (inequality $\mathcal{B}$).

We can express the multiplicity x'_i of the vote v_i in V^T as $x'_i = n_i + \sum_{j \neq i} t_{j,i} - \sum_{i \neq j} t_{i,j}$. For some $i \in [f(m)]$, let A'_i denote the set of linear inequalities over the variables in T that are obtained from the linear inequalities in A_i by substituting x_i with the above given expression for x'_i. Using the description of $\mathcal{E}$ with the given linear inequalities, we know that the preferred candidate c_1 wins in the $\mathcal{E}$-election (C, V^T) for some values of the variables $t_{i,j}$ if and only if these values satisfy the inequalities of A'_i for at least one $i \in [f(m)]$. Thus, our algorithm solves Swap Bribery by finding a non-negative assignment for the variables in T that satisfies both the inequalities $\mathcal{A}$, $\mathcal{B}$, and all inequalities in A'_i for some i.

Solving such a system of linear inequalities can be done in linear FPT time, if the parameter is the number of variables [24]. By $|T| = (m! - 1)m!$ the theorem follows. □

Similarly, we can also show fixed-parameter tractability for other problems if the parameter is the number of candidates, e.g. for Possible Winner (this was already obtained by Betzler et al. for several voting systems [4]), Manipulation (both for weighted and

unweighted voters), several variants of CONTROL (this result was obtained for Llull and Copeland voting by Faliszewski et al. [17]), or LOBBYING [8] (here, the parameter would be the number of issues in the election). Since our topic is SWAP BRIBERY, we omit the details.

5 Parameterizing with the number of votes

In this section, we consider SWAP BRIBERY for k-approval where the number of votes is a parameter. First, note that if k is unbounded and is part of the input, then the problem is NP-complete even for a single vote [13, Theorem 4.5]. Hence, we consider parameterizations of SWAP BRIBERY for k-approval where not only the number n of votes, but also either k or the budget β is regarded as a parameter.

We first recall that there is a simple brute force algorithm given in [13] for SWAP BRIBERY that runs in $m^{O(kn)}$ time. Looking at this running time, one can wonder whether it is possible to get k or n out of the exponent of m. Theorem 6, which makes use of the technique of color-coding [1], answers this question in the affirmative for the case of n, by providing an algorithm for SWAP BRIBERY for k-approval which is fixed-parameter tractable with parameter n, supposing that k is some fixed constant. Note that by Theorem 3, this result is best possible in the sense that the problem without parameterization remains NP-hard even if $k = 2$.

By contrast, we will see in Theorem 7 that we cannot expect a similar result for the case where n is constant but k is a parameter.

Theorem 6. SWAP BRIBERY *for k-approval can be solved with a randomized algorithm in $2^{2(\log n+\log k)nk}O(m^{k+1})$ expected time. The derandomized version of the algorithm has running time $2^{O((\log n+\log k)nk)}O(m^{k+1}\log m)$.*

Proof. We are going to apply the idea of color-coding [1] widely used to design parameterized algorithms. Let I be the given instance of SWAP BRIBERY with $V = \{v_1, \ldots, v_n\}$ and $C = \{p, c_1, \ldots, c_{m-1}\}$ denoting the set of votes and the set of candidates, respectively, where p is our preferred candidate.

For a set Γ of swaps for I, we call those candidates whose score in V^Γ is at least 1 *relevant* with respect to Γ, and denote the set of candidates that are relevant w.r.t. Γ by $C^{rel}(\Gamma)$. Clearly, $|C^{rel}(\Gamma)| \leq kn$ always holds, and if Γ is a solution for I, then p must be relevant w.r.t. Γ.

The key idea is to focus on the "pattern" of the relevant candidates in the bribed election. To this end, let us introduce some definitions that capture the structure of a solution. Let us call any k-size subset of the set $[nk] = \{1, \ldots, nk\}$ a *vote pattern*. We can think of a vote pattern as an "anonymized" version of the set of candidates that are in the first k positions in some vote; the elements of $[nk]$ serve as identifiers. Furthermore, we call an n-tuple of vote patterns an *election pattern*. Intuitively, an election pattern encodes the set of relevant candidates w.r.t. some bribery, reflecting also the way these candidates appear in the votes of the bribed election. Again, we can think of the nk integers as identifiers for these relevant candidates.

To describe the exact connection between election patterns and briberies, we need some additional concepts. We say that Γ is *compatible* with an election pattern $(P_1, \ldots, P_n)$, if there is an injective mapping $\sigma : C^{rel}(\Gamma) \to [nk]$ assigning different integers to each relevant candidate w.r.t. Γ such that for each vote $v_i \in V$, the set of integers assigned by σ to the first k candidates in the bribed vote v_i^Γ is exactly the vote pattern P_i; we also require that the preferred candidate is mapped to 1, i.e. $\sigma(p) = 1$. We say that the mapping σ is a *witness* for this compatibility. See Figure 3 for an illustration.

While election patterns capture the structure of any bribery using the notion of compatibility, we want to focus on the structure of those briberies that make p a winner. We say

V: $p \succ a \succ b \succ c$, $a \succ b \succ c \succ p$, $a \succ b \succ p \succ c$. election pattern $\mathcal{P} = (\{1,2\},\{2,3\},\{1,3\})$.

V^Γ: $p \succ \underline{b \succ a} \succ c$, $a \succ b \succ c \succ p$, $a \succ \underline{p \succ b} \succ c$. witness for the compatibility of Γ and $\mathcal{P}$: $p \mapsto 1$, $a \mapsto 3$, $b \mapsto 2$

V^Λ: $p \succ a \succ b \succ c$, $a \succ b \succ c \succ p$, $\underline{b \succ p \succ a} \succ c$. witness for the compatibility of Λ and $\mathcal{P}$: $p \mapsto 1$, $a \mapsto 2$, $b \mapsto 3$

Fig. 3. Illustration for the notion of election patterns and compatibility. The example shows a 2-approval election with candidate set $\{p, a, b, c\}$ and a set V of three votes. Our preferred candidate is p. If we assume that each swap has unit cost, then the briberies Γ and Λ are both optimal, having a cost of 2. (The places where swaps occured during the bribery are underlined.) The election pattern $\mathcal{P}$ is successful, and it is compatible with both Γ and Λ, as shown by the corresponding witnesses.

that an election pattern $\mathcal{P} = (P_1, \ldots, P_n)$ is *successful*, if the element 1 appears at least as many times in $\mathcal{P}$ as any other element, i.e. if $|\{i \mid 1 \in P_i\}| \geq |\{i \mid c \in P_i\}|$ for any $c \in [nk]$. The importance of this definition comes from the following two observations. On the one hand, if Γ is a solution, then any election pattern compatible with Γ is successful. On the other hand, if a bribery is compatible with a successful election pattern and its cost does not exceed the given budget, then it yields a solution for I.

Making use of these observations, our algorithm does the following: it enumerates every possible successful election pattern, and for each such pattern it looks for the cheapest bribery compatible with it. Note that there are at most $\binom{nk}{k}^n < (nk)^{nk}$ possible election patterns to check.

Given a successful election pattern $\mathcal{P} = (P_1, \ldots, P_n)$, let $A = \bigcup_{i \in [n]} P_i \setminus \{1\}$. We describe an algorithm $\mathcal{A}$ that, assuming that there exists a solution compatible with $\mathcal{P}$, finds a solution in $(nk)^{nk} O(m^{k+1})$ randomized time. So let us suppose from now on that I admits a solution Γ that is compatible with $\mathcal{P}$, and let σ be a witness for this. (Note that we do not know σ.) Our algorithm applies color-coding as follows: it colors each candidate in $C \setminus \{p\}$ with the colors of A randomly using a uniform and independent distribution. Let $\alpha(c)$ denote the color of a candidate c; we set $\alpha(p) = 1$. We say that the coloring α is *nice* if $\alpha(c) = \sigma(c)$ for each $c \in C^{rel}(\Gamma)$. Clearly, a random coloring α is nice with probability at least $|A|^{-|A|} \geq (nk-1)^{-(nk-1)}$.

Assuming that we have a nice coloring α, we can find a solution for I as follows. For each $v_i \in V$, we take every possible subset $C' \subseteq C$ of size k whose colors correspond to the vote pattern P_i, i.e. such that $\bigcup_{c \in C'} \alpha(c) = P_i$ holds. For each such C', we compute the minimum cost of a bribery that moves the candidates of C' to the first k positions in v_i. This can be done in $O(mk)$ time for some C', by simply swapping each candidate $c' \in C'$ with exactly those candidates in $C \setminus C'$ that precede c' in the vote v_i. Now, for each $v_i \in V$ we take the cheapest one among all these briberies over all possible sets C' colored by the colors of P_i; let B_i be the resulting bribery for v_i. We claim that the union of the swaps in $B_1, \ldots, B_n$ is a bribery B that yields a solution. Observe that B can be computed in $n\binom{m}{k} O(mk)$ time.

To prove our claim, first note that by our assumptions that Γ is a solution compatible with $\mathcal{P}$ and α is nice, we get that the bribery B cannot have cost greater than the cost of Γ, as the algorithm must have considered the restriction of Γ on v_i when choosing B_i for some i. Thus, B does not exceed the budget. It remains to show that p is a winner in V^B. First, observe that if B is compatible with $\mathcal{P}$, then this follows from the fact that $\mathcal{P}$

is a successful election pattern. Unfortunately, it might happen that B is not compatible with $\mathcal{P}$; the reason for this is that different candidates that in $C^{rel}(B)$ might have the same color. However, this will not cause any problems, since the score of any candidate c relevant w.r.t. B in V^B is upper-bounded by the number of occurrences of the element $\alpha(c)$ in the election pattern $\mathcal{P}$. Since $\mathcal{P}$ is successful, this latter cannot be greater than the score of p in V^B. In other words, p is a winner in V^B because for each $c \in C \setminus \{p\}$ we have

$$\text{score}(c, V^B) \leq |\{i : \alpha(c) \in P_i\}| \leq |\{i : 1 \in P_i\}| = \text{score}(p, V^B).$$

With this method, if the coloring is nice, then algorithm $\mathcal{A}$ finds a solution. By the above arguments, the randomized version of algorithm $\mathcal{A}$ finds a solution in $(nk)^{nk}O(m^{k+1})$ expected time, provided that there exists a solution compatible with $\mathcal{P}$. Therefore, checking every possible successful election pattern takes $(nk)^{2nk}O(m^{k+1}) = 2^{2(\log n+\log k)nk}O(m^{k+1})$ randomized time.

To derandomize the algorithm, we can apply standard techniques using nk-perfect hash functions [1] instead of randomly coloring the candidates of $C \setminus \{p\}$. This yields a deterministic algorithm with $2^{O((\log n+\log k)nk)}O(m^{k+1} \log m)$ running time. □

Next, we complement Theorem 6 by proving that there is no hope for getting k out of the exponent of m in any algorithm solving Swap Bribery for k-approval, as this problem remains W[1]-hard with parameter k even in the case $n = 1$, i.e. if there is only one vote in the instance.

Theorem 7. Swap Bribery *for k-approval with only one vote is W[1]-hard when parameterized by k.*

Proof. We provide a parameterized reduction from the W[1]-hard Clique problem, parameterized by the size of the desired clique. Let $G = (V, E)$ be the input graph given with $V = \{v_1, \ldots, v_N\}$, and let k be the parameter given. We are going to construct an instance I of Swap Bribery for $(k+1)$-approval consisting of an election with a single vote w, a cost function c, and a budget $\beta = (N-k)N^2 + kN - \binom{k}{2}$. We let $C = \{d_1, \ldots, d_{k+1}, c_1, \ldots, c_N, p\}$ denote the set of candidates, and we let $C_v = \{c_1, \ldots, c_N\}$. Our preferred candidate is p. The vote w is defined to be

$$w : d_1 \succ \cdots \succ d_{k+1} \succ c_1 \succ \cdots \succ c_N \succ p.$$

The values of the cost function, shown also in Table 3, are as follows. (For simplicity, the cost of swapping two candidates x_1 and x_2 in the vote w is denoted by $c(x_1, x_2)$ instead of $c(x_1, x_2, w)$, as there is only one voter.) We define the cost of swapping c_i with c_j for some $i, j \in [N]$ to be 1 if $v_i v_j$ is an edge in G, and 0 otherwise. We set the cost of swapping d_1 with $c_i \in C_v$ to be $N - |\{j < i \mid v_j v_i \in E\}|$. Furthermore, we let the cost of swapping p with any candidate $c_i \in C_v$ be N^2. All remaining swaps have zero cost. The construction takes time polynomial in $|V| + k$.

We claim that the constructed instance I is equivalent with the input of Clique in the sense that I has a solution if and only if G has a clique of size k.

First, note that $\beta < (N - k + 1)N^2$, which implies that any solution Γ can swap p with at most $N - k$ candidates from C_v. Therefore, p can only obtain a point in w^Γ if $\text{rank}(p, w^\Gamma) = k + 1$ and there exist k candidates $c_{i_1}, \ldots, c_{i_k}$ that precede p. The cost of a minimum bribery achieving this is

$$(N-k)N^2 + kN - \sum_{1 \leq h < j \leq k} c(c_{i_h}, c_{i_j}).$$

$c(c_i, c_j) = 1$	for each $i, j \in [N]$ where $v_i v_j \in E$
$c(c_i, c_j) = 0$	for each $i, j \in [N]$ where $v_i v_j \notin E$
$c(d_1, c_i) = N - \lvert\{j < i \mid v_j v_i \in E\}\rvert$	for each $i \in [N]$
$c(c_i, p) = N^2$	for each $i \in [N]$
$c(d_1, q) = 0$	for any $q \in C \setminus C_v$
$c(d_j, q) = 0$	for any $2 \leq j \leq k+1$ and $q \in C$

Table 3. The values of the cost function c in the proof of Theorem 7.

The first term in this sum is the cost of swapping p with the candidates $C \setminus \{c_{i_1}, \ldots, c_{i_k}, p\}$, and the remaining terms correspond to swapping the candidates $c_{i_1}, \ldots, c_{i_k}$ with every other candidate before them. Note that by the definition of the cost function, swapping c_{i_j} with *every* candidate preceding it in w has cost N, but we do not swap c_{i_j} with the candidates $c_{i_1}, \ldots, c_{i_{j-1}}$. Thus, it is clear that the cost of such a bribery is at most β if and only if $c(c_{i_h}, c_{i_j}) = 1$ for every $1 \leq h < j \leq k$. This holds if and only if the set $\{v_{i_1}, \ldots, v_{i_k}\}$ is a clique in G, showing that a solution exists if and only if there is a clique of size k in G. This implies the correctness of the reduction. □

Finally, we present a kernelization algorithm for the Swap Bribery problem for k-approval, where we consider both the budget β and the number n of votes as parameters.

Theorem 8. *If the minimum cost is* 1*, then* Swap Bribery *for k-approval (where k is part of the input) with combined parameters (n, β) admits a kernel with $O(n^2\beta)$ votes and $O(n^2\beta^2)$ candidates. Here, n is the number of votes and β is the budget.*

Proof. Let V, C, $p \in C$, and β denote the set of votes, the set of candidates, the preferred candidate, and the budget given, respectively; we write $|V| = n$. The idea of the kernelization algorithm is that not all candidates are interesting for the problem: only candidates that can be moved within the budget β from a zero-position to a one-position or vice versa are relevant.

Let Γ be a set of swaps with total cost at most β. Clearly, as the minimum possible cost of a swap is 1, we know that there are only 2β candidates c in a vote $v \in V$ for which $\mathrm{score}(c, v) \neq \mathrm{score}(c, v^\Gamma)$ is possible, namely, such a c has to fulfill $k - \beta + 1 \leq \mathrm{rank}(c, v) \leq k + \beta$. Thus, there are at most $2\beta n$ candidates for which $\mathrm{score}(c, V) \neq \mathrm{score}(c, V^\Gamma)$ is possible; let us denote the set of these candidates by $\widetilde{C}$. Let c^* be a candidate in $C \setminus \widetilde{C}$ whose score is the maximum among the candidates in $C \setminus \widetilde{C}$.

Note that a candidate $c \in C \setminus (\widetilde{C} \cup \{c^*, p\})$ has no effect on the answer to the problem instance. Indeed, if $\mathrm{score}(p, V^\Gamma) \geq \mathrm{score}(c^*, V^\Gamma)$, then the score of c is not relevant, and conversely, if $\mathrm{score}(p, V^\Gamma) < \mathrm{score}(c^*, V^\Gamma)$ then p loses anyway. Therefore, we can disregard each candidate in $C \setminus \widetilde{C}$ except for c^* and p.

The kernelization algorithm constructs an instance K which is equivalent to the original instance I as follows. In K, neither the budget, nor the preferred candidate will be changed. However, we will change the value of k to be $\beta + 1$, so the kernel instance K will contain a $(\beta + 1)$-approval election[5]. We define the set V_K of votes and the set C_K of candidates in K as follows.

First, the algorithm "truncates" each vote v, by deleting all its positions (together with the candidates in these positions) except for the 2β positions between $k - \beta + 1$ and $k + \beta$. Then again, we shall make use of dummy candidates (see the proof of Theorem 3); we will ensure $\mathrm{score}(d, V^\Gamma) \leq 1$ for each such dummy d. Swapping a dummy with any other

[5] We use $\beta + 1$ instead of β to avoid complications with the case $\beta = 0$.

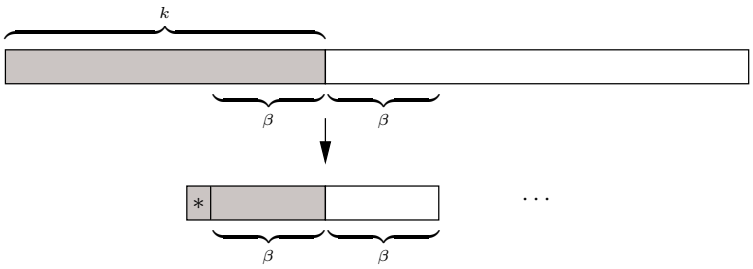

Fig. 4. Illustration of the truncation of the votes by the algorithm of Theorem 8. The grey and white areas show the one-positions and the zero-positions in the depicted vote, respectively. The symbol $*$ stands for a dummy candidate.

candidate will have cost 1 in K. Now, for each obtained truncated vote, the algorithm inserts a dummy candidate in the first position, so that the obtained votes have length $2\beta + 1$. In this step, the algorithm also determines the set $\widetilde{C}$ and the candidate c^*. This can be done in linear time. We denote the votes[6] obtained in this step by V_r. We do not change the costs of swapping candidates of $\widetilde{C} \cup \{c^*, p\}$ in some vote $v \in V_r$. For an illustration of this step, see Figure 4.

Next, to ensure that K is equivalent to the original instance, the algorithm constructs a set V_d of votes such that $\text{score}(c, V_r \cup V_d) = \text{score}(c, V)$ holds for each candidate c in $\widetilde{C} \cup \{p, c^*\}$. This can be done by constructing $\text{score}(c, V) - \text{score}(c, V_r)$ newly added votes where c is on the first position, and all the next 2β positions are taken by dummies. This way we ensure $\text{score}(c, V_d) = \text{score}(c, V_d^{\Gamma})$ for any set Γ of swaps with total cost at most β.

If D is the set of dummy candidates created so far, then let $C_K = \widetilde{C} \cup \{p, c^*\} \cup D$. To finish the construction of the votes, it suffices to add for each vote $v \in V_r \cup V_d$ the candidates not yet contained in v, by appending them at the end (starting from the $(2\beta+1)$-th position) in an arbitrary order. The obtained votes will be the votes V_K of the kernel.

The presented construction needs polynomial time. It is straightforward to verify that for any bribery Γ for I with cost at most β, there is an analogous bribery Γ_K for K which performs the same swaps in the votes V_r as Γ does in the corresponding votes in V. Also, Γ_K will have the same cost and the same effect as Γ, the latter meaning that $\text{score}(c, V^{\Gamma}) = \text{score}(c, V_K^{\Gamma_K})$ will hold for each $c \in \widetilde{C} \cup \{p, c^*\}$. Since any dummy can obtain at most one point in $V_K^{\Gamma_K}$, this shows that if I is solvable then so is K.

For the other direction, we need to observe that no successful bribery for K that is also minimal can contain any swap which involves a dummy. Thus, a minimal bribery solving the kernelized instance contains only swaps in the votes V_r. Clearly, performing these swaps in the corresponding votes of the original instance results in a successful bribery for I, showing the equivalence of the original and the kernel instance. Thus, it remains to bound the size of K.

Clearly, $|\widetilde{C} \cup \{p, c^*\}| \leq 2n\beta + 2$. The number of dummies introduced in the first phase is exactly $|V_r| = n$. As the score of any candidate in V is at most n, the number of votes created in the second phase is at most $(2n\beta+2)n$, which implies that the number of dummies created in this phase is at most $(2n\beta+2)n \cdot 2\beta$. This shows $|C_K| \leq n+(2n\beta+2)(2n\beta+1) = O(n^2\beta^2)$, and also $|V_K| \leq (2n\beta + 3)n = O(n^2\beta)$. □

We remark that if each cost is at least 1, then a kernel with $(k + \beta)n$ candidates and n votes is easy to obtain, by simply deleting every candidate from the instance whose rank is

[6] In fact, these cropped votes are not real votes yet in the sense that they do not contain each candidate.

greater than $k + \beta$ in all of the votes. This simple method might be favorable to the above result in cases where k is small.

6 Conclusion

We have taken the first step towards parameterized and multivariate investigations of Swap Bribery under certain voting systems, focusing on k-approval. We discussed how the complexity of this problem depends on the cost function. In response to an initiation of Elkind et al. [13] to identify natural cases of Swap Bribery that are computationally tractable, we showed that the case where all swaps have equal costs is polynomial-time solvable. By contrast, as soon as we have two different costs, the problem becomes NP-complete for k-approval for any fixed $k \geq 2$, and even W[1]-hard if the parameter is the budget β.

We provided a rather general result showing that Swap Bribery is FPT for a very large class of voting systems if the parameter is the number of candidates. This revaluates previous NP-hardness results: Swap Bribery could be computed efficiently if the number of candidates is small, which is a common setting e.g. in presidential elections. The technique used can be applied to different problems from voting theory, leading to fixed-parameter tractability with respect to the number of candidates in various settings.

We also shed some light on the complexity of Swap Bribery for k-approval when considering combined parameters. We hope that our results will help in understanding the intricate issues of the interplay between the parameters 'number of votes', the budget, or the value of k. On one hand, we strengthened the known NP-completeness result for a single vote by showing W[1]-hardness with respect to k in the case where k is part of the input. On the other hand, we proposed an FPT algorithm for the case where the parameter is the number of votes, but k is a fixed constant. In addition, we presented a polynomial kernel for the problem where the parameters are the number of votes and the budget.

There are plenty of possibilities to carry on our initiations. First, there are more parameterizations to be studied in the spirit of Niedermeier [27]. Examining the possibilities for kernelizations with respect to different parameters, as for instance was done by Betzler in [2], is an interesting approach.

Second, our FPT result for the case where the parameter is the number of votes relies on an integer linear program formulation, and uses a result by Lenstra. Since this approach does not provide running times that are suitable in practice, it would be interesting to give combinatorial algorithms that compute an optimal swap bribery. This might be particularly relevant for a scenario described by Elkind et al. [13], where bribery is not necessarily considered as an undesirable thing, like in the case of campaigning.

Also, we have focused our attention on k-approval, but the same questions could be studied for other voting systems, or for the special case of Shift Bribery which was shown to be NP-complete for several voting systems [13]. Other variants of the bribery problem, as mentioned in the introduction, could also be studied. For instance, we have only looked at *constructive* swap bribery, but the case of *destructive* swap bribery (where our aim is to achieve that a disliked candidate does *not* win) is worth further investigation as well.

Finally, we would like to point out that swap bribery is only one way to model bribery scenarios. It has several strengths: Swapping the position of two consecutive candidates in a vote is a very basic and natural operation, and the cost function that can take into account each of these swaps incorporates the small differences in the voters' preferences over the candidates. Also, this model is sufficiently general, since any vote can be transformed into any other vote by applying a whole sequence of swaps. However, one may find the model of swap bribery too detailed to correctly represent certain real life situations. Therefore, elaborating alternative models also constitutes a further challenge.

Acknowledgments. We thank Rolf Niedermeier for an inspiring initial discussion. We are also grateful for the helpful suggestions of the anonymous referees of *IPEC 2010* and *Algorithmica*.

References

1. N. Alon, R. Yuster, and U. Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.
2. N. Betzler. On problem kernels for possible winner determination under the k-approval protocol. In *MFCS 2010: Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science*, volume 6281 of *Lecture Notes in Computer Science*, pages 114–125. Springer, 2010.
3. N. Betzler and B. Dorn. Towards a dichotomy for the Possible Winner problem in elections based on scoring rules. *J. Comput. Syst. Sci.*, 76:812–836, 2010.
4. N. Betzler, S. Hemmann, and R. Niedermeier. A multivariate complexity analysis of determining possible winners given incomplete votes. In *IJCAI'09: Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 53–58, 2009.
5. N. Betzler and J. Uhlmann. Parameterized complexity of candidate control in elections and related digraph problems. *Theor. Comput. Sci.*, 410(52):5425–5442, 2009.
6. H. L. Bodlaender. Kernelization: New upper and lower bound techniques. In *IWPEC 2009: Proceedings of the 4th International Workshop on Parameterized and Exact Computation*, volume 5917 of *Lecture Notes in Computer Science*, pages 17–37, Berlin, 2009. Springer.
7. S. J. Brams and P. C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare*, volume 1, pages 173–236. Elsevier, 2002.
8. R. Christian, M. Fellows, F. Rosamond, and A. Slinko. On complexity of lobbying in multiple referenda. *Review of Economic Design*, 11(3):217–224, 2007.
9. V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *J. ACM*, 54(3):1–33, 2007.
10. R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness. II. On completeness for W[1]. *Theor. Comput. Sci.*, 141(1-2):109–131, 1995.
11. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Monographs in Computer Science. Springer, New York, 1999.
12. E. Elkind and P. Faliszewski. Approximation algorithms for campaign management. In *WINE 2010: Proceedings of the 6th Workshop on Internet and Network Economics*, volume 6484 of *Lecture Notes in Computer Science*, pages 473–482. Springer, 2010.
13. E. Elkind, P. Faliszewski, and A. M. Slinko. Swap bribery. In *SAGT 2009: Proceedings of the Second International Symposium on Algorithmic Game Theory*, volume 5814 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2009.
14. P. Faliszewski. Nonuniform bribery. In *AAMAS 2008: 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1569–1572, 2008.
15. P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. The complexity of bribery in elections. In *AAAI'06: Proceedings of the 21st National Conference on Artificial Intelligence*, pages 641–646, 2006.
16. P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Multimode control attacks on elections. *J. Artif. Intell. Res.*, 40:305–351, 2011.
17. P. Faliszewski, E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Llull and Copeland voting computationally resist bribery and constructive control. *J. Artif. Intell. Res. (JAIR)*, 35:275–341, 2009.
18. M. R. Fellows, D. Hermelin, F. A. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410:53–61, 2009.
19. J. Flum and M. Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer, New York, 2006.
20. L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
21. J. Guo and R. Niedermeier. Invitation to data reduction and problem kernelization. *SIGACT News*, 38(1):31–45, 2007.

22. E. Hemaspaandra and L. A. Hemaspaandra. Dichotomy for voting systems. *J. Comput. Syst. Sci.*, 73(1):73–83, 2007.
23. K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of the IJCAI-2005 Multidisciplinary Workshop on Advances in Preference Handling*, pages 124–129, 2005.
24. H. Lenstra. Integer programming with a fixed number of variables. *Math. of OR*, 8:538–548, 1983.
25. H. Liu, H. Feng, D. Zhu, and J. Luan. Parameterized computational complexity of control problems in voting systems. *Theor. Comput. Sci.*, 410(27-29):2746–2753, 2009.
26. R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2006.
27. R. Niedermeier. Reflections on multivariate algorithmics and problem parameterization. In *STACS 2010: Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, pages 17–32, 2010.
28. L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *AAAI'08: Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 196–201. AAAI Press, 2008.