Journal of Artificial Intelligence Research 60 (2017) 937-1002 Submitted 03/17; published 12/17

Elections with Few Voters: Candidate Control Can Be Easy

Jiehua Chen JIEHUA.CHEN2@GMAIL.COM
Ben-Gurion University of the Negev, Israel

Piotr Faliszewski FALISZEW@AGH.EDU.PL
AGH University of Science and Technology, Poland

Rolf Niedermeier ROLF.NIEDERMEIER@TU-BERLIN.DE
TU Berlin, Germany

Nimrod Talmon NIMRODTALMON77@GMAIL.COM
Weizmann Institute of Science, Israel

Abstract

We study the computational complexity of candidate control in elections with few voters, that is, we consider the parameterized complexity of candidate control in elections with respect to the number of voters as a parameter. We consider both the standard scenario of adding and deleting candidates, where one asks whether a given candidate can become a winner (or, in the destructive case, can be precluded from winning) by adding or deleting few candidates, as well as a combinatorial scenario where adding/deleting a candidate automatically means adding or deleting a whole group of candidates. Considering several fundamental voting rules, our results show that the parameterized complexity of candidate control, with the number of voters as the parameter, is much more varied than in the setting with many voters.

1. Introduction

Election control problems are concerned with affecting the result of an election by modifying the structure of the election. Such election modifications could be either introducing some new candidates or voters or removing some existing candidates or voters from the election or partitioning candidates or voters (Bartholdi III, Tovey, & Trick, 1992; Erdélyi, Nowak, & Rothe, 2009; Faliszewski, Hemaspaandra, Hemaspaandra, & Rothe, 2009; Hemaspaandra, Hemaspaandra, & Rothe, 2007; Menton, 2013; Menton & Singh, 2013; Faliszewski, Hemaspaandra, & Hemaspaandra, 2011; Faliszewski, Hemaspaandra, Hemaspaandra, & Rothe, 2011; Parkes & Xia, 2012). We focus on the computational complexity of election control by adding and deleting candidates (that is, candidate control), for the case where the election involves only a few voters. From the viewpoint of computational complexity, candidate control with few voters has not received sufficient study. We focus on very simple, practical voting rules such as Plurality, Veto, and t-Approval, but discuss several more involved rules as well. To analyze the effect of allowing only a small number of voters, we use the formal tools of parameterized complexity theory (Cygan, Fomin, Kowalik, Lokshtanov, Marx, Pilipczuk, Pilipczuk, & Saurabh, 2015; Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006).

From the viewpoint of classical complexity theory, most of the candidate control problems for most of the typically studied voting rules are NP-hard. Indeed, candidate control problems are NP-hard even for the Plurality rule; nonetheless, there are some natural examples of candidate control problems with polynomial-time algorithms. It turns out that for the case of elections with few voters, that is, for control problems parameterized by the number of voters, the computational complexity landscape of candidate control is much more varied and sometimes quite surprising. We present a high-level discussion of our results in Section 3 (to get a quick feel of the nature of the results we obtain, the reader might also wish to consult Table 1 in Section 3).

In addition to the standard candidate control problems, we also study their *combinatorial* variants, where instead of adding/deleting candidates one-by-one, we add/delete whole groups of candidates at unit cost. In this we continue our previous work, conducted by a slightly different set of authors, on combinatorial voter control (Bulteau, Chen, Faliszewski, Niedermeier, & Talmon, 2015) and combinatorial shift bribery (Bredereck, Faliszewski, Niedermeier, & Talmon, 2016b). We mention that a somewhat similar model of combinatorial control was also studied by Erdélyi, Hemaspaandra, and Hemaspaandra (2015).

The study of the computational complexity of control problems in elections was initiated by Bartholdi III et al. (1992) and was continued by many researchers; see, for example, the surveys (Faliszewski & Rothe, 2016; Faliszewski, Hemaspaandra, & Hemaspaandra, 2010) and our related work section. While many researchers have considered the case of few candidates—see, e.g., the classic work on election manipulation (Conitzer, Sandholm, & Lang, 2007) and subsequent papers regarding control (Faliszewski et al., 2009, 2011; Yang, 2014; Hemaspaandra, Lavaee, & Menton, 2016)—very little effort was invested into studying the case of few voters (perhaps the most notable example of a paper focusing on this case is Brandt, Harrenstein, Kardel, & Seedig, 2013; but Betzler, Slinko, & Uhlmann, 2013 and Betzler, Niedermeier, & Woeginger, 2011 also consider parameterization by the number of voters). One possible reason for this situation is that the case of few voters may seem somewhat less natural. After all, presidential elections, an archetype of elections, rarely involve more than a few candidates but do involve millions of voters. We argue that the case of few voters is as natural and as important to study, especially in the context of artificial intelligence and multi-agent settings and various non-political elections.[1]

1.1 Motivation

Let us now argue why we believe that elections with few voters are natural, and why (combinatorial) candidate control is an important issue in such elections. First, let us look at the following examples, which include few voters but possibly very many candidates:

1. Hiring Committee. Consider a university department which is going to hire a new faculty member. Typically, the committee consists of relatively few faculty members, but it may consider hundreds of applications for a given position. The members of the

1. We should mention that since the publication of this paper's conference version, parameterization by the number of voters has become more common. For example, Misra, Nabeel, and Singh (2015) considered winner determination parameterized by the number of voters under the Minimax Approval Voting rule, Faliszewski, Skowron, Slinko, and Talmon (2016) did the same for a class of committee scoring rules, and Bredereck, Faliszewski, Niedermeier, and Talmon (2016a) used this parameter in the study of shift bribery for committee elections. In each of these papers, parameterization by the number of voters leads to interesting fixed-parameter algorithms.

committee have to aggregate their opinions regarding the candidates and it is quite natural to assume that at some point this would be done through voting.

2. Holiday Planning. Consider a group of people who are planning to spend holidays together. The group typically would consist of no more than a dozen persons, but—technically—they have to choose from all possible options provided by the travel agents, hotels, airlines, etc. This example is particularly relevant to the case of multi-agent systems: one may foresee that in the near future we will delegate the task of finding the most satisfying holiday location to our personal software agents that will negotiate with travel agents and other travelers on our behalf.

3. Meta-search engine. Dwork, Kumar, Naor, and Sivakumar (2001) argued that one can build a web meta-search engine that queries several other search engines (the few voters) regarding a given query, aggregates their rankings of the web pages (the many candidates), and outputs the consensus ranking.

In all these examples, it is clear that prior to holding an election, the voters, or some particular individual, usually first shrinks the set of candidates. In the hiring committee case, most of the applications are removed from the considerations early in the evaluation process (based on the number of journal publications, for example). The group of people planning holidays first (implicitly) removes most of the possible holiday options and, then, removes those candidates that do not fully fit their preferences: for example, they might remove destinations which are too expensive, or holiday places by the sea when they are interested in hiking in the mountains, etc. The search engines usually disregard those web pages that appear completely irrelevant to a given query.

This natural process of modifying the candidate set, however, creates a natural opportunity for manipulating the result. A particularly crafty agent may remove those candidates that prevent his or her favorite candidate from winning. Similarly, after the initial process of thinning down the candidate set, a voter may request that some candidates are added back into consideration, possibly to help his or her favorite candidate. More importantly, it is quite realistic to assume that the voters in a small group know each other so well as to reliably predict each others' votes (this is particularly applicable to the example of the hiring committee). Thus, it is natural and relevant to study the computational complexity of candidate control parameterized by the number of voters. While control problems do not model the full game-theoretic process of adding/deleting candidates, they allow agents to compute what effects they might be able to achieve, and, if the corresponding computational problem is tractable, also how to achieve their goals.[2]

Finally, it is quite natural to consider the case where deleting (adding) a particular candidate means also deleting (adding) a number of other ones. For example, if a hiring committee removes some candidate from consideration, it might have to also remove all those with weaker publication records; if people planning holidays disregard some expensive hotel,

2. To the best of our knowledge, game-theoretic aspects of this process of adding/deleting candidates have not been studied in detail. There is, however, a related line of research regarding *strategic candidacy*, where the candidates themselves may decide to run or not (Dutta, Jackson, & Le Breton, 2001; for recent results see also, for example, Lang, Maudet, & Polukarov, 2013; Polukarov, Obraztsova, Rabinovich, Kruglyi, & Jennings, 2015; Obraztsova, Elkind, Polukarov, & Rabinovich, 2015).

they might also want to remove those that cost more. Thus, we also study *combinatorial variants* of candidate control problems which model such settings.

1.2 Main Contributions

Our research sheds light on some surprising patterns that were not (nearly as) visible in the context of classical complexity analysis of election control. The two most interesting patterns can be summarized as follows (by constructive control we mean variants of our problems where we want to ensure some candidate's victory, whereas by destructive control we mean cases where the goal is to prevent some candidate from winning):

1. In the non-combinatorial setting, destructive candidate control is easy for all our voting rules, either in the fixed-parameter tractability sense or via outright polynomial-time algorithms.
2. In the combinatorial setting, control by deleting candidates appears to be computationally harder than control by adding candidates.

We also found an interesting difference in the complexity of non-combinatorial constructive control by deleting candidates between Plurality and Veto rules (under Plurality we elect the candidate that is ranked first most often; under Veto we elect the candidate that is ranked last least often). This is especially interesting since the rules are so similar and there is no such difference for the case of adding candidates.

Our results (see Section 3 and Table 1; formal definitions are given in the next section) are of four types. For each of our problems we show that it is either:

(1) in P,

(2) in FPT (that is, is fixed-parameter tractable),

(3) is $\mathsf{W[1]}$-hard but is in XP, or

(4) is $\mathsf{Para\text{-}NP}$-hard.[3]

For each case, the parameterization is by the number of voters. Naturally, results of Type (1) are the most positive[4] because they give unconditionally efficient algorithms. Results of Type (2) are quite positive too (the exponential part of the running time of the corresponding algorithms depends only on the number of voters and not on the whole input size). The third kind is less positive (under the usual assumption that $\mathsf{FPT} \neq \mathsf{W[1]}$, $\mathsf{W[1]}$-hardness precludes the existence of FPT algorithms, but membership in XP means that there are algorithms which are polynomial-time if the number of voters is assumed to be a constant). Results of Type (4) are the most negative ones (they mean that the corresponding problems are NP-hard even for a constant number of voters; this precludes membership in XP, under the usual assumption that $\mathsf{P} \neq \mathsf{NP}$).

3. There is one exception. For t-Veto-Comb-DCAC ($t \geq 2$) we only show XP membership. Thus, from our point of view, the case of t-Veto-Comb-DCAC is still partially open.
4. We evaluate the results from the computational complexity perspective and, hence, regard computational efficiency as positive.

We emphasize that almost all of our results follow by applying proof techniques that might be useful in further research on the complexity of election problems with few voters. In particular, our W[1]-hardness results follow via reductions from the W[1]-complete Multi-Colored Clique problem and have quite a universal structure. Similarly, our Para-NP-hardness proofs follow either via reductions from the NP-complete Cubic Vertex Cover problem and use a universal trick to encode graphs into elections with eight votes, or are based on embedding Set Cover instances in our problems. Our FPT algorithms also have a fairly universal structure and are based on what we call the Signature Proof Technique. Indeed, we believe that introducing these proof techniques is an important contribution of this paper.

1.3 Related Work

The complexity study of election control was introduced by Bartholdi III et al. (1992), who were later followed by numerous researchers—mostly working within the field of artificial intelligence—including, for example, Hemaspaandra et al. (2007) (who introduced the destructive variants of the control problems), Meir, Procaccia, Rosenschein, and Zohar (2008) (who considered control in multiwinner voting rules and introduced a model that unifies the constructive and the destructive settings), Conitzer, Lang, and Xia (2009) (who considered a form of election control for elections over sets of interrelated candidates), and many others. We point the reader to the surveys of Faliszewski et al. (2010) and Rothe and Schend (2013), the book chapter of Faliszewski and Rothe (2016), and to several recent papers on the topic, including those focusing on the Schulze and Ranked Pairs rules (Parkes & Xia, 2012; Menton & Singh, 2013), and those focusing on Bucklin and Fallback rule, also including experimental studies (Erdélyi, Fellows, Rothe, & Schend, 2015a, 2015b). Briefly put, it turns out that for standard voting rules, control problems are typically NP-hard (however, it is worth noting that some of these hardness results disappear in restricted domains, as pointed out, e.g., in Brandt, Brill, Hemaspaandra, & Hemaspaandra, 2015; Faliszewski et al., 2011; Magiera & Faliszewski, 2014).

There is a growing body of research regarding the parameterized complexity of voting problems (see, for example, the survey Betzler, Bredereck, Chen, & Niedermeier, 2012), where typical parameters include the solution size (for example, the number of candidates which can be added, that is, the budget) and parameters related to the election size (usually, the number of candidates; or, as is in our case, the number of voters). When considering the solution size as the parameter, control problems typically turn out to be hard (for examples, see Betzler & Uhlmann, 2009, Liu, Feng, Zhu, & Luan, 2009, and Liu & Zhu, 2010). On the contrary, taking the number of candidates as the parameter almost always leads to FPT results (for examples, see Koutecky, Knop, & Mnich, 2017, Bredereck, Faliszewski, Niedermeier, Skowron, & Talmon, 2015, Faliszewski et al., 2011, and Hemaspaandra et al., 2016). So far, however, only Betzler and Uhlmann (2009) considered a *control* problem parameterized by the number of voters (for the Copeland rule), and Brandt et al. (2013) showed NP-hardness results of several winner determination problems even for constant numbers of voters. The parameter "number of voters" also received some attention in other voting settings (for example, Betzler, Guo, & Niedermeier, 2010 studies winner determination for Dodgson and Young rules, and Dorn & Schlotter, 2012 studies swap bribery problems in t-Approval rules), and it is currently receiving increased attention (Bredereck, Chen, Faliszewski, Nichterlein, & Niedermeier, 2016; Misra et al., 2015; Faliszewski et al., 2016; Bredereck et al., 2016a).

The study of *combinatorial* control was initiated in our recent paper (Bulteau et al., 2015), where we focused on constructive control by adding (bundles of) voters. Kellerhals, Korenwein, Zschoche, Bredereck, and Chen (2017) extended our work and investigated other types of combinatorial voter control problems. A different notion of combinatorial control was studied by Erdélyi et al. (2015), and some of us also considered combinatorial shift bribery (Bredereck et al., 2016b).

We stress that our combinatorial view of control is different from the studies of combinatorial voting domains considered, for example, by Boutilier, Brafman, Domshlak, Hoos, and Poole (2004), Xia and Conitzer (2010), and Mattei, Pini, Rossi, and Venable (2013). There, the authors consider sets of candidates of possibly exponential size and voters who express their preferences succinctly, using special formalisms (such as, for example, CP-nets). In our case, sets of candidates and preference orders are expressed directly, and the combinatorial flavor comes from considering bundles of candidates.

Finally, we mention that the original construction in our multi-colored clique proofs had a very subtle bug that was noted and fixed by Maushagen and Rothe (2016); we incorporate the fix here.

1.4 Organization

The paper is organized as follows. In the next section we provide some preliminaries regarding the standard election model, formal definitions of our control problems, and a brief review of relevant notions from parameterized complexity theory. Then, in Section 3, we discuss our results. Specifically, in Section 3.3 we discuss our proof techniques in a high-level fashion, giving the main intuitive ideas, and provide the most illustrative full proofs in the following sections: in Section 4 we discuss W[1]-hardness proofs based on the Multi-Colored Clique Proof Technique, in Section 5 we discuss Para-NP-hardness proofs based on the Cubic Vertex Cover Proof Technique, in Section 6 we discuss Para-NP-hardness proofs based on the Set-Embedding Proof Technique, in Section 7 we discuss FPT algorithms based on the Signature Proof Technique, and in Section 8 we consider the remaining problems. Finally, in Section 9, we present challenges for future research. The proofs not presented in the main text are deferred to an appendix.

2. Preliminaries

In this section, we provide some relevant notions concerning elections and voting rules, control problems, and parameterized complexity. We denote the set $\{1, 2, \dots, z\}$ by $[z]$. For a set A, we write $\mathcal{P}(A)$ to denote the family of all subsets of A.

2.1 Elections

We consider the standard, ordinal model of elections (for example, see Arrow, Sen, & Suzumura, 2002 for a general overview of elections and social choice theory, or Brandt, Conitzer, Endriss, Lang, & Procaccia, 2016 for a more computational perspective on the topic). An election $E = (C, V)$ consists of a set $C = \{c_1, \ldots, c_m\}$ of candidates and a collection $V = \{v_1, \ldots, v_n\}$ of voters. Each voter v_ℓ has a preference order (that is, a vote), often denoted by $\succ_\ell$, which ranks the candidates from the one that v_ℓ likes most to the one that v_ℓ likes least. For example, if $C = \{c_1, c_2, c_3\}$ then voter i with preference order $c_1 \succ_i c_2 \succ_i c_3$ would most like c_1, then c_2, and then c_3. Throughout the text, we use masculine forms to refer to the candidates and feminine forms to refer to the voters.

For a voter v_ℓ and two candidates, c_i and c_j, we sometimes write $v_\ell \colon c_i \succ c_j$ to indicate that v_ℓ prefers c_i to c_j. For a subset A of candidates, by writing A within a preference order description (for example, $A \succ a \succ b$, where a and b are some candidates not in A) we mean listing the members of A in some arbitrary but fixed order. By writing $\overleftarrow{A}$ we mean listing the candidates in the reverse of this arbitrary but fixed order. Given an election $E = (C, V)$, for each two candidates, $c_i, c_j \in C$, we define $N_E(c_i, c_j) \coloneqq |\{v_\ell \in V \mid v_\ell \colon c_i \succ c_j\}|$ (that is, $N_E(c_i, c_j)$ is the number of voters preferring c_i to c_j).

A voting rule $\mathcal{R}$ is a function that given an election $E = (C, V)$ outputs a set $\mathcal{R}(E) \subseteq C$ of candidates that tie as winners (we use the non-unique-winner model where the candidates in $\mathcal{R}(E)$ are equally successful). We study the following standard voting rules (in each case, the candidates who receive the highest number of points are the winners).

2.1.1 t-Approval and t-Veto

Under t-Approval, $t \geq 1$, each candidate gets a point from each voter that ranks him among the top t positions. For m candidates, t-Veto is a synonym for $(m-t)$-Approval (we often view the score of a candidate under t-Veto as the number of vetoes, that is, the number of times he is ranked among the bottom t positions). We refer to 1-Approval and 1-Veto as the Plurality rule and the Veto rule, respectively, and we jointly refer to the voting rules in this group as approval-based rules.

2.1.2 Borda Rule and Maximin Rule

Under the Borda rule, in election $E = (C, V)$ each candidate $c \in C$ receives $\sum_{d \in C \setminus \{c\}} N_E(c, d)$ points. It is also convenient to think that under the Borda rule each voter gives each candidate c as many points as there are candidates that this voter ranks below c. Under the Maximin rule, each candidate $c \in C$ receives $\min_{d \in C \setminus \{c\}} N_E(c, d)$ points. Intuitively, a Maximin winner is a candidate which loses to other candidates by the minimum margin.

2.1.3 Copeland$^\alpha$ Rule

Under the Copeland$^\alpha$ rule (where α is a rational number, $0 \leq \alpha \leq 1$), in election $E = (C, V)$ each candidate c receives $|\{d \in C \setminus \{c\} \colon N_E(c, d) > N_E(d, c)\}| + \alpha \cdot |\{d \in C \setminus \{c\} \colon N_E(c, d) = N_E(d, c)\}|$ points. Intuitively, under the Copeland$^\alpha$ rule we conduct a head-to-head contest among each pair of candidates. For a given pair of candidates, c and d, the candidate who is preferred to the other one by a majority of voters receives one point. If there is a tie, both candidates receive α points.

2.2 Control Problems

We study *candidate control* in elections, considering both constructive control (CC) and destructive control (DC), by either adding candidates (AC) or deleting candidates (DC). Thus, for example, CCAC refers to constructive control by adding candidates.

For the case of problems of control by adding candidates, we make the standard assumption that the voters have preference orders over all the candidates—both those already registered and those that can be added (that is, the unregistered candidates). Naturally, when the election is conducted we consider the preference orders to be restricted only to those candidates that either were originally registered or were added. Similarly, for the case of control by deleting candidates, to compute the result of the election we restrict the preference orders only to those candidates that were not deleted.

We consider combinatorial variants of our problems, where adding/deleting a single candidate also automatically adds/deletes a whole group of other candidates (in this we follow our earlier work on combinatorial voter control, Bulteau et al., 2015; see also Erdélyi et al., 2015). In these *combinatorial variants* (denoted with a prefix Comb), we use bundling functions κ such that for each candidate c, $\kappa(c)$ is the set of candidates which are also added if c is added (or, respectively, which are also deleted if c is deleted). For each candidate c, we require that $c \in \kappa(c)$ and call $\kappa(c)$ the *bundle* of c. For a given subset B of candidates, we write $\kappa(B)$ to denote $\bigcup_{c\in B} \kappa(c)$. Bundling functions are encoded by explicitly listing their values for all arguments.

Formally, given a voting rule $\mathcal{R}$, our problems are defined as follows (we only list the combinatorial generalizations; the non-combinatorial variants can be "derived" by using the identity function as κ).

$\mathcal{R}$-Comb-CCAC
Input: An election (C, V), a set A of unregistered candidates such that the voters from V have preference orders over $C \cup A$, a preferred candidate $p \in C$, a bundling function $\kappa\colon A \to \mathcal{P}(A)$, and a non-negative integer k.
Question: Is there a subset $A' \subseteq A$ with $|A'| \leq k$ such that $p \in \mathcal{R}(C \cup \kappa(A'), V)$?

$\mathcal{R}$-Comb-CCDC
Input: An election (C, V), a preferred candidate $p \in C$, a bundling function $\kappa\colon C \to \mathcal{P}(C)$, and a non-negative integer k.
Question: Is there a subset $C' \subseteq C$ with $|C'| \leq k$ such that $p \in \mathcal{R}(C \setminus \kappa(C'), V)$?

The destructive variants of our problems, $\mathcal{R}$-Comb-DCAC and $\mathcal{R}$-Comb-DCDC, are defined analogously, except that we replace the preferred candidate p with the despised candidate d, and we ask whether it is possible to ensure that d is *not* a winner of the election. In the DCDC case, we explicitly disallow deleting any bundle containing the despised candidate. In the standard, non-combinatorial variants of control we omit the prefix "Comb" and assume that for each candidate c we have $\kappa(c) = \{c\}$, omitting the bundling function in the respective discussions.

We believe that our model of combinatorial candidate control is fairly simple, but that it captures some of the most important features of our motivating examples. We view it as simple because in a scenario with m candidates, there are at most m corresponding bundles of candidates that can be added/deleted. While in real life one might expect many more (perhaps even exponentially many with respect to m), this is enough to associate each candidate with a "single reason" for which it could be added or deleted (for example, in the hiring committee the single reason associated with candidate c could be "having only as many journal papers as c has is not enough to be considered for the position at hand"). Then, each candidate's bundle would correspond to a set of candidates who also meet the condition associated with him. Note that each candidate could have a bundle defined through a considerably different condition and, so, the bundles could have quite an arbitrary structure. Notably, even this—quite simple—model turns out to be computationally difficult.

Indeed, one could consider an even simpler model. For example, one could associate each candidate with a label and consider actions of adding/deleting all candidates with a given label. (This is the model proposed in Erdélyi et al., 2015 in the context of voter control.) This model would not allow for a given candidate to belong to more than one bundle and, in effect, might be computationally much simpler than ours. Indeed, our techniques for showing hardness of the combinatorial control problems would not apply to it.

2.3 Parameterized Complexity

A parameterized problem is in the complexity class FPT (termed fixed-parameter tractable) if there exists an algorithm that, given an instance I of this problem (with parameter value p and instance size $|I|$; in this paper, the parameter value is always the number of voters involved), computes an answer for this instance in $f(p) \cdot |I|^{O(1)}$ time, where f is some computable function. A presumably larger complexity class is XP which consists of parameterized problems solvable in $O(|I|^{f(p)})$ time. Indeed, while the problems from both complexity classes, FPT and XP, come under the "polynomial-time solvable when the parameter p is a constant" description, it is decisive that in the case of FPT the degree of the polynomial does not depend on p, which is not the case for XP. Problems in FPT are viewed as tractable, whereas the class XP is rather considered to be at the high-level of the parameterized intractability hierarchy. Specifically, it holds that $\mathsf{FPT} \subseteq \mathsf{W[1]} \subseteq \mathsf{W[2]} \subseteq \cdots \subseteq \mathsf{XP}$.

Originally, the class W[1] was defined in terms of certain circuit-based computations. For our purposes, however, it is much easier to define W[1] through its complete problems. Specifically, W[1] contains those problems that reduce to Multi-Colored Clique (see Definition 1 below) in the parameterized sense. A parameterized reduction from a parameterized problem L to a parameterized problem L' is a function that, given an instance (I, p), computes in time $f(p) \cdot |I|^{O(1)}$ an instance (I', p') such that $p' \leq g(p)$ and $(I, p) \in L \Leftrightarrow (I', p') \in L'$, where f and g are some computable functions. We mention that in this paper all reductions can in fact be performed in polynomial time.

Definition 1. An input instance of Multi-Colored Clique consists of an undirected graph $G = (V(G), E(G))$ and a non-negative integer h such that the vertex set $V(G)$ is partitioned into h disjoint subsets, $V_1(G), \ldots, V_h(G)$, where each subset corresponds to one of the h colors (in a one-to-one manner). We ask whether there exist h vertices $v_1, \ldots, v_h$ such that for each i, $1 \leq i \leq h$, it holds that $v_i \in V_i(G)$, and each pair is connected by an edge. We call the set of these h vertices a *multi-colored clique* of order h.

We say that a problem is Para-NP-hard if there is a proof of its NP-hardness that produces instances in which the value of the parameter is upper-bounded by a constant. If a problem is Para-NP-hard for some parameter, then it cannot even belong to XP for this parameter (unless P = NP). Similarly, if a problem is W[1]-hard, it cannot be in FPT (unless FPT = W[1]). In our Para-NP-hardness proofs we will mostly rely on the following NP-hard problem (and on its restricted variant, the Cubic Vertex Cover problem; see Section 5).

Definition 2. An instance of Set Cover consists of a ground set $X = \{x_1, \ldots, x_{n'}\}$, a family $\mathcal{S} = \{S_1, \ldots, S_{m'}\}$ of subsets of X, and a non-negative integer h. We ask whether it is possible to pick at most h sets from $\mathcal{S}$ so that their union is X. Such a collection of h sets is called a *set cover* of order h.

For more details on parameterized complexity and parameterized algorithms, we point the readers to respective textbooks (Cygan et al., 2015; Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006).

3. Discussion of Our Results and Proof Techniques

In this section, we review our results, discuss some relevant patterns we identified in them, and provide a high-level overview of our proof techniques. This section can be viewed as a guide for helping the reader to better understand the implications of our results, presented in Table 1, and to get an intuitive understanding of our means of obtaining them. We begin by discussing the results regarding the approval-based voting rules, then discuss the results for the other voting rules, and finally describe our proof techniques.

3.1 Results for Approval-Based Voting Rules

Approval-based rules are perhaps the simplest and the most frequently used ones, so results regarding them are of particular interest. Further, they exhibit quite interesting behavior with respect to the complexity of candidate control parameterized by the number of voters. For the approval-based voting rules discussed here, the results regarding Plurality (and, to some extent, Veto), are by far the most important ones.

In terms of standard complexity theory, all constructive and destructive candidate control problems for Plurality and Veto are NP-complete (see Bartholdi III et al., 1992; Hemaspaandra et al., 2007 for the results regarding the Plurality rule; the results for Veto are easy to derive based on those for Plurality; for example, Elkind, Faliszewski, & Slinko, 2011 showed that Veto-CCAC is NP-complete[5]). Yet, if we consider parameterization by the number of voters, the results change quite drastically. We make the following observations:

1. The results for Plurality and Veto are no longer the same (specifically, Plurality-CCDC is in FPT whereas Veto-CCDC is W[1]-hard). This is quite surprising given both the similarities between these rules and the fact that their standard complexity-theoretic results for control are identical. Yet, we mention that some results for them were known to be different previously; for example, weighted coalitional manipulation

5. For the cases of t-Approval and t-Veto, $t \geq 2$, the results are due to Lin (2012).

(a) Approval-based voting rules

Problem	Plurality	Veto	t-Approval	t-Veto
$\mathcal{R}$-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathcal{R}$-CCDC	FPT	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathcal{R}$-DCAC	FPT	FPT	FPT	FPT
$\mathcal{R}$-DCDC	FPT	FPT	FPT	FPT
$\mathcal{R}$-Comb-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP
$\mathcal{R}$-Comb-CCDC	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1)
$\mathcal{R}$-Comb-DCAC	FPT	FPT	W[1]-h / XP	? / XP
$\mathcal{R}$-Comb-DCDC	Para-NP-h (3)	Para-NP-h (1)	Para-NP-h (2)	Para-NP-h (1)

(b) Other voting rules

Problem	Maximin	Borda	Copeland$^\alpha$
$\mathcal{R}$-CCAC	Para-NP-h (10)	Para-NP-h (10)	Para-NP-h (20) ♠
$\mathcal{R}$-CCDC	P ♣	Para-NP-h (10)	Para-NP-h (26) ♠
$\mathcal{R}$-DCAC	P ♣	P ♡	P ♢
$\mathcal{R}$-DCDC	P ♣	P ♡	P ♢
$\mathcal{R}$-Comb-CCAC	Para-NP-h (6)	Para-NP-h (2)	Para-NP-h (3) ♠
$\mathcal{R}$-Comb-CCDC	Para-NP-h (1)	Para-NP-h (1)	Para-NP-h (1) ♠
$\mathcal{R}$-Comb-DCAC	P	Para-NP-h (2)	Para-NP-h (3)
$\mathcal{R}$-Comb-DCDC	Para-NP-h (5)	Para-NP-h (2)	Para-NP-h (3)

Table 1: The complexity of constructive control (CC) and destructive control (DC) by adding candidates (AC) and by deleting candidates (DC) for various voting rules $\mathcal{R}$, parameterized by the number of voters (for t-Approval and t-Veto we mean $t \geq 2$; for Copeland$^\alpha$, we mean $0 \leq \alpha \leq 1$, where α is rational; note that the results from Betzler & Uhlmann, 2009 hold only for $\alpha \in \{0, 1\}$). Results marked with ♣ and ♢ are due to Faliszewski et al. (2009, 2011), those marked with ♡ are due to Loreggia et al. (2014), and those marked with ♠ follow from the work of Betzler and Uhlmann for $\alpha \in \{0, 1\}$ and are due to this paper for the remaining values. Cells containing statements of the form "Para-NP-h (z)" mean that the relevant problem is NP-hard even with only z voters. The question mark (?) means that the computational complexity is still open.

problem is in P for the Plurality rule and is NP-complete for the Veto rule (Hemaspaandra & Hemaspaandra, 2007).

2. For all the t-Approval and t-Veto rules (including Plurality and Veto), the destructive non-combinatorial candidate control problems are fixed parameter tractable. The constructive variants of these problems—with the exception of Plurality-CCDC—are W[1]-hard.

3. For the combinatorial setting there is a sharp difference between control by adding candidates and control by deleting candidates. Specifically, for both the Plurality rule and the Veto rule, Comb-DCAC is fixed-parameter tractable and Comb-CCAC is W[1]-hard, whereas Comb-CCDC and Comb-DCDC are Para-NP-hard. For t-Approval and

t-Veto with $t \geq 2$, the patterns are simpler. All the adding-candidates cases are W[1]-hard (with one open case), and all the deleting-candidates cases are Para-NP-hard.

We conclude by noting that in each of the W[1]-hard cases discussed here we also provide an XP algorithm. This means that, under the assumption $\mathsf{P} \neq \mathsf{NP}$, these cases cannot be strengthened to Para-NP-hardness results and, thus, in some sense our results are tight. It is quite interesting that in the non-combinatorial setting the demarcation line between fixed-parameter tractable and W[1]-hard problems goes along the constructive-vs-destructive axis, whereas for the combinatorial setting the line between W[1]-hard (or, in-FPT for Plurality and Veto) and Para-NP-hard problems goes along the adding-vs-deleting-candidates axis.

3.2 Results for Maximin, Borda, and Copeland

The results for Maximin, Borda, and Copeland$^\alpha$ rules are quite different from those for t-Approval and t-Veto. Here, instead of FPT and W[1]-hardness results we find polynomial-time algorithms and Para-NP-hardness results. Specifically, it has already been reported in the literature that there are polynomial-time algorithms for destructive candidate control under the Borda rule (Loreggia et al., 2014), the Copeland$^\alpha$ rule (Faliszewski et al., 2009), and the Maximin rule (Faliszewski et al., 2011). For constructive candidate control, Para-NP-hardness was already known for Copeland^0 and Copeland^1 (Betzler & Uhlmann, 2009), while in this paper we establish the same Para-NP-hardness for the remaining values of α (that is, for $0 < \alpha < 1$), for the Borda rule, and for the Maximin rule (in the latter case, only for CCAC; CCDC is known to be in P, as shown in Faliszewski et al., 2011).

For the combinatorial setting, almost all of our problems are Para-NP-hard. The only exception is Maximin-Comb-DCAC, which can be solved in polynomial time using an algorithm that, in essence, is identical to the one for the non-combinatorial setting. Our proofs of the Para-NP-hardness results mostly rely on a set-embedding technique (see the next section), which more-or-less directly embeds instances of Set Cover in our control problems. Due to the generality of this approach, we also prove that for every voting rule $\mathcal{R}$ that satisfies the unanimity principle (that is, for every voting rule that chooses as the unique winner the candidate that is ranked first by all the voters, if such a candidate exists), $\mathcal{R}$-Comb-CCDC is Para-NP-hard.

Summarizing the discussion above, for our more involved voting rules, the high-level intuition is that constructive candidate control is Para-NP-hard even in the non-combinatorial setting, whereas destructive candidate control is polynomial-time solvable in the non-combinatorial settings, but becomes Para-NP-hard in the combinatorial ones. The only exception from this rule is Maximin for the destructive control by adding candidates (Maximin-(Comb)-DCAC).

3.3 Proof Techniques

We believe that one of the most important contributions of this paper comes from identifying four very general proof techniques for establishing our results. Indeed, we believe that these techniques might be useful in studying the complexity of other election problems (especially, parameterized by the number of voters) and below we provide their high-level, intuitive descriptions.

3.3.1 Overview of the Multi-Colored Clique Proof Technique

This is a technique used for establishing W[1]-hardness results. The main idea is to provide a reduction from the Multi-Colored Clique (MCC) problem parameterized by the order of the desired clique. This problem, which is a variant of the standard Clique problem, is more naturally suited for parameterized complexity analysis than Clique itself (Fellows, Hermelin, Rosamond, & Vialette, 2009). Specifically, each vertex is associated with one color out of h colors overall and we seek a clique of order h containing one vertex of each color. The high-level description of our technique is as follows. We provide a reduction that, given an MCC-instance, introduces a candidate for each vertex and two candidates for each edge. We have to ensure that the only successful control actions add exactly the candidates (delete all but exactly the candidates) which correspond to a multi-colored clique (we mean both the candidates corresponding to the vertices of the clique and the candidates corresponding to the edges between them). We enforce this constraint using pairs of carefully crafted votes such that if we have two vertices but not an edge between them, then some candidate receives one more point than it should have for our preferred candidate to win. Note that the colors help us to upper-bound the number of voters needed for the construction (specifically, the number of voters created in the construction depends only on the parameter, therefore giving rise to the parameterized hardness). Formal proofs using this technique are given in Section 4.

3.3.2 Overview of the Cubic Vertex Cover Proof Technique

This is a technique used for establishing Para-NP-hardness results for non-combinatorial constructive candidate control problems. The main idea is to give a reduction from the Cubic Vertex Cover (CVC) problem. This problem is a variant of the standard Vertex Cover problem where the input graph is guaranteed to be cubic (that is, each vertex is incident to exactly three edges). The crucial observation used in this technique is that the edges of a cubic graph can be partitioned into four disjoint matchings. This fact allows us to encode all information regarding the graph in a constant number of votes, in a way that ensures that the actions of adding/deleting candidates correspond to covering edges. Formal proofs using this technique are given in Section 5.

3.3.3 Overview of the Set-Embedding Proof Technique

This is a fairly simple technique for showing Para-NP-hardness results for combinatorial control by adding/deleting candidates. The idea is to reduce from the Set Cover problem using the bundling function to encode the given sets. Due to the power of bundling, a constant number of voters suffices for the reduction. Formal proofs using this technique are given in Section 6.

3.3.4 Overview of the Signature Proof Technique

This is a group of two similar techniques for showing FPT results for the case of destructive control under t-Approval/t-Veto rules. The first technique in the group works for problems of destructive control by adding candidates. The main idea is to group candidates by finding some equivalences between them in the sense that it does not make a difference which candidate from the set we add. In other words, often it is possible to limit the number of candidates that one has to consider by identifying their most crucial properties (such as the subsets of voters where the candidates are ranked ahead of some given candidate); we refer to these properties as signatures. Building upon this idea, and by upper-bounding the number of such groups with function solely depending on the number of voters, we achieve fixed-parameter tractability results. The second technique is of similar nature and applies to problems of destructive control by deleting candidates. Formal proofs using this technique are given in Section 7.

Almost all of the proofs in this paper follow by applying one of the above four techniques. The few remaining ones follow by direct arguments and are given in Section 8. The following sections are ordered by the proof technique employed and present the most notable results. Proofs omitted here are presented in the appendix.

4. Multi-Colored Clique Proof Technique

We start our technical discussion by describing the Multi-Colored Clique proof technique. We apply it to showing W[1]-hardness of candidate control problems for t-Approval/t-Veto rules. Indeed, all W[1]-hardness proofs in this paper are based on this technique. Specifically, we prove the following statements (all results are for the parameterization by the number of voters):

1. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathcal{R}$-CCAC (and therefore also $\mathcal{R}$-Comb-CCAC) is W[1]-hard.
2. For each fixed integer $t \geq 2$ and for each voting rule $\mathcal{R} \in \{\text{Veto}, t\text{-Approval}, t\text{-Veto}\}$, $\mathcal{R}$-CCDC is W[1]-hard.
3. For each fixed integer $t \geq 2$, t-Veto-Comb-DCAC is W[1]-hard.

All these results follow by reductions from Multi-Colored Clique (hence the name of the technique) and are quite similar in spirit. Thus, we start by providing some common notation and observations for all of them.

Let $I = (G, h)$ be a Multi-Colored Clique instance with graph G and non-negative integer h (recall Definition 1). The vertices of G are partitioned into h sets, $V_1(G), \dots, V_h(G)$, each containing all vertices colored with a given color. Without loss of generality, we assume that $h > 2$ and each $V_i(G)$ contains the same number of vertices, denoted by n', and we rename the vertices so that for each color i, $1 \leq i \leq h$, we have $V_i(G) = \{v_1^{(i)}, \dots, v_{n'}^{(i)}\}$. The task is to decide whether there is a clique of order h where each vertex comes from a different set $V_i(G)$. Moreover, and without loss of generality, we assume that each edge in G connects vertices with different colors, that the input graph contains at least two vertices, and that this graph is connected.

In our reductions, given an instance $I = (G, h)$, we build elections with the following candidates related to the graph G (in addition to the candidates specific to each particular reduction). For each vertex $v \in V(G)$, we introduce a candidate denoted by the same symbol. For each edge $e = \{u, v\}$, we introduce two candidates (u, v) and (v, u) (while our original graph is undirected, for our construction we treat each undirected edge as two directed ones, one in each direction).

In the description of our preference orders, we will use the following orders over subsets of candidates:

1. For each color i, when we write $V_i(G)$ in a preference order, we mean the order

$$V_i(G)\colon\ v_1^{(i)} \succ v_2^{(i)} \succ \cdots \succ v_{n'}^{(i)}.$$

2. For each color i, each vertex $v_\ell^{(i)} \in V_i(G)$, and each color j with $j \neq i$, we write $L(v_\ell^{(i)}, j)$ to denote the order obtained from

$$L(v_\ell^{(i)}, j)\colon\ (v_\ell^{(i)}, v_1^{(j)}) \succ \cdots \succ (v_\ell^{(i)}, v_{n'}^{(j)})$$

by removing those candidates $(v_\ell^{(i)}, v_h^{(j)})$ for which there is no edge $\{v_\ell^{(i)}, v_h^{(j)}\}$ in G. Intuitively, $L(v_\ell^{(i)}, j)$ lists all edge candidates for edges which have endpoint $v_\ell^{(i)}$ and go to vertices of color j (the particular order of these edges in $L(v_\ell^{(i)}, j)$ is irrelevant for our constructions).

3. The following two preference orders are crucial for the MULTI-COLORED CLIQUE technique. For each two distinct colors i and j ($1 \le i, j \le h$, $i \neq j$) we define $R(i,j)$ and $R'(i,j)$ as follows:

$$R(i,j)\colon v_1^{(i)} \succ L(v_1^{(i)}, j) \succ v_2^{(i)} \succ L(v_2^{(i)}, j) \succ \cdots \succ v_{n'}^{(i)} \succ L(v_{n'}^{(i)}, j),$$

$$R'(i,j)\colon L(v_1^{(i)}, j) \succ v_1^{(i)} \succ L(v_2^{(i)}, j) \succ v_2^{(i)} \succ \cdots \succ L(v_{n'}^{(i)}, j) \succ v_{n'}^{(i)}.$$

The idea behind $R(i,j)$ and $R'(i,j)$ is as follows. Consider a setting where u is a vertex of color i and v is a vertex of color j (that is, $u \in V_i(G)$ and $v \in V_j(G)$). Note that $R(i,j)$ and $R'(i,j)$ contain all candidates from $V_i(G)$ and all edge candidates (u,v) where $u \in V_i(G)$ and $v \in V_j(G)$. If we restrict these two preference orders to candidates u and (u,v), then they will become $u \succ (u,v)$ and $(u,v) \succ u$. That is, in this case they are reverses of each other. However, if we restrict them to u and some candidate (u',v') with $u \neq u'$, then either they will be both $u \succ (u',v')$ or they will be both $(u',v') \succ u$. Using this effect is at the heart of our constructions.

4. For each pair $\{i,j\}$ of colors ($1 \le i < j \le h$, $i \neq j$), let $E(\{i,j\}) = (e_1^{\{i,j\}}, \ldots, e_t^{\{i,j\}})$ denote some fixed sequence of all the edges between the vertices from $V_i(G)$ and $V_j(G)$ (the specific order of the edges in this sequence is irrelevant and we pick an easily computable one). Given an edge $e_\ell^{\{i,j\}} = \{u,v\}$ from this sequence, with $u \in V_i(G)$ and $v \in V_j(G)$, by $e_\ell^{(i,j)}$ we mean the candidate (u,v) and by $e_\ell^{(j,i)}$ we mean the candidate (v,u). We define the following two preference orders:

$$E(i,j)\colon\ e_1^{(i,j)} \succ e_1^{(j,i)} \succ e_2^{(i,j)} \succ e_2^{(j,i)} \succ \cdots \succ e_t^{(i,j)} \succ e_t^{(j,i)},$$

$$E(j,i)\colon\ e_1^{(j,i)} \succ e_1^{(i,j)} \succ e_2^{(j,i)} \succ e_2^{(i,j)} \succ \cdots \succ e_t^{(j,i)} \succ e_t^{(i,j)}.$$

Both $E(i,j)$ and $E(j,i)$ list all the candidates that correspond to the edges between vertices of colors i and j and the difference is that for each edge of the form $\{v_\ell^{(i)}, v_h^{(j)}\}$, in $E(i,j)$ we have $(v_\ell^{(i)}, v_h^{(j)}) \succ (v_h^{(j)}, v_\ell^{(i)})$ and in $E(j,i)$ we have $(v_h^{(j)}, v_\ell^{(i)}) \succ (v_\ell^{(i)}, v_h^{(j)})$. This construction of $E(i,j)$ and $E(j,i)$ is due to Maushagen and Rothe (2016). We thank them for pointing out the flaw in our original construction and repairing it (we explain why the current construction works within the proofs).

With the above setup, we are ready to prove the results of this section. Here we give the most interesting examples of proofs; the remaining ones are in Appendix A.

Theorem 4.1. *Plurality-*CCAC*, parameterized by the number of voters, is* W[1]*-hard.*

Proof. Let $I = (G, h)$ be our input instance of Multi-Colored Clique with graph G and non-negative integer h. Let the notation be the same as described above the theorem statement. We form an instance I' of Plurality-CCAC as follows. Let the registered candidate set C consist of two candidates, p and d, and let the set A of unregistered candidates contain all vertex candidates and all edge candidates for G. Let p be the preferred candidate. We construct the election such that the current winner is d. We introduce the following groups of voters (when we write "$\cdots$" in a preference order, we mean listing all the remaining candidates in some arbitrary order; we illustrate the construction in Example 1 after the proof):

1. For each color i ($1 \le i \le h$), we have one voter with preference order of the form
$$V_i(G) \succ d \succ \cdots \succ p.$$

2. For each two distinct colors i and j ($1 \le i, j \le h$, $i \ne j$), we have $h-1$ voters with preference order of the form
$$E(i,j) \succ d \succ \cdots \succ p.$$

3. For each two distinct colors i and j ($1 \le i, j \le h$, $i \ne j$) we have two voters, one with preference order of the form
$$R(i,j) \succ d \succ \cdots \succ p,$$
and one with preference order of the form
$$R'(i,j) \succ d \succ \cdots \succ p.$$

4. We have h voters with preference order of the form
$$d \succ \cdots \succ p,$$
and h voters with preference order of the form
$$p \succ \cdots \succ d.$$

Note that the total number of voters is $3h + 2(h+1) \cdot \binom{h}{2}$ and that the current winner is d with the score of $(2h + 2(h+1) \cdot \binom{h}{2})$ points. We set the budget $k := h + 2\binom{h}{2} = h^2$. This completes the construction (which is a parameterized reduction).

We now claim that it is possible to ensure that p becomes a winner by adding at most k candidates if and only if I is a "yes"-instance.

First, assume that I is a "yes"-instance of Multi-Colored Clique and let Q be a size-h subset of vertices that forms a multi-colored clique in I. If we add to our election the h vertex candidates from Q and all edge candidates that correspond to edges between the candidates from Q, then, in the resulting election, each candidate (including p and d) will have h points (for example, each of the added vertex candidates will receive one point from the first group of voters and $h-1$ points from the third group of voters). Thus, everyone will win.

Now, assume that it is possible to ensure p's victory by adding at most k candidates. Let A' be a subset of candidates with $|A'| \le k = h + 2\binom{h}{2}$ such that adding the candidates from A' to the election ensures that p is a winner. We claim that A' has exactly k candidates: Irrespective of the contents of the set A', in the resulting election p will have h points. Thus, it follows that d must lose all the points from the first three groups of voters. Observe that there is a total of $h + 2(h+1) \cdot \binom{h}{2} = h^3 = h \cdot k$ voters in the first three voter groups and they all give points to the newly added candidates. Since each added candidate can have at most h points, it follows that $|A'| = k$ and moreover, that each added candidate receives exactly h points. We also have the following facts:

(i) For each color i, $1 \le i \le h$, A' contains *exactly* one vertex candidate from $V_i(G)$. First, A' contains at least one vertex candidate from $V_i(G)$ since otherwise d would not lose all the points from the first group of voters. If A' would contain two vertex candidates u and u' with the same color, then since they could only obtain points from the first and the third group of voters and since the preference orders of both groups restricted to u and u' are the same, one of u and u' would have at least one point less than the other—a contradiction to the fact that every newly added candidate should receive exactly h points.

(ii) For each added vertex candidate $u \in A'$ with color i and each other color j with $j \in \{1, 2, \dots, h\} \setminus \{i\}$, A' contains *exactly* one edge candidate (u, v) with $v \in V_j(G)$. First, by the cardinality of A' and by the reasoning in point (i), A' contains at most one edge candidate for each (i, j). Now, suppose for the sake of contradiction that A' contains a vertex candidate u with color i and there is another color j with $j \in \{1, 2, \dots, h\} \setminus \{i\}$, such that $A' \cap (V_i(G) \times V_j(G)) = \emptyset$. This implies that u will obtain one point from the first group of voters and two points from the voters corresponding to the ordered pair (i, j) in the third group of voters. Then, since u should receive exactly h points, there must be a third color $j' \in \{1, 2, \dots, h\} \setminus \{i, j\}$ for which A' contains an edge candidate (u', v') such that $u' \in V_i(G)$ and $v' \in V_{j'}(G)$ and such that (u', v') receives two points from the voter corresponding to the ordered pair (i, j'). However, by the preference order of the second group voters, (u', v') will obtain more than h points—a contradiction.

Now, to show that G has a multi-colored clique of order h, it suffices to show that for each two distinct vertex candidates $u, v \in A'$ we have that $\{u, v\} \in E$. Suppose for the sake

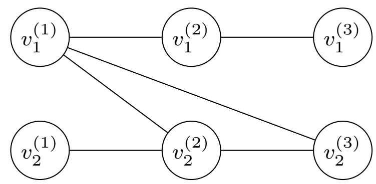

Figure 1: The input graph considered in our example for Theorem 4.1.

of contradiction that A' contains two vertex candidates $u, v \in A'$ but $\{u, v\} \notin E$. Let i and j be the colors of u and v, respectively. First of all, by the second point in the above reasoning, A' contains two edge candidates (u, v') and (v, u') with $u' \in V_i(G) \setminus \{u\}$ and $v' \in V_j(G) \setminus \{v\}$. Now observe that the preference orders of the second group restricted to (u, v') and (v, u') are the same (note that $u' \neq u$ and $v' \neq v$). This implies that one of (u, v') and (v, u') will receive all $2(h-1)$ points from the corresponding voters—a contradiction to each added candidate receives exactly h points. ☐

Example 1. *We provide an example for the reduction described in the proof of Theorem 4.1. The input graph is depicted in Figure 1, we take $h \coloneqq 3$, and the election constructed by the reduction is given in Table 2. In this table, the registered candidates are typeset in bold, the unregistered ones are typeset normally, and the added candidates are marked with gray background. These added candidates, who correspond to the multi-colored clique $\{v_1^{(1)}, v_2^{(2)}, v_2^{(3)}\}$ are:*

$$
\begin{aligned}
&v_1^{(1)}, v_2^{(2)}, v_2^{(3)},\\
&(v_1^{(1)}, v_2^{(2)}), (v_1^{(1)}, v_2^{(3)}), (v_2^{(2)}, v_2^{(3)}),\\
&(v_2^{(2)}, v_1^{(1)}), (v_2^{(3)}, v_1^{(1)}), (v_2^{(3)}, v_2^{(2)}).
\end{aligned}
$$

We see that p, with three points, is among the winners of this election (as are all the other candidates).

We now consider the Veto-CCAC case, which, despite being a simple modification of the last proof, is quite intriguing.

Theorem 4.2. *Veto-*CCAC*, parameterized by the number of voters, is* W[1]*-hard.*

Proof. One can use the same construction (and proof) as for the Plurality-CCAC case (Theorem 4.1), but with the following modifications (note that the order is important, that is, we perform the second modification only after we have performed the first modification):

1. swap the occurrences of p and d in every vote, and
2. reverse each vote.

In effect, prior to adding candidates, p is vetoed by all but h voters and d is vetoed by exactly h voters. If we add vertex candidates and edge candidates that correspond to a multi-colored clique, then every candidate in the election is vetoed by exactly h voters and all the candidates are winners.

Group, Color(s)	Preference order	
1, (1)	$v_1^{(1)} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$	
1, (2)	$v_1^{(2)} \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$	
1, (3)	$v_1^{(3)} \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$	
2, (1, 2)	$(v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(2)}, v_1^{(1)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(1)}, v_2^{(2)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
2, (2, 1)	$(v_1^{(2)}, v_1^{(1)}) \succ (v_1^{(1)}, v_1^{(2)}) \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
2, (1, 3)	$(v_1^{(1)}, v_2^{(3)}) \succ (v_2^{(3)}, v_1^{(1)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
2, (3, 1)	$(v_2^{(3)}, v_1^{(1)}) \succ (v_1^{(1)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
2, (2, 3)	$(v_1^{(2)}, v_1^{(3)}) \succ (v_1^{(3)}, v_1^{(2)}) \succ (v_2^{(2)}, v_2^{(3)}) \succ (v_2^{(3)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
2, (3, 2)	$(v_1^{(3)}, v_1^{(2)}) \succ (v_1^{(2)}, v_1^{(3)}) \succ (v_2^{(3)}, v_2^{(2)}) \succ (v_2^{(2)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$	(2 copies)
3, (1, 2)	$v_1^{(1)} \succ (v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ v_2^{(1)} \succ (v_2^{(1)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$	
3, (1, 2)	$(v_1^{(1)}, v_1^{(2)}) \succ (v_1^{(1)}, v_2^{(2)}) \succ v_1^{(1)} \succ (v_2^{(1)}, v_2^{(2)}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$	
3, (2, 1)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(1)}) \succ v_2^{(2)} \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ \mathbf{d} \succ \cdots$	
3, (2, 1)	$(v_1^{(2)}, v_1^{(1)}) \succ v_1^{(2)} \succ (v_2^{(2)}, v_1^{(1)}) \succ (v_2^{(2)}, v_2^{(1)}) \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$	
3, (1, 3)	$v_1^{(1)} \succ (v_1^{(1)}, v_2^{(3)}) \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$	
3, (1, 3)	$(v_1^{(1)}, v_2^{(3)}) \succ v_1^{(1)} \succ v_2^{(1)} \succ \mathbf{d} \succ \cdots$	
3, (3, 1)	$v_1^{(3)} \succ v_2^{(3)} \succ (v_2^{(3)}, v_1^{(1)}) \succ \mathbf{d} \succ \cdots$	
3, (3, 1)	$v_1^{(3)} \succ (v_2^{(3)}, v_1^{(1)}) \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$	
3, (2, 3)	$v_1^{(2)} \succ (v_1^{(2)}, v_1^{(3)}) \succ v_2^{(2)} \succ (v_2^{(2)}, v_2^{(3)}) \succ \mathbf{d} \succ \cdots$	
3, (2, 3)	$(v_1^{(2)}, v_1^{(3)}) \succ v_1^{(2)} \succ (v_2^{(2)}, v_2^{(3)}) \succ v_2^{(2)} \succ \mathbf{d} \succ \cdots$	
3, (3, 2)	$v_1^{(3)} \succ (v_1^{(3)}, v_1^{(2)}) \succ v_2^{(3)} \succ (v_2^{(3)}, v_2^{(2)}) \succ \mathbf{d} \succ \cdots$	
3, (3, 2)	$(v_1^{(3)}, v_1^{(2)}) \succ v_1^{(3)} \succ (v_2^{(3)}, v_2^{(2)}) \succ v_2^{(3)} \succ \mathbf{d} \succ \cdots$	
4	$\mathbf{d} \succ \cdots$	(3 copies)
4	$\mathbf{p} \succ \cdots$	(3 copies)

Table 2: The election constructed in the proof of Theorem 4.1 for our example, that is, for the input graph depicted in Figure 1. The registered candidates (typeset in bold) are d and p. The added candidates corresponding to picking the multi-colored clique $\{v_1^{(1)}, v_2^{(2)}, v_2^{(3)}\}$ are typeset with gray background.

For the reverse direction, analogously as in the Plurality case (Theorem 4.1), we note that we have to add exactly one vertex candidate of each color and exactly one edge candidate for each (ordered) pair of colors (otherwise p would receive more than h vetoes, or one of the added candidates would receive fewer than h vetoes). To argue that for each pair of vertex candidates u and v that we add, we also have to add edge candidate (u, v), we use the same reasoning as in the Plurality case, but pointing out that if some candidate receives two vetoes from the third group of voters, then some other one receives, altogether, fewer than h vetoes and p is not a winner. ☐

To see why this result is intriguing, let us consider the following voting rule, that we call *TrueVeto*. Under TrueVeto, a candidate c is a winner if none of the voters ranks c last. TrueVeto-CCAC is indeed NP-complete (this can be proved by a reduction from Set Cover, for example), but it is also in FPT (when parameterized by the number of voters; an algorithm similar to that for Plurality-DCAC, based on our signature technique, works; see Section 3.3). If a Veto election contained more candidates than voters, then at least one candidate would never be vetoed and, in effect, the election would be held according to the TrueVeto rule. This means that in the proof which shows that Veto-CCAC is W[1]-hard, the election has fewer candidates than voters, even after adding the candidates (and keep in mind that the number of voters is the parameter!). Thus, the hardness of the problem lays in picking a few spoiler candidates to add from a large group of them. If we were adding more candidates than we had voters, then the problem would be in FPT.

Now, we move on to the deleting candidates case. We will give a detailed proof for Veto-CCDC (on the one hand, Plurality-CCDC is in FPT, and, on the other hand, it is instructive to see a detailed proof for the case of Veto). The proof still follows the general ideas of the Multi-Colored Clique technique, but since we delete candidates, we have to adapt the approach.

Theorem 4.3. *Veto-*CCDC*, parameterized by the number of voters, is* W[1]*-hard.*

Proof. We provide a parameterized reduction from the Multi-Colored Clique problem. Let $I = (G, h)$ be our input instance with graph G and non-negative integer h, and let the notation be as described in the introduction to the section. We form an instance I' of Veto-CCDC as follows. Let the registered candidate set C consist of all vertex candidates plus all edge candidates for G, plus the preferred candidate p. We construct the following groups of voters (set $H = 2\binom{h}{2} = h \cdot (h-1)$):

1. For each color i, $1 \le i \le h$, we introduce $2H - (h-1)$ voters with preference order of the form
$$\cdots \succ p \succ V_i(G).$$

2. For each two distinct colors i and j ($1 \le i, j \le h$, $i \neq j$) we introduce $2H - 1$ voters with preference order of the form
$$\cdots \succ p \succ E(i, j).$$

3. For each two distinct colors i and j ($1 \le i, j \le h$, $i \neq j$) we introduce two voters, one with preference order of the form
$$\cdots \succ p \succ R(i, j),$$

and one with preference order of the form

$$\cdots \succ p \succ R'(i,j).$$

4. We introduce $2H$ voters with preference order of the form $\cdots \succ p$.

We set the number k of candidates that can be deleted to $|V(G)| - h + 2|E(G)| - H$ (with the intention that one should delete all the candidates except for p and those corresponding to the vertices and edges of the multi-colored clique of order h). This completes the construction. Note that the total number of voters is

$$(2H - (h-1)) \cdot h + (2H-1) \cdot H + H \cdot 2 + 2H \cdot 1 = 2H \cdot (H + h + 1).$$

Since the input graph is connected and contains at least two vertices (which means that the election has more than $H+h+1$ candidates), there is at least one candidate, either a vertex candidate or an edge candidate, which has fewer than $2H$ vetoes. Thus, p is currently not a winner.

We claim that p can become a winner by deleting at most k candidates if and only if I is a "yes"-instance. First, if G contains an order-h multi-colored clique and Q is the set of h vertices that form such a clique, then we can ensure that p is a winner. It suffices to delete all candidates from $V(G) \setminus Q$ and all edge candidates except the ones of the form (u,v), where both u and v belong to Q. In effect, each remaining candidate will have $2H$ vetoes and all the candidates will tie for victory. To see this, note that after deleting the candidates, p still receives $2H$ vetoes from the last group of voters. Now, for each color i, $1 \leq i \leq h$, consider the remaining vertex candidate of color i (call this vertex $v^{(i)}$). This candidate receives $2H - (h-1)$ vetoes from the first group of voters. Further, there are exactly $h-1$ voters in the third group that each give one veto to $v^{(i)}$ (these are the voters that correspond to the edges that connect $v^{(i)}$ with the other vertices of the clique). No other voter vetoes $v^{(i)}$. Now, for each pair of colors i and j, $1 \leq i,j \leq h$, $i \neq j$, consider the two edge candidates, call them (u,v) and (v,u), whose corresponding edges are incident to the vertices of color i (candidate u) and color j (candidate v). Both (u,v) and (v,u) still get $2H-1$ vetoes from the second group of voters. Each of them receives one veto from the third group of voters (for the case of (u,v), this veto comes from the first voter corresponding to the color choice (i,j), and in the case of v, this veto comes from the first voter corresponding to the color choice (j,i)).

Now we come to the reverse direction. Assume that it is possible to ensure p's victory by deleting at most k candidates. Prior to deleting any candidates, p has $2H$ vetoes and, of course, deleting candidates cannot decrease this number. Thus, we have to ensure that each non-deleted candidate has at least $2H$ vetoes.

Consider two colors i and j, $1 \leq i$, $j \leq h$, $i \neq j$. Each edge candidate (u,v) (where the corresponding vertex u has color i and the corresponding vertex v has color j) appears below p in $2H-1$ votes from the second group of voters and in two votes from the third one. If we keep two edge candidates, say (u',v') and (u'',v'') (where $u',u'' \in V_i(G)$ and $v',v'' \in V_j(G)$), then they are both ranked below p in the same $2H-1$ votes from the second group and in the same two votes from the third one. If neither (u',v') nor (u'',v'') is deleted, then one of them will receive fewer than $2H$ vetoes. This means that for each pair of colors i and j, we have to delete all except possibly one edge candidate of the form (u,v), where $u \in V_i(G)$ and $v \in V_j(G)$.

Similarly, for each color i, $1 \le i \le h$, each vertex candidate from $V_i(G)$ appears below p in $2H-(h-1)$ votes from the first group of voters and in $2(h-1)$ votes from the third group. Each two candidates of the same color are ranked below p in the same votes in the first group. Thus, if two vertex candidates of the same color were left in the election (after deleting candidates), then at least one of them would have fewer than $2H$ vetoes.

In consequence, and since we can delete at most $k = |V(G)|-h+2|E(G)|-H$ candidates, which means that at least $h+H$ candidates except p must remain in the final election, if p is to become a winner, then after deleting the candidates the election must contain exactly one vertex candidate of each color, and exactly one edge candidate for each ordered pair of colors.

Assume that p is among the winners after deleting candidates and consider two remaining vertex candidates u and v, $u \in V_i(G)$ and $v \in V_j(G)$ ($i \neq j$); they must exist by the previous observations. We claim that edge candidates (u,v) and (v,u) also must remain. Due to symmetry, it suffices to consider (u,v). Careful inspection of voters in the third group shows that if (u,v) is not among the remaining candidates, then (using the observation regarding orders $R(i,j)$ and $R'(i,j)$) we have that the two voters from the third group that correspond to the color pair (i,j) either both rank u last or both rank the same edge candidate last. In either case, a simple counting argument shows that either u has fewer than $2H$ vetoes or the edge candidate corresponding to the ordered color pair (i,j) has fewer than $2H$ vetoes. In either case, p is not a winner. This shows that the remaining candidates correspond to an order-h multi-colored clique. ☐

Our final example of the application of the multi-colored clique technique is for t-Approval-Comb-DCAC for $t \ge 2$. We use an approach very similar to the one used in the preceding proofs, but since we are in the combinatorial setting, we use the bundling function to ensure consistency between the added edge candidates and the added vertex candidates. This is crucial since t-Approval-DCAC parameterized by the number of voters is fixed-parameter tractable (Theorem 7.4).

Theorem 4.4. *For each fixed integer $t \ge 2$, t-Approval-*Comb-DCAC*, parameterized by the number of voters, is* W[1]*-hard.*

Proof. Given an instance (G,h) for the Multi-Colored Clique problem, we construct an instance of t-Approval-Comb-DCAC. For the combinatorial setting it is more natural to create only one candidate for each edge, and not two "directed" ones. We let the set of registered candidates be $C = \{p,d\} \cup D$, where D is the following set of dummy candidates:

$$\begin{aligned} D = \{&a_z^{\{i,j\}} \mid i,j \in [h], i \neq j, z \in [t-1]\} \\ \cup\ \{&b_z^{(i)} \mid i \in [h], z \in [t-1]\} \\ \cup\ \{&c_z^{(i)} \mid i \in [h-1], z \in [t-1]\}. \end{aligned}$$

Candidate d is the despised one whose victory we want to preclude. We let the set of the unregistered candidates be

$$A = V(G) \cup E(G).$$

That is, A contains all vertex candidates and all edge candidates. We set the bundling function κ so that for each edge candidate e whose corresponding edge is incident to u and v, we have $\kappa(e) = \{e, u, v\}$, and for each vertex candidate v we have $\kappa(v) = \{v\}$. We introduce the following voters:

1. For each pair $\{i, j\} \subseteq [h]$ of colors ($1 \leq i < j \leq h$, $i \neq j$), we have one voter with the following preference order (recall that we write $E(\{i, j\})$ to mean an arbitrarily chosen order over the edge candidates that link vertices of color i with those of color j; the first occurrence of "$\cdots$" regards the candidates in $\{a_z^{\{i,j\}} \mid z \in [t-1]\}$ only):

$$E(\{i, j\}) \succ a_1^{\{i,j\}} \succ \cdots \succ a_{t-1}^{\{i,j\}} \succ d \succ \cdots .$$

   Note that in the initial election, d gets a point from this voter, but it is sufficient (and we will make sure that it is also necessary) to add one candidate from $E(\{i, j\})$ to prevent d from getting this point.

2. For each color i, $1 \leq i \leq h$, we have a voter with the following preference order (recall that $V_i(G)$ is a sequence consisting of all vertex candidates that correspond to the vertices of the same color i; the first occurrence of "$\cdots$" regards the candidates in $\{b_z^{(i)} \mid z \in [t-2]\}$ only):

$$V_i(G) \succ b_1^{(i)} \succ \cdots \succ b_{t-2}^{(i)} \succ p \succ b_{t-1}^{(i)} \succ \cdots .$$

   Note that in the initial election p gets a point from this voter, but if more than one candidate from $V_i(G)$ is added, then p does not gain this point.

3. For each number $i \in [h-1]$, we have a voter with the following preference order (the first occurrence of "$\cdots$" regards the candidates in $\{c_z^{(i)} \mid z \in [t-1]\}$ only):

$$d \succ c_1^{(i)} \succ \cdots \succ c_{t-1}^{(i)} \succ \cdots .$$

   Note that, altogether, d gets h points from the voters in this group.

Note that our parameter, the number of voters, has value $\binom{h}{2} + h + (h-1)$. First, prior to adding any candidates, d has $h - 1 + \binom{h}{2}$ points while p has h points, and each of the dummy candidates has one point. We show next that it is possible to ensure that d is not a winner of this election by adding at most $k := \binom{h}{2}$ (bundles of) candidates if and only if G has a multi-colored clique of order h.

It follows that if G has a multi-colored clique V' of order h, then adding the edge candidates corresponding to the edges in the induced complete subgraph $G[V']$ makes d lose all $\binom{h}{2}$ points from the first group while maintaining that p still has h points. As a result, d is a not a winner.

For the reverse direction, assume that it is possible to preclude d from winning by adding at most $\binom{h}{2}$ (bundles of) candidates. First of all, note that candidate p is the only candidate that can reach a higher score than d since each registered dummy candidate and each newly added candidate will have at most one point. However, to make p have more points than d, candidate d must lose all $\binom{h}{2}$ points that d initially got from the first group of voters, and p must retain all the points from the second group of voters. Moreover, note that adding vertex candidates does not help. Thus, the added candidates must be exactly $\binom{h}{2}$ edge candidates whose bundles do not add two vertices of the same color. That is, these $\binom{h}{2}$ added edge candidates must correspond to a multi-colored clique of order h. $\square$

We conclude this section by mentioning that the following results also follow by applying the Multi-Colored Clique technique. The proofs are available in Appendix A.

Theorem 4.5. *For each fixed integer t, $t \geq 2$, t-Approval-*CCAC*, parameterized by the number of voters, is* W[1]*-hard.*

Theorem 4.6. *For each fixed integer t, $t \geq 2$, t-Veto-*CCAC*, parameterized by the number of voters, is* W[1]*-hard.*

Theorem 4.7. *For each fixed integer $t \geq 1$, t-Veto-*CCDC*, parameterized by the number of voters, is* W[1]*-hard.*

Theorem 4.8. 2*-Approval-*CCDC*, parameterized by the number of voters, is* W[1]*-hard.*

Theorem 4.9. *For each fixed integer t, $t \geq 3$, t-Approval-*CCDC*, parameterized by the number of voters, is* W[1]*-hard.*

5. Cubic Vertex Cover Proof Technique

We now move on to the Cubic Vertex Cover proof technique. Specifically, we use it to obtain the following results (again, all results are for the parameterization by the number of voters):

1. Borda-CCAC and Borda-CCDC are NP-hard (this holds already for elections with ten voters).

2. For each rational α, $0 \leq \alpha \leq 1$, Copeland$^{\alpha}$-CCAC and Copeland$^{\alpha}$-CCDC are NP-hard (this holds already for elections with twenty and twenty-six voters, respectively).

3. Maximin-CCAC is NP-hard (this holds already for elections with ten voters).

In other words, we use the Cubic Vertex Cover technique for all our non-combinatorial Para-NP-hardness results. In this section we provide proofs for the cases of Borda-CCDC and Maximin-CCAC, while the remaining ones are in Appendix B. We made this choice because the proofs for Borda-CCDC and Maximin-CCAC illustrate the essential elements of the technique (as applied both to an adding-candidates case and to a deleting-candidates case, and both to a scoring rule and a Condorcet consistent rule).

The general idea of the Cubic Vertex Cover technique is to prove Para-NP-hardness via reductions from the Cubic Vertex Cover problem—known to be NP-hard (Garey, Johnson, & Stockmeyer, 1976)—using the fact that cubic graphs[6] can be encoded using a constant number of votes. Formally, the Cubic Vertex Cover problem is defined as follows.

6. In a cubic graph, each vertex is of degree exactly three, that is, it has exactly three neighbors.

Definition 3. An instance of Cubic Vertex Cover consists of an undirected graph $G = (V(G), E(G))$, where each vertex of G has degree exactly three, and a non-negative integer h. We ask if there is a subset (*vertex cover*) of at most h vertices such that each edge is incident to at least one vertex in the subset.

All the reductions in this section use the following common setup. Let I be an instance of Cubic Vertex Cover with a graph G and non-negative integer h. From a classic result (Vizing, 1965), we know that there is an edge-coloring of G with four colors (that is, it is possible to assign one out of four colors to each edge so that no two edges incident to the same vertex have the same color). Further, it is possible to compute this coloring in polynomial time (Misra & Gries, 1992). This is equivalent to saying that it is possible to decompose the set of G's edges into four disjoint matchings. Our reductions start by computing this decomposition. We rename the edges of G so that these four disjoint matchings are:

$$E^{(1)} = \{e_1^{(1)}, \dots, e_{m_1}^{(1)}\},$$
$$E^{(2)} = \{e_1^{(2)}, \dots, e_{m_2}^{(2)}\},$$
$$E^{(3)} = \{e_1^{(3)}, \dots, e_{m_3}^{(3)}\},$$
$$E^{(4)} = \{e_1^{(4)}, \dots, e_{m_4}^{(4)}\}.$$

We set $m' = m_1 + m_2 + m_3 + m_4 = |E(G)|$ and $n' = |V(G)|$. For each edge e of the graph, we arbitrarily order its vertices and we write $v'(e)$ and $v''(e)$ to refer to the first vertex and to the second vertex, respectively. For each ℓ, $1 \leq \ell \leq 4$, we write $E^{(-\ell)}$ to mean $E(G) \setminus E^{(\ell)}$. We write $V^{(-\ell)}$ to mean the set of vertices that are not incident to any of the edges in $E^{(\ell)}$.

The crucial point of our approach is to use the above decomposition to create eight votes (two for each matching) that encode the graph. We will now provide useful notation for describing these eight votes. For each edge e of the graph, we define the following four orders over e, $v'(e)$, and $v''(e)$:

$$P(e)\colon e \succ v'(e) \succ v''(e),$$
$$P'(e)\colon e \succ v''(e) \succ v'(e),$$
$$Q(e)\colon v'(e) \succ v''(e) \succ e,$$
$$Q'(e)\colon v''(e) \succ v'(e) \succ e.$$

For each ℓ, $1 \leq \ell \leq 4$, we define the following orders over $V(G) \cup E(G)$:

$$A(\ell)\colon P(e_1^{(\ell)}) \succ P(e_2^{(\ell)}) \succ \cdots \succ P(e_{m_\ell}^{(\ell)}),$$
$$A'(\ell)\colon P'(e_{m_\ell}^{(\ell)}) \succ \cdots \succ P'(e_2^{(\ell)}) \succ \cdots \succ P'(e_1^{(\ell)}),$$
$$B(\ell)\colon Q(e_1^{(\ell)}) \succ Q(e_2^{(\ell)}) \succ \cdots \succ Q(e_{m_\ell}^{(\ell)}),$$
$$B'(\ell)\colon Q'(e_{m_\ell}^{(\ell)}) \succ \cdots \succ Q'(e_2^{(\ell)}) \succ \cdots \succ Q'(e_1^{(\ell)}).$$

Note that since each $E^{(\ell)}$ is a matching, each of the above orders is well-defined. The first two of these families of orders (that is, $A(\ell)$ and $A'(\ell)$) will be useful in the hardness proofs for the cases of deleting candidates, and the latter two (that is, $B(\ell)$ and $B'(\ell)$) in the hardness proofs for the cases of adding candidates. The intuitive idea behind orders $A(\ell)$ and $A'(\ell)$ (or $B(\ell)$ and $B'(\ell)$) is that, at a high level, they are reverses of each other, but they treat edges and their endpoints in a slightly asymmetric way (we will describe this in detail in the respective proofs).

We are ready to show examples of applying the Cubic Vertex Cover technique. We start with the case of Borda-CCDC (we present the theorem and its proof first, and right after that, we show an example of applying the reduction).

Theorem 5.1. *Borda-*CCDC *is* NP*-hard, even for elections with only ten voters.*

Proof. We give a reduction from the Cubic Vertex Cover problem. Let I be our input instance that contains graph $G = (V(G), E(G))$ and non-negative integer h. We use the notation introduced in the beginning of the section. We form an election $E = (C, V)$, where $C = \{p, d\} \cup V(G) \cup E(G)$. We introduce the following ten voters; recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X:

1. For each ℓ, $1 \leq \ell \leq 4$, we have the following two voters:

$$\mu(\ell)\colon A(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$
$$\mu'(\ell)\colon p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).$$

2. We have one voter with preference order $p \succ d \succ V(G) \succ E(G)$ and one voter with preference order $\overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$.

We claim that p can become a winner of this election by deleting at most $k := h$ candidates if and only if there is a vertex cover of size h for G.

Let us first calculate the scores of all the candidates:

1. Candidate p has $5(n' + m') + 6$ points (that is, $4(n' + m' + 1)$ points from the first eight voters and $n' + m' + 2$ points from the last two voters).

2. Each vertex candidate v has $5(n' + m') + 2$ points (for each of the three pairs of voters $\mu(\ell)$, $\mu'(\ell)$, $1 \leq \ell \leq 4$, such that v is incident to some edge in $E^{(\ell)}$, v gets $n' + m'$ points; v gets $n' + m' + 1$ points from the remaining pair of voters in the first group and, additionally, $n' + m' + 1$ points from the last two voters).

3. Each edge candidate e has $5(n' + m') + 7$ points, that is, $n' + m' + 3$ points from the pair of voters $\mu(\ell)$, $\mu'(\ell)$ such that $e \in E^{(\ell)}$, $n' + m' + 1$ points from each pair of the remaining three pairs of voters in the first group, and $n' + m' + 1$ points from the last two voters.

4. Candidate d has $5(n' + m') + 4$ points (that is, $4(n' + m' + 1)$ points from the voters in the first group and $n' + m'$ points from the last two voters.

Prior to deleting any of the candidates, p is not a winner because the edge candidates have higher scores. However, the score of p is higher than the score of the vertex candidates and the score of d.

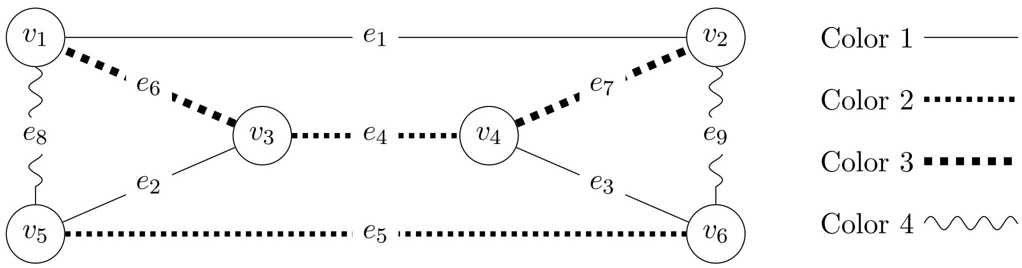

Figure 2: The input graph for our example for Theorem 5.1. The four different line styles represent the colors of the edges according to an assumed partition to four colors. For example, the edge e_6 has the third color.

We now describe how deleting candidates affects the scores of the candidates. Let v be some vertex candidate. Deleting v from our election causes the following effects: The score of each edge candidate e such that $v = v'(e)$ or $v = v''(e)$ decreases by six; the score of each remaining candidate decreases by five. This means that if we delete h vertex candidates that correspond to a vertex cover of G, then the scores of p, d, and all the vertex candidates decrease by $5h$, while the scores of all edge candidates decrease by at least $5h + 1$. As a result, we have p as a winner of the election.

For the reverse direction, assume that it is possible to ensure p's victory by deleting at most h candidates. Deleting candidate d decreases the score of p by six, whereas it decreases the scores of every other candidate by five. Thus, we can assume that there is a solution that does not delete d. Similarly, one can verify that if there is a solution that deletes some edge e, then a solution that is identical but instead of e deletes either $v'(e)$ or $v''(e)$ (it is irrelevant which one) is also correct. We conclude that it is possible to ensure p's victory by deleting at most h vertex candidates. However, by the discussion of the effects of deleting vertex candidates and the fact that prior to any deleting each edge candidate has one point more than p, we have that these at-most-h deleted vertex candidates must correspond to a vertex cover of G. This completes the proof. □

Example 2. *We provide an example for the reduction described in the proof of Theorem 5.1. The input graph is depicted in Figure 2 and we take* $h := 4$. *We present the constructed election in Table 3. The election that results from deleting candidates* $\{v_1, v_3, v_4, v_6\}$ *that correspond to a vertex cover is presented in Table 4.*

Let us now show an application of the Cubic Vertex Cover technique to the case of adding candidates. Specifically, we consider Maximin-CCAC.

Theorem 5.2. *Maximin-*CCAC *is* NP*-hard, even for elections with only ten voters.*

Proof. We give a reduction from Cubic Vertex Cover (we use the notation as provided at the beginning of this section). Given an instance (G, h) for Cubic Vertex Cover, we construct an instance for Maximin-CCAC. We let the registered candidate set be $C = \{p, q\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct ten voters:

$$\mu(1)\colon e_1 \succ v_1 \succ v_2 \succ e_2 \succ v_3 \succ v_5 \succ e_3 \succ v_4 \succ v_6 \succ E^{(-1)} \succ V^{(-1)} \succ d \succ p$$
$$\mu'(1)\colon p \succ d \succ \overleftarrow{V^{(-1)}} \succ \overleftarrow{E^{(-1)}} \succ e_3 \succ v_6 \succ v_4 \succ e_2 \succ v_5 \succ v_3 \succ e_1 \succ v_2 \succ v_1$$
$$\mu(2)\colon e_4 \succ v_3 \succ v_4 \succ e_5 \succ v_5 \succ v_6 \succ E^{(-2)} \succ V^{(-2)} \succ d \succ p$$
$$\mu'(2)\colon p \succ d \succ \overleftarrow{V^{(-2)}} \succ \overleftarrow{E^{(-2)}} \succ e_5 \succ v_6 \succ v_5 \succ e_4 \succ v_4 \succ v_3$$
$$\mu(3)\colon e_6 \succ v_1 \succ v_3 \succ e_7 \succ v_2 \succ v_4 \succ E^{(-3)} \succ V^{(-3)} \succ d \succ p$$
$$\mu'(3)\colon p \succ d \succ \overleftarrow{V^{(-3)}} \succ \overleftarrow{E^{(-3)}} \succ e_7 \succ v_4 \succ v_2 \succ e_6 \succ v_3 \succ v_1$$
$$\mu(4)\colon e_8 \succ v_1 \succ v_5 \succ e_9 \succ v_2 \succ v_6 \succ E^{(-4)} \succ V^{(-4)} \succ d \succ p$$
$$\mu'(4)\colon p \succ d \succ \overleftarrow{V^{(-4)}} \succ \overleftarrow{E^{(-4)}} \succ e_9 \succ v_6 \succ v_2 \succ e_8 \succ v_5 \succ v_1$$
$$\text{one voter}\colon p \succ d \succ V(G) \succ E(G)$$
$$\text{one voter}\colon \overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$$

Table 3: The election constructed in the proof of Theorem 5.1 for the input graph from Figure 2.

1. For each ℓ, $1 \le \ell \le 4$, we have the following two voters; recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X:
$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ p \succ q,$$
$$\mu'(\ell)\colon p \succ q \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2. We have one voter with preference order $q \succ V(G) \succ E(G) \succ p$ and one voter with preference order $\overleftarrow{E(G)} \succ q \succ p \succ \overleftarrow{V(G)}$.

Let E be the constructed election (including all registered and unregistered candidates). We have the following values of the $N_E(\cdot,\cdot)$ function (recall that this function represents the head-to-head contests):

1. For each vertex $v \in V(G)$, we have $N_E(p,v) = 5$ (and thus, $N_E(v,p) = 5$), and $N_E(q,v) = 6$ (and thus, $N_E(v,q) = 4$).
2. For each edge $e \in E(G)$, we have $N_E(p,e) = 4$ (and thus, $N_E(e,p) = 6$), and $N_E(q,e) = 5$ (and thus, $N_E(e,q) = 5$).
3. $N_E(p,q) = 8$ (and thus, $N_E(q,p) = 2$).
4. For each vertex $v \in V(G)$ and each edge $e \in E(G)$ we have the following: If v is an endpoint of e, then $N_E(v,e) = 6$ (so $N_E(e,v) = 4$), and otherwise we have $N_E(v,e) = 5$ (so $N_E(e,v) = 5$).

$$\mu(1)\colon e_1 \succ \cancel{v_1} \succ v_2 \succ e_2 \succ \cancel{v_3} \succ v_5 \succ e_3 \succ \cancel{v_4} \succ \cancel{v_6} \succ E^{(-1)} \succ V^{(-1)} \succ d \succ p$$

$$\mu'(1)\colon p \succ d \succ \overleftarrow{V^{(-1)}} \succ \overleftarrow{E^{(-1)}} \succ e_3 \succ \cancel{v_6} \succ \cancel{v_4} \succ e_2 \succ v_5 \succ \cancel{v_3} \succ e_1 \succ v_2 \succ \cancel{v_1}$$

$$\mu(2)\colon e_4 \succ \cancel{v_3} \succ \cancel{v_4} \succ e_5 \succ v_5 \succ \cancel{v_6} \succ E^{(-2)} \succ V^{(-2)} \succ d \succ p$$

$$\mu'(2)\colon p \succ d \succ \overleftarrow{V^{(-2)}} \succ \overleftarrow{E^{(-2)}} \succ e_5 \succ \cancel{v_6} \succ v_5 \succ e_4 \succ \cancel{v_4} \succ \cancel{v_3}$$

$$\mu(3)\colon \{v_1, v_3\} \succ \cancel{v_1} \succ \cancel{v_3} \succ e_7 \succ v_2 \succ \cancel{v_4} \succ E^{(-3)} \succ V^{(-3)} \succ d \succ p$$

$$\mu'(3)\colon p \succ d \succ \overleftarrow{V^{(-3)}} \succ \overleftarrow{E^{(-3)}} \succ e_7 \succ \cancel{v_4} \succ v_2 \succ \{v_1, v_3\} \succ \cancel{v_3} \succ \cancel{v_1}$$

$$\mu(4)\colon e_8 \succ \cancel{v_1} \succ v_5 \succ e_9 \succ v_2 \succ \cancel{v_6} \succ E^{(-4)} \succ V^{(-4)} \succ d \succ p$$

$$\mu'(4)\colon p \succ d \succ \overleftarrow{V^{(-4)}} \succ \overleftarrow{E^{(-4)}} \succ e_9 \succ \cancel{v_6} \succ v_2 \succ e_8 \succ v_5 \succ \cancel{v_1}$$

$$\text{one voter}\colon p \succ d \succ V(G) \succ E(G)$$

$$\text{one voter}\colon \overleftarrow{E(G)} \succ \overleftarrow{V(G)} \succ p \succ d$$

Table 4: The election from Table 3 with the candidates corresponding to the vertex cover $\{v_1, v_3, v_4, v_6\}$ deleted.

5. For each pair of vertices, $v', v'' \in V(G)$, $N_E(v', v'') = 5$.

6. For each pair of edges, $e', e'' \in E(G)$, $N_E(e', e'') = 5$.

In effect, prior to adding the candidates, the score of p is four, the score of each edge candidate is five, and the score of q is two. Adding a vertex candidate v to the election has the following consequence: This vertex candidate v will have score four (because of q), the score of p does not change, but the score of each edge candidate that is incident to v will be decreased to four. Thus, it is possible to ensure p's victory by adding at most h candidates if and only if there is a size-h vertex cover for G. $\square$

We conclude the section by mentioning that the following results, whose proofs are in Appendix B, also follow by applying the Cubic Vertex Cover technique.

Theorem 5.3. *Borda*-CCAC *is* NP*-hard, even for elections with only ten voters.*

Theorem 5.4. *For each rational number α, $0 \le \alpha \le 1$, Copeland$^\alpha$*-CCAC *is* NP*-hard, even for elections with only twenty voters.*

Theorem 5.5. *For each rational number α, $0 \le \alpha \le 1$, Copeland$^\alpha$*-CCDC *is* NP*-hard, even for elections with only twenty-six voters.*

6. Set-Embedding Proof Technique for Combinatorial Variants

In this section we present our Set-Embedding proof technique for the combinatorial variants of our control problems. Specifically, we prove the following statements (again, all results are for the parameterization by the number of voters):

1. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}, \text{Borda}, \text{Copeland}^\alpha$ (for $0 \leq \alpha \leq 1$), Maximin$\}$, both $\mathcal{R}$-COMB-CCDC and $\mathcal{R}$-COMB-DCDC are Para-NP-hard.

2. For each voting rule $\mathcal{R} \in \{\text{Borda}, \text{Copeland}^\alpha$ (for $0 \leq \alpha \leq 1$)$\}$, Maximin$\}$, $\mathcal{R}$-COMB-CCAC is Para-NP-hard.

3. For each voting rule $\mathcal{R} \in \{\text{Borda}, \text{Copeland}^\alpha$ (for $0 \leq \alpha \leq 1$)$\}$, $\mathcal{R}$-COMB-DCAC is Para-NP-hard.

That is, in this section we provide all our Para-NP-hardness results for the combinatorial variants of our problems.

All proofs follow by reducing the SET COVER problem (recall Definition 2) to the respective problem in a way which uses the bundling function to encode the sets from the SET COVER instances (hence the name of the technique). We start by providing some common notation and observations common to all of these results.

Let $I = (X, \mathcal{S}, h)$ be an input instance of SET COVER (which is NP-hard; see, e.g., the book Garey & Johnson, 1979). We construct elections with candidate sets that include the elements from X and the sets from $\mathcal{S}$. Specifically, for each element $x_i \in X$, we introduce a candidate with the same name, and for each set $S_j \in \mathcal{S}$, we introduce a candidate named s_j. We denote the set of all element candidates by X_{cand} and denote the set of all set candidates by $\mathcal{S}_{\text{cand}}$. Further, we will typically have candidates p and d. For the constructive cases, p will be the preferred candidate while for the destructive cases, d will be the despised one.

Unless stated otherwise, in each of our proofs we use a bundling function κ defined as follows: for each set candidate s_j, we have $\kappa(s_j) = \{s_j\} \cup \{x_i \mid x_i \in S_j\}$, and for each non-set candidate c, we have $\kappa(c) = \{c\}$. We refer to this bundling function as the *set-embedding bundling function*.

The general idea of our proofs is that to ensure p's victory (for the constructive cases) or d's defeat (for the destructive cases), one has to add/delete all the candidates from X_{cand}, and due to the bound on the number of candidates that we can add/delete, this has to be achieved by adding/deleting the candidates from $\mathcal{S}_{\text{cand}}$ and relying on the bundling function.

With the above setup ready, we move on to proving our results. Most of the proofs are in Appendix C, but for each type of problem (COMB-CCAC, COMB-CCDC, COMB-DCAC, COMB-DCDC) we give one sample proof.

6.1 Constructive Control by Deleting Candidates

We start by looking at constructive control by deleting candidates because in this case we obtain a very general hardness result that applies to all the voting rules which satisfy the unanimity principle. A rule satisfies the *unanimity principle* if in each election where a unique candidate c is ranked first by all the voters, this candidate c is the unique winner.

Theorem 6.1. *Let $\mathcal{R}$ be a voting rule that satisfies the unanimity principle.* $\mathcal{R}$-Comb-CCDC *is* NP*-hard, even for the case of elections with just a single voter.*

Proof. Let the notation be as in the introduction to this section. Given an instance $I := (X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of $\mathcal{R}$-Comb-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$ and where V contains a single voter with the following preference order:

$$X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}.$$

We use the set-embedding bundling function. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to ensure p's victory by deleting at most h (bundles of) candidates.

On one hand, if I is a "yes"-instance of Set Cover, then I' is a "yes"-instance of $\mathcal{R}$-Comb-CCDC. Indeed, if $\mathcal{S}'$ is a subfamily of $\mathcal{S}$ such that $|\mathcal{S}'| \leq h$ and $\bigcup_{S_j \in \mathcal{S}'} S_j = X$, then it suffices to delete the candidates C' that correspond to the sets in $\mathcal{S}'$ from the election to ensure that p is ranked first (and, by the unanimity of $\mathcal{R}$, is a winner).

On the other hand, assume that I' is a "yes"-instance of $\mathcal{R}$-Comb-CCDC. Since $\mathcal{R}$ satisfies the unanimity property, the candidate ranked first by the only voter in our election is always the unique winner. This means that if I' is a "yes"-instance of $\mathcal{R}$-Comb-CCDC, then there is a subset C' of candidates such that $p \notin C'$ and $X \subseteq \bigcup_{c \in C'} \kappa(c)$. Without loss of generality, we can assume that C' contains only candidates from the set $\mathcal{S}_{\text{cand}}$ (if C' contained some candidate x_i, we could replace x_i with an arbitrary candidate s_j such that $x_i \in S_j$). However, this immediately implies that setting $\mathcal{S}' := \{S_j \mid s_j \in C'\}$ results in a set cover of size at most h. Therefore I is a "yes"-instance of I. ☐

As Plurality, Borda, Copeland$^\alpha$, and Maximin all satisfy the unanimity property, we conclude the following.

Corollary 6.2. *For each voting rule $\mathcal{R} \in \{$Plurality, Borda, Copeland$^\alpha$, Maximin$\}$,* $\mathcal{R}$-Comb-*CCDC is* NP*-hard, even for elections with only a single voter.*

By applying minor tweaks to the above construction, we obtain the following results (for the proofs see Appendix C).

Theorem 6.3. *For each fixed integer $t \geq 2$, t-Approval-*Comb-CCDC *is* NP*-hard, even for elections with only a single voter.*

Theorem 6.4. *For each fixed integer $t \geq 1$, t-Veto-*Comb-CCDC *is* NP*-hard, even for elections with only a single voter.*

6.2 Destructive Control by Deleting Candidates

While the very general proof for the combinatorial variant of constructive control by deleting candidates is very simple, occasionally our set-embedding proofs become slightly more involved. For example, our proof that Maximin-Comb-DCDC is NP-hard even for elections with only few voters requires a bit more care.

Theorem 6.5. *Maximin-*Comb-DCDC *is* NP*-hard, even for elections with only five voters.*

Proof. Given an instance $(X, \mathcal{S}, h)$ for SET COVER, we construct an instance $(E = (C, V), d, \kappa, k)$ for Maximin-COMB-DCDC. We construct an election $E = (C, V)$ where $C := \{p, d, e\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$ and where the voter set consists of the following five voters:

$$\begin{aligned}\text{one voter}\colon & p \succ d \succ X_{\text{cand}} \succ e \succ \mathcal{S}_{\text{cand}},\\ \text{two voters}\colon & d \succ X_{\text{cand}} \succ p \succ e \succ \mathcal{S}_{\text{cand}},\\ \text{two voters}\colon & e \succ \overleftarrow{X_{\text{cand}}} \succ p \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}.\end{aligned}$$

Recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X. We set $k := h$. We use the set-embedding bundling functions. We claim that I is a "yes"-instance of SET COVER if and only if it is possible to ensure that d is not a winner by deleting at most $k = h$ (bundles of) candidates.

The values of the $N_E(\cdot, \cdot)$ function are given in the table below (the entry for row a and column b gives the value of $N_E(a, b)$; we assume $i' \neq i''$ and $j' \neq j''$).

	p	d	e	$x_{i'}$	$s_{j'}$
p	-	3	3	1	5
d	2	-	3	3	5
e	2	2	-	2	5
$x_{i''}$	4	2	3	2 or 3	5
$s_{j''}$	0	0	0	0	2 or 3

We have the following scores of the candidates: p has one point (because of the members of X_{cand}), d has two points (because of p), e has two points (because of p, d, and the members of X_{cand}), the members of X_{cand} have two points each (because of d), and the members of $\mathcal{S}_{\text{cand}}$ have zero points each (because of all other candidates).

Now, if there is a set cover for I of size h, then deleting the set candidates corresponding to the cover deletes all members of X_{cand} and ensures that p has three points, whereas d has only two. In effect, d certainly is not a winner.

Now consider the other direction. Since deleting a candidate can never decrease the score of any remaining candidate, the only way of making d lose is to increase some remaining candidate's score. Since for each candidate other than p, at least three voters prefer d to this candidate, only p has any chance of getting a higher score than d. To let p have a higher score than d, we need to make sure that p has at least three points (note that d will still have score two because of p). Thus, all candidates that cause p to have less than three points must disappear. As we can see from the table, this means that all members of X_{cand} have to disappear. As in the previous set-embedding proofs, this is possible to do by deleting at most h candidates only if there is a set cover of size at most h for I. ☐

Yet, for most of the other results it suffices to use proofs very similar to that for Theorem 6.1. However, for the case of t-Approval-COMB-DCDC we have to use either two voters (if $t \geq 2$) or three voters (if $t = 1$ and we are dealing with Plurality). The reason is that if we have a single voter and candidate d is a t-Approval winner, then it is impossible to prevent d from winning by deleting candidates (no matter what we do, d will still have the highest possible score, one). A similar reasoning applies to the case of Plurality and two voters. We omit the proofs of the following results (they are available in Appendix C).

Theorem 6.6. *Plurality-*COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

Theorem 6.7. *For each fixed integer $t \geq 2$, t-Approval-*COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Theorem 6.8. *For each fixed integer $t \geq 1$, t-Veto-*COMB-DCDC *is* NP*-hard, even for elections with only a single voter.*

Theorem 6.9. *Borda-*COMB-DCDC *is* NP*-hard, even for elections with only two voters.*

Theorem 6.10. *Copeland$^\alpha$-*COMB-DCDC *is* NP*-hard, even for elections with only three voters.*

6.3 Constructive and Destructive Control by Adding Candidates

For the case of combinatorial control by adding candidates, we give sample proofs for the cases of Borda.

Theorem 6.11. *Borda-*COMB-CCAC *and Borda-*COMB-DCAC *are both* NP*-hard, even for elections with only two voters.*

Proof. We first show the NP-hardness result for Borda-COMB-CCAC and then show how to modify the proof for Borda-COMB-DCAC.

Let the notation be as in the introduction to this section. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER with $n' := |X_{\text{cand}}|$, we create an instance I' of Borda-COMB-CCAC as follows. We construct the registered candidate set $C = \{d, p\} \cup D$, where $D = \{d_1, \dots, d_{n'}\}$. We construct the unregistered candidate set $A = X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. We construct two voters with the following preference orders:

$$d \succ D \succ p \succ \mathcal{S}_{\text{cand}} \succ X_{\text{cand}} \text{ and}$$
$$p \succ \overleftarrow{X_{\text{cand}}} \succ d \succ \mathcal{S}_{\text{cand}} \succ \overleftarrow{D}.$$

Recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X. We use the set-embedding bundling function. We claim that I is a "yes"-instance of SET COVER if and only if it is possible to ensure p's victory by adding at most h (bundles of) candidates. Note that d gets n' points more than p from the first voter. Given a set cover of size h, we add the corresponding s_j's to the election. Simple calculation shows that in this case p and d tie as winners.

For the reverse direction, note that the relative scores of p and d in the first vote do not change irrespective which candidates we add. The relative scores of p and d, however, do change in the second vote in the following way: For each unregistered candidate x_i added to the election, p's score increases by one but d's score remains unchanged. Thus, the only way to ensure that p is a winner is by bringing all the candidates from X_{cand} to the election. Doing so by adding at most h candidates is possible only if there is a size-h cover for I.

The construction for Borda-COMB-DCAC is the same, except that, first, we do not want p to win but d to lose (that is, we define d to be the despised candidate, and second, we define D to have only $n - 1$ dummy candidates. ☐

The proofs for the remaining results are available in Appendix C. Note that technically similar Para-NP-hardness results already follow from our discussion of the non-combinatorial variants; using the set-embedding technique we can give proofs that use fewer voters.

Theorem 6.12. *For each rational number α, $0 \leq \alpha \leq 1$, Copeland$^\alpha$-*COMB-DCAC *and Copeland$^\alpha$-*COMB-CCAC *are* NP*-hard, even for elections with only three voters.*

Theorem 6.13. *Maximin-*COMB-CCAC *is* NP*-hard, even for elections with only six voters.*

7. Signature Proof Technique for Destructive Control

We now move on to the discussion of the signature technique of obtaining FPT algorithms. Specifically, in this section we show the following results for the parameter "number of voters":

1. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathcal{R}$-DCAC is in FPT.

2. Plurality-COMB-DCAC and Veto-COMB-DCAC are in FPT.

3. For each fixed integer $t \geq 1$ and for each voting rule $\mathcal{R} \in \{t\text{-Approval}, t\text{-Veto}\}$, $\mathcal{R}$-DCDC is in FPT.

That is, we apply the technique to the case of destructive candidate control, under t-Approval and t-Veto elections. The main idea of the signature technique is to identify certain properties of the candidates to be added/deleted that allow us to treat them as equivalent. We refer to these properties as signatures and we build FPT algorithms based on the observations that the number of different signatures (in a given context) is a function of the number of voters only.

The results from this section apply, in particular, to the cases of Plurality-DCDC and Veto-DCDC. However, these problems are simple enough that direct algorithms for them that are easier and faster; we provide such algorithms in Section 8.1.

7.1 Destructive Control by Adding Candidates

Let us consider an instance of the Destructive Control by Adding Candidates problem for the case of t-Approval/t-Veto (for now we focus on the non-combinatorial variant). The instance consists of the set C of registered candidates, the set A of unregistered candidates, the collection V of n voters, the despised candidate $d \in C$, and an integer k bounding the number of candidates that we can add. We assume that d is a winner in election (C, V) (otherwise we can trivially accept the input). The general scheme for our FPT algorithm (parameterized by the number n of the voters) is as follows:

1. We guess a candidate $p \in C \cup A$ ($p \neq d$). The role of p is to defeat d, that is, to obtain more points than d. Altogether there are $m := |C| + |A|$ candidates and we repeat our algorithm for each possible choice of p.

2. For each choice of p, we "kernelize" the input instance, that is, we bound the number of "relevant" candidates by a function of the parameter n, and search for an optimal

solution in a brute-force manner over this "kernel" (indeed, the signature technique regards computing this kernel).[7]

3. We accept if the best solution found adds at most k candidates and we reject otherwise.

We now describe how to perform the kernelization step. Let us consider the registered candidates first. It turns out that it suffices to focus on a few relevant ones only.

Definition 4 (Relevant registered candidates)**.** Fix an integer t, $t \geq 1$, and consider an instance of t-Approval-DCAC. We call a registered candidate *relevant* if this candidate receives *at least* one point. For the case of t-Veto-DCAC, we call a registered candidate *relevant* if this candidate receives *at least* one veto. We refer to those candidates that are not relevant as *irrelevant*.

For the case of t-Approval-DCAC, we can safely remove all the irrelevant registered candidates. This is so for two reasons: First, removing an irrelevant candidate does not change the score of any other registered (or later-added) candidate. Second, an irrelevant candidate can never obtain score higher than the despised one because, on the one hand, initially this candidate has score zero, and, on the other hand, under t-Approval adding candidates never increases the scores of those already registered.

For the case of t-Veto-DCAC, if there is an irrelevant candidate then there are two possibilities. If d receives at least one veto, then d already is not a winner (because the irrelevant candidate receives no vetoes and, thus, defeats d). If d does not receive any vetoes, then this stays so, irrespective what candidates we add, and d remains a winner. In either case, we can immediately output the correct answer.

Thus, from now on we assume that all the registered candidates are relevant. For each t-Approval-DCAC instance (t-Veto-DCAC instance) with n voters, at most $t \cdot n$ candidates are relevant.

To deal with the unregistered candidates, we introduce the notion of a $\{d, p\}$-signature (recall that d is the despised candidate and p is a candidate whose goal is to defeat d). Let c be some unregistered candidate. Each voter can rank c in three different ways relative to p and d: This voter can rank c ahead of both p and d, below both p and d, or between p and d (a finer distinction is not necessary). A $\{d, p\}$-signature for c is a vector in $[3]^n$ that for each voter indicates which of these cases holds. Formally, we use the following definition.

Definition 5 ($\{d, p\}$-Signature)**.** Consider an election $(C \cup A, V)$, a candidate $d \in C$, and a candidate $p \in C \cup A$. Let n be the number of voters in V. A $\{d, p\}$-signature of candidate $c \in (C \cup A) \setminus \{d, p\}$ is a size-n vector $\vec{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_n) \in [3]^n$ such that for each voter $v_i \in V$, it holds that:

$$\gamma_i = \begin{cases} 3, & \text{if } v_i \text{ prefers } c \text{ to both } p \text{ and } d, \\ 1, & \text{if } v_i \text{ prefers both } p \text{ and } d \text{ to } c, \\ 2, & \text{otherwise.} \end{cases}$$

7. We mention that this kind of kernelization is called *Turing kernelization*. See the works of Binkele-Raible, Fernau, Fomin, Lokshtanov, Saurabh, and Villanger (2012) and Schäfer, Komusiewicz, Moser, and Niedermeier (2012) for examples of this concept in the context of graph problems.

Now, observe that for a given choice of p, adding exactly t candidates with the same $\{d, p\}$-signature has the same effect on the relative scores of p and d as adding more than t such candidates.

Lemma 7.1. *Consider an instance $I := ((C, V), A, d \in C, k)$ of t-Approval-DCAC (of t-Veto-DCAC), with the despised candidate d, and with some arbitrarily selected candidate $p \in C \cup A$. Let $\vec{\gamma}$ be some $\{d, p\}$-signature for this election. Adding t unregistered candidates with signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding more than t candidates with this signature.*

Proof. Let us focus on the case of t-Approval-DCAC. Let n be the number of voters in instance I. We have $\vec{\gamma} = (\gamma_1, \ldots, \gamma_n)$. Consider the ith voter.

1. If $\gamma_i = 3$, then after adding t candidates with signature $\vec{\gamma}$, the ith voter will give 0 points to both p and d.

2. If $\gamma_i = 1$, then the ith voter will give the same number of points to p (respectively, to d) as prior to adding candidates, irrespective how many candidates with signature $\vec{\gamma}$ we add.

3. If $\gamma_i = 2$, then for each candidate c with signature $\vec{\gamma}$, the ith voter ranks c between candidates p and d. Since this voter may have preferences either $p \succ d$ or $d \succ p$, this implies that either for each candidate c with signature $\vec{\gamma}$, the ith voter has preference order $p \succ c \succ d$, or for each candidate c with signature $\vec{\gamma}$, the ith voter has preference order $d \succ c \succ p$. In the first case, adding t (or more) candidates with signature $\vec{\gamma}$ will ensure that the ith voter gives zero points to d and gives the same number of points to p as prior to adding the candidates. In the second case, the situation is the same, but with the roles of p and d swapped.

Summing over the points provided by all voters, this proves that adding t candidates with a given signature $\vec{\gamma}$ has the same effect on the relative scores of p and d as adding any more such candidates.

The argument for the case of t-Veto-DCAC is analogous. ☐

In effect, it suffices to keep at most t candidates with each signature. This results in having at most $t \cdot 3^n$ unregistered candidates. We can now formally describe our kernelization process.

Theorem 7.2. *For each fixed integer $t \geq 1$, t-Approval-*DCAC *and t-Veto-*DCAC *admit Turing kernels of size $O(t \cdot 3^n)$, where n is the number of voters.*

Proof. Consider an instance I of t-Approval-DCAC (of t-Veto-DCAC). Let d be the despised candidate and let n be the number of voters in the instance. As per our discussion, we can assume that the instances are non-trivial and that all registered candidates are relevant. Thus, there are at most $t \cdot n$ registered candidates. By Lemma 7.1, for each choice of p it suffices to consider 3^n $\{d, p\}$-signatures, and for each signature at most t candidates. Altogether, for each choice of candidate p among the registered and unregistered candidates, we produce an instance of t-Approval-DCAC (of t-Veto-DCAC), with at most $t \cdot n$ registered candidates and at most $t \cdot 3^n$ unregistered ones (for each possible signature we keep up to t arbitrarily chosen unregistered candidates); in each instance we can add either the same number of candidates as in I, or one less, if p is an "added" candidate already. It is possible to preclude d from winning in the original instance if and only if it is possible to do so in one of the produced instances. $\square$

Using a brute-force approach on top of the kernelization given by Theorem 7.2, it is possible to solve both t-Approval-DCAC and t-Veto-DCAC in FPT time: straightforward application of a brute-force search to each instance produced by Theorem 7.2 gives running time $O^*(\binom{t \cdot 3^n}{k})$, where the O^* notation suppresses polynomial terms. However, it never makes sense to add more than $t \cdot n$ candidates (intuitively, if we added more than $t \cdot n$ candidates, then at least one would be irrelevant and we could as well not add him). Thus we can assume that $k \leq t \cdot n$. In effect, the straightforward brute-force algorithm running on top of Theorem 7.2 has running time $O^*((t \cdot 3^n)^{t \cdot n})$. However, if we are willing to spend more space, then we can obtain significantly better running times.

Theorem 7.3. *Plurality*-DCAC *can be solved in time $O(m \cdot n \cdot 2^n)$, using $O^*(2^n)$ space, where m is the total number of candidates and n is the number of voters.*

Proof. Our algorithm uses a similar general structure as before. We assume that we are given a non-trivial instance, where all the registered candidates are relevant. First, we guess a candidate p whose goal is to defeat d and from now on we focus on a situation where we have both p and d, and the goal is to ensure that p gets more points than d. (If p is an unregistered candidate, then we add p to the election, decrease the number of candidates that we can add by one, and proceed as if p were a registered candidate to begin with.)

We define a simplified notion of a candidate's signature. A *signature* for an unregistered candidate c is a size-n binary vector $\vec{\tau} = (\tau_1, \ldots, \tau_n) \in \{0,1\}^n$ such that the following hold:

1. We have $\tau_i = 1$ if the ith voter ranks candidate c ahead of all the registered candidates.

2. We have $\tau_i = 0$ if the ith voter ranks candidate c below some registered candidate.

We define the *signature* $\vec{\tau}$ of a set A' of unregistered candidates analogously: Value one at a given position means that some candidate from A' is ranked ahead of all the registered candidates and value zero means that some registered candidate is ranked ahead of all the members of A'.

Let k be the number of candidates that we are allowed to add. Using the new notion of signatures, we maintain a size-2^n table $\vec{\mathcal{Z}} := (\mathcal{Z}_{\vec{\tau}}) \in [k+1]^{2^n}$, which for each signature $\vec{\tau} \in \{0,1\}^n$ stores the minimum number $\mathcal{Z}_{\vec{\tau}}$ of unregistered candidates such that there is a size-$\mathcal{Z}_{\vec{\tau}}$ ($\mathcal{Z}_{\vec{\tau}} \leq k$) candidate subset $A^{(\vec{\tau})} \subseteq A \setminus \{p\}$ with signature $\vec{\tau}$ (value $k+1$ indicates that there is no such set).

To describe our algorithm for computing table $\vec{\mathcal{Z}}$, we need one more piece of notation. For each pair of signatures $\vec{\tau}$ and $\vec{\tau}'$, we define a "merged" signature $\vec{\tau} \oplus \vec{\tau}' = (\max\{\tau_i, \tau'_i\})_{i \in [n]}$. In other words, we apply the coordinate-wise max operator. We compute table $\vec{\mathcal{Z}}$ as follows (our algorithm is slightly more complicated than necessary for the case of Plurality rule, but we will also use it as a base for more involved settings):

1. We initiate the table by setting $\mathcal{Z}_{\vec{\tau}} := 1$ if there is at least one unregistered candidate with signature $\vec{\tau}$, and we set $\mathcal{Z}_{\vec{\tau}} := k + 1$ otherwise.

2. For each unregistered candidate a we perform the following operations:

   (a) We compute a's signature $\vec{\tau}_a$.

   (b) We compute a new table $\mathcal{Z}'$, by setting, for each signature $\vec{\tau}$:

   $$\mathcal{Z}'_{\vec{\tau}} = \min(\{\mathcal{Z}_{\vec{\tau}}\} \cup \{\mathcal{Z}_{\vec{\tau}'} + 1 \mid \vec{\tau} = \vec{\tau}' \oplus \vec{\tau}_a\} \cup \{k+1\}).$$

   (c) We copy the contents of $\mathcal{Z}'$ to $\mathcal{Z}$. (At this point, for each signature $\vec{\tau}$, $\mathcal{Z}_{\vec{\tau}}$ is the number of candidates in the smallest set (of size up to k) composed of the so-far processed candidates that jointly have this signature, or is $k + 1$ if no such set exists.)

3. We pick a signature $\vec{\tau}$ such that $\mathcal{Z}_{\vec{\tau}}$ has a minimum value and adding the candidate set $A_{\vec{\tau}}$ that implements this signature ensures that p has more points than d (note that this last condition is easy to check: Given a signature $\vec{\tau}$, if the ith component τ_i is zero, then the ith voter gives one point to whomever this voter ranks first among the registered candidates; if τ_i is one, then the point goes to a candidate from $A_{\vec{\tau}}$, that is, neither to p nor d). If $\mathcal{Z}_{\vec{\tau}}$ is smaller or equal to k, the number of candidates that we can add, then we accept. Otherwise we reject (for this choice of p).

Let us first consider the algorithm's running time. The most time-consuming part of the algorithm is the loop in the second step of the procedure computing the table $\mathcal{Z}$. For each out of at most m candidates, computing $\mathcal{Z}'$ requires filling in $O(2^n)$ entries of the table. If we first copy the then-current contents of $\mathcal{Z}$ to $\mathcal{Z}'$, and then perform the remaining updates, this can be done in time $O(m \cdot n \cdot 2^n)$. This dominates the running time of the remaining parts of the algorithm.

Now let us consider the correctness of the algorithm. Assume that we have guessed the correct candidate p and that there is a subset of unregistered candidates $A' = \{a_1, \ldots, a_\ell\}$ such that p has more points than d after we add the candidates from A' and $\ell \leq k$. If $\vec{\tau}$ is the signature of the set A', then notice that the algorithm indeed computes value $\mathcal{Z}_{\vec{\tau}} \leq \ell$. Further, if the algorithm accepts, then it must have found a solution. Thus the algorithm is correct. □

We can apply the above ideas to the case of t-Approval and t-Veto as well. The proofs are in Appendix D.

Theorem 7.4. *For each fixed integer $t \geq 2$, t-Approval-*DCAC *can be solved in time* $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$*, where m is the total number of candidates and n is the number of voters.*

Adapting the algorithms in a straightforward way (basically by inverting, or reversing, the signatures) used for Theorem 7.3 and Theorem 7.4, we can show analogous results for the case of t-Veto-DCAC.

Corollary 7.5. *For each fixed integer $t \geq 1$, t-Veto-DCAC can be solved in time* $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$*, where m is the total number of candidates and n is the number of voters.*

To conclude the discussion of the signature technique for the case of destructive control by adding candidates, we consider the combinatorial variant of the problem. The situation is more complicated because now we are adding bundles of candidates instead of individual candidates. In effect, we cannot upper-bound the number of bundles to add in t-Approval-Comb-DCAC (or t-Veto-Comb-DCAC). This is so because bundles with the same signature but with different sizes may have different effects on the score difference between the despised candidate d and a specific guessed candidate p (indeed, t-Approval-Comb-DCAC is W[1]-hard for $t \geq 2$, as shown in Section 4). Yet, for Plurality and for Veto only the first (or the last) position gets a point (a veto). This structural observation allows us to use our non-combinatorial algorithms.

Corollary 7.6. *Plurality-*Comb*-DCAC and Veto-*Comb*-DCAC, parameterized by the number of voters, are fixed-parameter tractable.*

Proof. For the case of Plurality, it suffices to use, for example, the same algorithm as in Theorem 7.3, but with the following changes:

1. For each choice of candidate p, we also consider each way of adding p to the election if p were unregistered (p might belong to several different bundles and we try each possibility).

2. Each unregistered candidate's signature is replaced by the signature of the set of candidates in its bundle.

Since under Plurality each voter gives a point only to whomever this voter ranks first, this strategy suffices. The case of the Veto rule is handled analogously. □

7.2 Destructive Control by Deleting Candidates

Let us now move on to the case of destructive control by deleting candidates. The (Turing) kernelization approach from the previous section cannot be easily transferred to the case of deleting candidates. This is because we cannot upper-bound the number of candidates that have to be deleted in terms of the number n of the voters in the election. However, applying our signature technique followed by casting the remaining task as an integer linear program (ILP), we can show fixed-parameter tractability (for our parameterization by the number of voters). We present the proof of the following result in Appendix D (while the proof is quite interesting technically and we encourage the reader to read it, we also believe that the previous proofs have presented the signature technique sufficiently well).

Theorem 7.7. *For each fixed integer $t \geq 1$, both t-Approval-*DCDC *and t-Veto-*DCDC *can be solved in time $O^*(m \cdot 4^n \cdot (3^n)^{3^n})$, where m is the total number of candidates and n is the number of voters.*

We leave it as an open question whether there is a direct (combinatorial) FPT algorithm that avoids ILPs as used in Theorem 7.7; notably, for many voting problems which can be shown to be FPT via ILPs, it is the case that no other, direct algorithm is known (Bredereck, Chen, Faliszewski, Guo, Niedermeier, & Woeginger, 2014).

8. Remaining Results: Membership in XP, FPT, and P

In this section we present our remaining algorithms that show membership of our problems in XP, FPT, and P. In the cases of XP and FPT membership, our algorithms use a simple brute-force approach.

8.1 Fixed-Parameter Tractability Results

We now show simple, fast FPT algorithms for Plurality-CCDC, Plurality-DCDC, and Veto-DCDC. The main idea for the algorithms in this section is to guess a subset of voters that will give a specific candidate one point under either Plurality or Veto. The key observation is that in the case of deleting candidates, after guessing this subset of voters, it is trivial to find the set of candidates to delete to "implement" this guess.

Theorem 8.1. *Plurality-*CCDC *can be solved in $O(m \cdot n \cdot 2^n)$ time, where n is the number of voters and m is the number of candidates in the input election.*

Proof. Let $I \coloneqq ((C, V), p, k)$ be a Plurality-CCDC instance. If I is a yes-instance, then after deleting at most k candidates, there must be a subset of voters who each give candidate p one point, and no other candidate has more points than p. Observe that in order to let p gain one point from a voter, one has to delete all the candidates this voter prefers to p. Our algorithm, based on these observations, proceeds as follows.

We consider all 2^n subsets of n voters. For each considered set V' of voters we do the following: For each voter $v' \in V'$, we delete all candidates that v' prefers to p. In effect, all members of V' rank p first. Then, we keep deleting all candidates that have more than $|V'|$ points (note that deleting some candidate that has more than $|V'|$ points may result in some other candidate exceeding this bound). If in the end no candidate has more than $|V'|$ points and we deleted at most k candidates, then we accept. Otherwise, we proceed to the next subset of voters. If we did not accept after going over all subsets of voters, then we reject.

To see why the algorithm is correct, note that whenever it accepts, it has constructed a correct solution. If, however, there is a correct solution in which, after deleting the candidates, p gets points exactly from the voters in some subset V', then the algorithm will accept when considering this subset. Establishing the running time is straightforward. $\square$

It is straightforward to see how to adapt the algorithm from Theorem 8.1 to the destructive case. In essence, it suffices to try all choices of a candidate p whose goal is to defeat the despised candidate d and for each such choice guess a subset of voters that are to give points to p. If after deleting the candidates that these voters prefer to p (assuming that none of them prefers d to p) the despised candidate d has fewer points than p, then we accept. In the destructive case there is no need to have the final loop of deleting candidates scoring higher than p.

Corollary 8.2. *Plurality-DCDC can be solved in $O(m^2 \cdot n \cdot 2^n)$ time, where n is the number of voters and m is the number of candidates in the input election.*

We provide an analogous result for the case of Veto.

Theorem 8.3. *Veto-*DCDC *can be solved in* $O(m \cdot n \cdot 2^n)$ *time, where* n *is the number of voters and* m *is the number of candidates in the input election.*

Proof. We use almost the same approach as for Theorem 8.1. First, we guess candidate p whose goal is to have fewer vetoes than d. Deleting candidates can only increase the number of vetoes that a remaining candidate has. Thus, our algorithm proceeds as follows.

We consider every subset V' of voters that prefer p to d in the election. For each voter v' in the guessed subset, we delete all candidates that this voter ranks below d (by choice of V', p is never deleted). If in effect d has more vetoes than p, we accept. Otherwise we try the next subset of voters. If we do not accept after processing all subsets of voters, then we reject.

Establishing the correctness and the running time of the algorithm is straightforward. □

8.2 XP Results

In this section, we establish XP results for all our W[1]-hard problems. This implies that if the number of voters is a constant, then the problems are polynomial-time solvable.

Theorem 8.4. *For each fixed integer* t*,* $t \geq 1$*, and for each control type* $\mathcal{K} \in \{CCAC, CCDC\}$*,* t*-Approval-*$\mathcal{K}$ *and* t*-Veto-*$\mathcal{K}$ *can be solved in time* $O(m^{tn} \cdot m \cdot n)$*, where* m *is the number of candidates and* n *is the number of voters.*

Proof. We consider the CCAC and the CCDC cases jointly, in parallel for both t-Approval and t-Veto. Our algorithm first guesses for each voter the set of t candidates that this voter will rank first (for the case of t-Approval) or last (for the case of t-Veto). There are $O(m^{tn})$ possible different guesses.

For each guess, for each voter, we verify which candidates have to be added (for the case of CCAC) or deleted (for the case of DCAC) to ensure that the voter ranks the guessed t candidates on top. If it suffices to add/delete k candidates to implement the guess, and in effect of implementing the guess our preferred candidate is a winner, then we accept. Otherwise we proceed to the next guess. If no guess leads to acceptance, then we reject. Establishing the correctness and the running time of the algorithm is immediate. □

Theorem 8.5. *For each fixed integer* t*,* $t \geq 1$*, and each control type* $\mathcal{K} \in \{CCAC, DCAC\}$*,* t*-Approval-*Comb*-*$\mathcal{K}$ *and* t*-Veto-*Comb*-*$\mathcal{K}$ *can be solved in time* $O(m^{2tn} \cdot m \cdot n)$*, where* m *is the total number of candidates and* n *is the number of voters.*

Proof. We use the same approach as described in the proof of Theorem 8.4, but in addition to guessing the first t candidates for each vote, we also guess for each added candidate c the candidate to whose bundle c belongs. □

8.3 Polynomial-Time Solvable Case

There is one case that is still missing, which we show next.

Theorem 8.6. *Maximin-*Comb-DCAC *can be solved in* $O(m^3 \cdot n)$ *time, where* m *is the number of candidates and* n *is the number of voters.*

Proof. It was shown by Faliszewski et al. (2011) that Maximin-DCAC is polynomial-time solvable. The same strategy can be applied for the combinatorial case as well. To this end, let $((C,V),A,d,\kappa,k)$ be an Instance for Maximin-Comb-DCAC.

The algorithm is simple and can be described as follows: We guess up to two candidates, add their bundles to the election, and check whether the despised candidate d is no longer a winner; if so, we accept and otherwise we reject.

To see why this simple algorithm is correct, consider a solution, that is, a set A' of at most k unregistered candidates whose bundles are to be added. If A' consists of at most two candidates, then we are done. Otherwise, let us take a closer look at the set A'. It is clear that in the resulting election $E' := (C \cup \kappa(A'), V)$, d is not a winner. Therefore, there must be at least one other candidate p that has a higher score than d. Consider the bundle b_p of some candidate in $\kappa(A')$ such that b_p includes p (indeed, there might be several such candidates, and we can choose any one of them arbitrarily; it is also possible that p is present in the original election, in which case we take b_p to be an "empty" bundle). Further, consider some candidate z in $\kappa(A')$ such that the Maximin score of candidate d in the election E' is exactly $N_{E'}(d,z)$. There may be several such candidates and we choose one arbitrarily. Finally, we choose an arbitrary candidate from A' whose bundle b_z includes z (in fact, it is possible that z is present in the original election, in which case we take b_z to be an "empty" bundle).

It is clear that p defeats d in the election $(C \cup \kappa(x,y), V)$ where x and y are the leaders of the bundles b_p and b_z, respectively (if either of these bundles is "empty", then we simply disregard it). Thus, each "yes"-instance of Maximin-DCAC has a solution that consists of at most two candidates and, consequently, it is enough to guess and test at most two unregistered candidates. □

9. Outlook

Our work motivates several possible research directions, some of which are listed below.

1. We still do not know the exact complexity of 2-Veto-Comb-DCAC. This open question is marked by a question mark (?) in Table 1.

2. Some of our hardness results state specific number of voters for which the respective problems are hard. In certain cases our results are possibly tight, but in others perhaps not. One might consider tightening these results in additional hope to better understand those problems and perhaps to devise better proof techniques.

3. Some of our proofs show hardness for making p a co-winner, while others show hardness for making p the unique co-winner. It might be interesting to further investigate whether requiring specific variant could vary the computational complexity of the corresponding problems.

4. It is natural to consider an even more diverse set of voting rules. This might allow for understanding our general techniques better, and might help in devising new techniques as well. For example, it would be interesting to consider the Bucklin rule. On the one hand, candidate control problems for Bucklin are NP-complete (Erdélyi et al., 2015a) and, on the other hand, the rule can be seen as an adaptive variant of

t-Approval. Thus it would be interesting to see if our techniques can be applied to the case of Bucklin.

5. One can experiment with real-world elections to understand the practical relevance of our theoretical findings or heuristically solve our proven worst-case intractable cases. One possible starting point for such an analysis would be the experimental paper of Erdélyi et al. (2015b).

6. It is interesting to consider some game-theoretic aspects of candidate control, where several agents perform the control actions. So far, doing this even for the simplest rules such as Plurality was hampered by the fact that these control problems are NP-hard. Our (partial) tractability results might help in overcoming this obstacle.

7. It might be worthwhile to study the multimode control framework (Faliszewski et al., 2011) for the case of few voters. In multimode control one can perform control actions of several types at the same time (for example, one can add candidates and delete voters). Faliszewski et al. expected that combining two types of easy control actions would lead to a possibly computationally hard multimode control problem, but they did not observe such effects among natural voting rules. One possible explanation for this fact is that they did not have enough easy control problems available to combine. We have shown that many candidate control problems become easy (in FPT) when they are parameterized by the number of voters and, thus, there are more opportunities for studying multimode control problems.

Generally, we believe that the case of few voters did not receive sufficient attention in the computational social choice literature and many other problems can (and should) be studied with respect to this parameter. The main two reasons for the study of this parameter are as follows. First, it is very well motivated, as discussed in Section 1.1. Second, in our control problems we observe a rich (parameterized) complexity landscape. We hope that such rich landscapes exist for other voting problems, when parameterized by the number of voters. Indeed, this turned out to be the case in several recent studies conducted by some of the authors of this paper (Faliszewski et al., 2016; Bredereck et al., 2016a) as well as by other researchers (Misra et al., 2015).

Acknowledgments

Jiehua Chen was supported by TU Berlin while she was affiliated with TU Berlin, where the major part of the work was done. She was also supported by the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) under REA grant agreement number 631163.11 and Israel Science Foundation (grant no. 551145/14). Piotr Faliszewski was supported by DFG project PAWS (NI 369/10) and by AGH University grant 11.11.230.337 (statutory research). Nimrod Talmon was supported by DFG, Research Training Group "Methods for Discrete Structures" (GRK 1408), while the author was affiliated with TU Berlin, where most of the work was done. This work was also partly supported by COST Action IC1205 on Computational Social Choice.

A preliminary short version of this work has been presented at the 29th AAAI Conference on Artificial Intelligence (AAAI '15), Austin Texas, January, 2015 (Chen, Faliszewski,

Niedermeier, & Talmon, 2015). In this version, in addition to providing all proofs, we fixed a flaw in our multi-colored clique proof technique. The flaw was pointed out and fixed by Maushagen and Rothe (2016).

Appendix A. Deferred Proofs for the Multi-Colored Clique Proof Technique

Theorem 4.5. For each fixed integer t, $t \geq 2$, t-Approval-CCAC, parameterized by the number of voters, is W[1]-hard.

Proof. One can use the same proof as in the case of Theorem 4.1, but for each voter we introduce $t-1$ additional registered dummy candidates which this voter ranks first (each voter ranks all the remaining dummy candidates last). In this way, each dummy candidate has exactly one point. The reasoning for the proof of correctness works in the same way. ☐

Theorem 4.6. For each fixed integer t, $t \geq 2$, t-Veto-CCAC, parameterized by the number of voters, is W[1]-hard.

Proof. One can use the same proof as in Theorem 4.2, but we introduce $t-1$ additional registered dummy candidates whom every voter ranks last. In this way, each dummy candidate receives exactly one veto from each voter, while p and d receive the same number of vetoes as in the election constructed in the proof for Theorem 4.2.

One can verify that the arguments from that proof apply here as well. ☐

Theorem 4.7. For each fixed integer $t \geq 1$, t-Veto-CCDC, parameterized by the number of voters, is W[1]-hard.

Proof. We use almost the same proof as in Theorem 4.3, but we add sufficiently many dummy (padding) candidates to ensure that we can only delete vertex and edge candidates. Let $I = (G, h)$ be an input instance of Multi-Colored Clique. Let $E' = (C', V')$ be the election created by the reduction from the proof of Theorem 4.3 on input I and set $k := |V(G)| - h + 2|E(G)| - H$.

We modify this election by extending C' to contain a set D of t dummy candidates, $D = \{d_1, \ldots, d_t\}$, and modify the voter collection V' as follows (recall that the number $|V'|$ of voters is a function polynomially bounded by h; set $n' := |V'|$):

1. For each voter v in V' except the last group of voters, we modify v's preference order to rank the dummies $d_1, \ldots, d_{t-1}$ last and d_t first.

2. For each voter v in the last group of V', we rank all candidates from D such that v will have a preference order of the form
$$d_t \succ \cdots \succ (D \setminus \{d_t\}) \succ p.$$

3. We add n' voters, all with preference order of the form
$$\cdots \succ p \succ D.$$

One can verify that each newly added candidate d_i, $1 \leq i \leq t-1$, has $2n'$ vetoes and d_t has n' vetoes. Since we assume the input graph to be connected and to have at least two vertices, at least one candidate from the edge and vertex candidates receives fewer vetoes than p. Thus, p is not a winner initially.

We claim that p (the preferred candidate from the proof of Theorem 4.3) can become a winner by deleting at most k candidates if and only if I is a "yes"-instance.

First, we note that if we delete any of the new dummy candidates from $D \setminus \{d_t\}$, then p certainly does not become a winner since p will have at least $n' + 2H$ vetoes and d_t will have exactly n' vetoes. If we delete dummy candidate d_t, then p will receive $2n'$ vetoes, but there is at least one remaining vertex or edge candidate which is not vetoed by the last group of voters and has hence less than $2n'$ vetoes. In consequence, no dummy candidate can be deleted. Thus, none of them will have fewer vetoes than p and (ignoring the dummy candidates) the election will behave as if it was held according to the Veto rule. The argument from the proof of correctness in Theorem 4.3 holds. ☐

Theorem 4.8. 2-Approval-CCDC, parameterized by the number of voters, is W[1]-hard.

Proof. The proof is quite similar to that for the case of Veto-CCDC, but now the construction is a bit more involved. Again, we give a parameterized reduction from the Multi-Colored Clique problem. Let $I = (G, h)$ be our input instance with graph G and non-negative integer h, and let the notation be as described in the introduction to Section 4. We form an instance I' of 2-Approval-CCDC based on I. We build our candidate set C as follows, where we set $T = |V(G)| + |E(G)|$ with the intended meaning that T is an integer larger than the number of candidates that we can delete; we set $H \coloneqq 2\binom{h}{2} = (h-1) \cdot h$:

1. Introduce the preferred candidate p.
2. Introduce T candidates $B = b_1, \ldots, b_T$ (these are the *blocker* candidates whose role, on the one hand, is to ensure that p has to obtain a given number of points and, on the other hand, is to prevent deleting too many candidates of other types).
3. For each vertex $v \in V(G)$, introduce candidate v.
4. For each edge $\{u, v\} \in E(G)$, introduce two candidates, (u, v) and (v, u).
5. Introduce two sets $D = \{d_1, \ldots d_h\}$ and $F = \{f_{(i,j)} \mid 1 \leq i, j \leq h, i \neq j\}$ of dummy candidates.

Before we describe the set of voters, we need some additional notation. By writing B in a preference order, we mean
$$b_1 \succ b_2 \succ \cdots \succ b_T.$$
For each two colors i, j ($1 \leq i, j \leq h$, $i \neq j$), by $F(i, j)$ we mean an arbitrary (but fixed) ranking of all the candidates of the form (u, v), where $u \in V_i(G)$ and $v \in V_j(G)$. The set of voters consists of the following groups:

1. We have $h + 3H$ voters, each with preference order of the form
$$B \succ \cdots \succ p.$$

2. For each color i, $1 \leq i \leq h$, there are $3H+1$ voters, where the first of them has preference order of the form
$$V_i(G) \succ p \succ B \succ \cdots ,$$
and the remaining ones have preference order of the form
$$V_i(G) \succ d_i \succ B \succ \cdots .$$

3. For each two distinct color i and j ($1 \leq i, j \leq h$, $i \neq j$), there are $3H+h-1$ voters, where the first of them has preference order of the form
$$F(i,j) \succ p \succ B \succ \cdots ,$$
the next two have preference orders of the form
$$f_{(i,j)} \succ E(i,j) \succ B \succ \cdots ,$$
and the remaining ones have preference orders of the form
$$F(i,j) \succ f_{(i,j)} \succ B \succ \cdots .$$

4. For each two distinct color i and j ($1 \leq i, j \leq h$, $i \neq j$), we introduce two voters with the following preference orders of the forms
$$p \succ R(i,j) \succ B \succ \cdots ,$$
$$p \succ R'(i,j) \succ B \succ \cdots .$$

Note that the total number of constructed voters is polynomially bounded by h:
$$h + 3H + (3H+1) \cdot h + (3H+h-1) \cdot H + 2H = \quad 2h + 4H + 4H \cdot h + 3H^2.$$

We set the number of candidates that can be deleted to $k := |V(G)| - h + 2|E(G)| - H$, with the intention that p can become a winner if and only if it is possible to delete all of the vertex candidates and all of the edge candidates except for the ones corresponding to a multi-colored clique of order h. We note that if G indeed contains an order-h multi-colored clique Q, then deleting all candidates in $V(G) \setminus Q$ and all edge candidates of the form (u, v) where either $u \notin Q$ or $v \notin Q$ indeed ensures that p is a winner (in this case p, and all of the vertex and edge candidates have $h + 3H$ points each, and all of the blocker candidates have at most $h + 3H$ points each).

Now we come to the reverse direction. Assume that it is possible to ensure p's victory by deleting at most k candidates and let $C' \subseteq C$ be a set of at most k candidates such that p is a winner of $E' = (C \setminus C', V)$. Note that $k < T - 1$ and so there are at least two blocker candidates that receive $h + 3H$ points each from the first group of voters. The only voters from whom p can obtain additional points after deleting at most k candidates are the ones in the second and third group and there are exactly $h + H$ of them (h in the second group and H in the third group). However, p can obtain the points from the second and the third groups of voters without, at the same time, increasing the score of the highest-scoring blocker candidate if and only if: (a) we delete all-but-one vertex candidates of each color, and (b) for each two distinct colors i and j ($1 \leq i, j \leq h$, $i \neq j$) all-but-one edge candidates of the form (u, v), where $u \in V_i(G)$ and $v \in V_j(G)$. This means deleting exactly k candidates.

Let us now argue that for each two distinct colors i and j ($1 \leq i, j \leq h$, $i \neq j$), the two remaining edge candidates (u, v) and (u', v') such that $u \in V_i(G), v \in V_j(G)$ and $u' \in V_j(G), v' \in V_i(G)$ regard the same edge, that is, $\{u, v\} = \{u', v'\}$. Indeed, if it were not the case, then one of the edge candidates (u, v) and (u', v') would receive at least $3H + h + 1$ points (from the third group of the voters, for color choices (i, j) and (j, i)) and p would not be a winner. Thus our claim holds.

Further, we claim that if p is a winner of E', then for each pair of not-deleted vertex candidates u and v, it must be the case that both edge candidates (u, v) and (v, u) are still in the election, meaning that there is an edge between u and v in the original graph. It suffices to consider the case of (u, v) (the case of (v, u) is symmetric). If instead of (u, v) the only not-deleted edge candidate for the pair of colors of u and v is some edge candidate (u', v') (where $(u', v') \neq (u, v)$), then one of the two following cases must happen: either u and v would receive more than $h - 1$ points from the fourth group and, therefore, would have more than $h + 3H$ points altogether, causing p not to be a winner, or (u', v') would receive more than one point from the fourth group, again causing p to not be a winner (this latter case holds because from the previous paragraph we know that (v', u') must be included in the election). Thus p can become a winner by deleting at most H candidates if and only if G contains a multi-colored clique of order h.

It is clear that the given reduction can be computed in polynomial time and that it is a parameterized reduction, therefore the proof is complete. ◻

Theorem 4.9. For each fixed integer t, $t \geq 3$, t-Approval-CCDC, parameterized by the number of voters, is W[1]-hard.

Proof. Let $E' = (C', V')$ be the election constructed in the proof for Theorem 4.8. One can use the same proof as for Theorem 4.8 except that now for each voter $v_i \in V'$ we introduce a group of $t-2$ new dummy candidates, $d_1^i, d_2^i, \ldots, d_{t-2}^i$, that are ranked first, and for each such introduced group, we introduce two new dummies, c_1^i and c_2^i, and one voter with preference order of the form (we write D_i to refer to the preference order $d_1^i \succ d_2^i \succ \ldots \succ d_{t-2}^i$):

$$D_i \succ c_1^i \succ c_2^i \succ B \succ \cdots .$$

These voters ensure that none of the new dummy candidates can be deleted without increasing the score of the highest-scoring blocker candidate. If a score of a highest-scoring blocker candidate increases, then the preferred candidate no longer has any chance of winning. If none of the new dummy candidates can be deleted, then the correctness proof works the same as the one given for Theorem 4.8.

The number of voters is still polynomially bounded by the clique order h. ◻

Appendix B. Deferred Proofs for the Cubic Vertex Cover Proof Technique

Theorem 5.3. Borda-CCAC is NP-hard, even for elections with only ten voters.

Proof. We give a reduction from CUBIC VERTEX COVER (we use the notation as provided at the beginning of Section 5). Given an instance $I = (G, h)$ for CUBIC VERTEX COVER, we construct an instance for Borda-CCAC. We let the registered candidate set C be $\{p, d\} \cup E(G)$, and we let $V(G)$ be the set of unregistered candidates. We construct the following voters:

1. For each ℓ, $1 \leq \ell \leq 3$, we have the following two voters; recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X:

$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$
$$\mu'(\ell)\colon p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2. For $\ell = 4$, we have the following two voters:

$$\mu(\ell)\colon B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$
$$\mu'(\ell)\colon d \succ p \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

3. We have two voters with preference orders

$$E(G) \succ p \succ d \succ V(G),$$
$$p \succ \overleftarrow{E(G)} \succ d \succ \overleftarrow{V(G)}.$$

We claim that it is possible to ensure p's victory by adding $k \coloneqq h$ candidates if and only if there is a vertex cover of size h for G.

Let $m' \coloneqq |E(G)|$ be the number of edges in $E(G)$. Note that at the beginning, p has $5m' + 5$ points, d has $4m' + 5$ points, and each edge candidate has $5m' + 6$ points. Thus p is not a winner. Adding each unregistered vertex candidate v causes the scores of all candidates to increase: For the edge candidates that include v as an endpoint this increase is by five points, whereas for all other candidates this increase is by six points. Note that the last two voters always prefer the registered candidates to any vertex candidate. Thus, by simple counting, each of these h vertex candidates may obtain at most $4m' + 5h + 7$ points and will never obtain more points than p as long as $m' + h \geq 2$.

Thus, if we have a vertex cover of size h, then it is possible to ensure p's victory by adding all vertex candidates that correspond to this vertex cover. For the other direction, assume that it is possible to ensure p's victory by adding at most h candidates and let S be such a set of candidates. For the sake of contradiction, assume that there is an edge candidate e which is not covered by some vertex candidate in S. It follows that the score of e is greater than the score of p, which is a contradiction. Thus S must correspond to a vertex cover in G. $\square$

Theorem 5.4. For each rational number α, $0 \leq \alpha \leq 1$, Copeland$^{\alpha}$-CCAC is NP-hard, even for elections with only twenty voters.

Proof. We give a reduction from Cubic Vertex Cover (we use the notation as provided at the beginning of Section 5). Given an instance (G, h) for Cubic Vertex Cover, we construct an instance for Copeland$^{\alpha}$-CCAC. Let the registered candidate set C be $\{p, d\} \cup E(G)$, and let $V(G)$ be the set of unregistered candidates. We introduce the following voters; recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X:

1. For each ℓ, $1 \leq \ell \leq 4$, we construct four voters, two voters with the following preference order:
$$B(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ d \succ p,$$
and two voters with the following preference order:
$$p \succ d \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ B'(\ell).$$

2. One voter with the preference order $E \succ V \succ d \succ p$, and one voter with the preference order $d \succ p \succ \overleftarrow{E} \succ \overleftarrow{V}$.

3. One voter with the preference order $p \succ V \succ E \succ d$, and one voter with the preference order $\overleftarrow{E} \succ d \succ p \succ \overleftarrow{V}$.

We illustrate the results of head-to-head contests between the candidates in Figure 3. We claim that there is a vertex cover of size at most h for G if and only if p can become a winner of the election by adding at most $k := h$ candidates.

Sometimes, when we say that a vertex candidate and an edge candidate are *adjacent* to each other, we mean that the corresponding vertex and edge are adjacent to each other. Consider a situation where we have added some subset A' of k candidates ($k \leq h$; take $k = 0$ to see the situation prior to adding any of the unregistered candidates). The candidates have the following scores:

1. p has score $\alpha m' + k$ (p ties head-to-head contests with all edge candidates and wins all head-to-head contests with the vertex candidates).

2. d has score $1 + \alpha k$ (d wins the head-to-head contest with p and ties all head-to-head contests with the vertex candidates).

3. Each added vertex candidate v has score $3 + \alpha k$ (v ties the head-to-head contests with d and the remaining $k - 1$ vertex candidates and wins the head-to-head contests with the three edge candidates that are adjacent to v).

4. Each edge candidate e has score $\alpha m' + k + 1 - c(e)$, where $c(e)$ is the number of vertex candidates from A' that are adjacent to e (e ties head-to-head contests with p and the remaining edge candidates and wins head-to-head contests with d and all added vertex candidates except those that are adjacent to e).

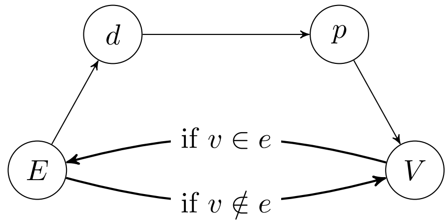

Figure 3: Illustration for the reduction used in the proof of Theorem 5.4. Each vertex in the graph corresponds to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are omitted. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoints of the edge candidate.

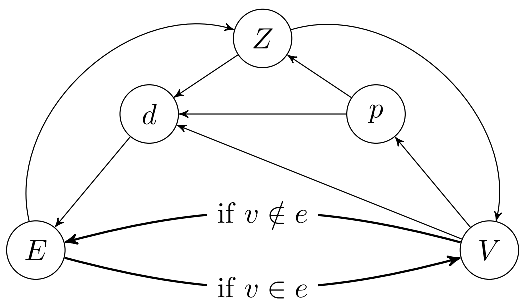

Figure 4: Illustration for the reduction used in the proof of Theorem 5.5. Each vertex in the graph corresponds to a candidate or a set of candidates, and there is an arc going from a vertex u_1 to a vertex u_2 if u_1 beats u_2 in a head-to-head contest. Edges indicating ties are omitted. The main idea is that an edge candidate beats a vertex candidate if and only if the vertex candidate is one of the endpoints of the edge candidate.

In effect, it holds that p is a winner of the election if and only if A' corresponds to a vertex cover of G. $\square$

Theorem 5.5. For each rational number α, $0 \leq \alpha \leq 1$, Copeland$^{\alpha}$-CCDC is NP-hard, even for elections with only twenty-six voters.

Proof. We give a reduction from Cubic Vertex Cover (we use the notation as provided at the beginning of Section 5). Given an instance $I = (G, h)$ for Cubic Vertex Cover, we construct an instance for Copeland$^{\alpha}$-CCDC. The candidate set contains the edge candidates, the vertex candidates, the preferred candidate p, a dummy candidate d, and a set of additional dummy candidates $Z = \{z_1, \ldots, z_{m'+n'}\}$ (recall that $m' \coloneqq |E(G)|$ denotes the number of edges in $E(G)$ and that $n' \coloneqq |V(G)|$ denotes the number of vertices in $V(G)$). We construct the following voters; recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X:

1. For each ℓ, $1 \leq \ell \leq 4$, we construct two voters with preference order:

$$A(\ell) \succ E^{(-\ell)} \succ V^{(-\ell)} \succ Z \succ d \succ p,$$

and two voters with preference order:

$$p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V^{(-\ell)}} \succ \overleftarrow{E^{(-\ell)}} \succ A'(\ell).$$

2. We also construct the following ten voters:

$$\begin{aligned}
v_1 &: V \succ E \succ Z \succ d \succ p, \\
v_1' &: p \succ d \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ \overleftarrow{E}, \\
v_2 &: V \succ p \succ d \succ E \succ Z, \\
v_2' &: \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V} \succ p \succ d, \\
v_3 &: p \succ Z \succ d \succ V \succ E, \\
v_3' &: \overleftarrow{E} \succ \overleftarrow{V} \succ p \succ \overleftarrow{Z} \succ d, \\
v_4 &: d \succ E \succ Z \succ V \succ p, \\
v_4' &: p \succ \overleftarrow{V} \succ \overleftarrow{Z} \succ d \succ \overleftarrow{E}, \\
v_5 &: Z \succ V \succ E \succ d \succ p, \\
v_5' &: p \succ d \succ \overleftarrow{E} \succ \overleftarrow{Z} \succ \overleftarrow{V}.
\end{aligned}$$

Figure 4 illustrates the results of the head-to-head contests among the candidates. Prior to deleting any of the candidates, we have the following scores:

1. each edge candidate e has $m'+n'+\alpha m'+2$ points (e wins head-to-head contests against all candidates in Z due to voters v_2 and v_2', wins head-to-head contests against its "incident" vertex candidates due to the first group of voters, and ties with p and the remaining edge candidates),

2. each vertex candidate u has $\alpha(n'-1)+m'-1$ points (u wins head-to-head contests against all edge candidates that are *not* "incident" to u due to voters from the first group, and ties with the remaining vertex candidates),

3. each candidate z from Z has $n'+1+\alpha(m'+n'-1)$ points (z wins head-to-head contests against all vertex candidates and d due to voters v_3, v_3', v_5, v_5', and ties with the remaining candidates from Z),

4. d has m' points (d wins head-to-head contests against all edge candidates due to voters v_4 and v_4'), and

5. p has $m'+n'+\alpha m'+1$ points (p wins head-to-head contests against all candidates from Z due to voters v_3 and v_3', wins head-to-head contests against d due to voters v_2, v_2', v_3, v_3', and ties with all edge candidates).

Thus, all edge candidates are co-winners, and p is not a winner because each edge candidate has one point more than p. However, p has more points than any other non-edge candidate. Note that in the input graph it holds that $m' = 3n'/2$.

We claim that it is possible to ensure that p is a winner by deleting at most $k \coloneqq h$ candidates if and only if there is a vertex cover of size h for G.

If there is a vertex cover for G of size h, then deleting the corresponding h vertices ensures that p is a winner. To see why this is the case, note that after deleting vertices corresponding to a vertex cover the score of p does not change, but the score of each edge candidate decreases by at least one. The scores of other candidates do not increase, so p is a winner.

For the reverse direction, assume that it is possible to ensure that p is a winner by deleting at most h candidates. Deleting candidates cannot increase p's score, so it must be the case that each edge candidate loses at least one point.

Observe that deleting candidates other than the vertex candidates will not make the edge candidates lose more than one point than p. The only possibility of deleting a candidate such that an edge candidate e loses a point but p does not is by deleting one of the vertex candidates $v'(e)$ or $v''(e)$. Thus, if it is possible to ensure that p is a winner, then we must delete vertices that correspond to a vertex cover. ☐

Appendix C. Deferred Proofs for the Set-Embedding Proof Technique

Theorem 6.3. For each fixed integer $t \geq 2$, t-Approval-Comb-CCDC is NP-hard, even for elections with only a single voter.

Proof. We build upon the proof of Theorem 6.1, but add $t-1$ dummy candidates. Specifically, given an instance $I \coloneqq (X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of t-Approval-Comb-CCDC as follows. We construct an election $E = (C, V)$ where $C = \{p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D$, where $D = \{d_1, \ldots, d_{t-1}\}$, and where V contains a single voter with the following preference order:

$$D \succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}.$$

We use the bundling function as described in the introduction to the set-embedding section. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to ensure p's victory by deleting at most h (bundles of) candidates.

To see the correctness of the argument, note that if there is a solution that ensures p's victory by deleting a specific number of candidates, then there is also a solution that achieves the same and does not delete any of the dummy candidates (it is always at least as useful to delete one of the set candidates instead of a dummy one). ☐

Theorem 6.4. For each fixed integer $t \geq 1$, t-Veto-Comb-CCDC is NP-hard, even for elections with only a single voter.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I \coloneqq (X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of t-Veto-Comb-CCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, z\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D,$$

where $D = \{d_1, \ldots, d_{t-1}\}$ is a set of dummy candidates (indeed, for $t = 1$, that is, for Veto, $D = \emptyset$), and with the voter collection V containing a single voter with the following preference order:

$$z \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}} \succ D \succ p.$$

We use the set-embedding bundling function, with the added feature that $\kappa(z) = \mathcal{S}_{\text{cand}}$. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to ensure p's victory by deleting at most $h+1$ bundles.

Using similar reasoning as used in Theorem 6.3, it follows that the only way of ensuring that p is a winner is to let all the remaining candidates receive no points at all. The only way to achieve this is to first delete up to h candidates from $\{s_1, \ldots, s_m\}$ that correspond to a cover of the ground set and then to delete z. □

Theorem 6.6. Plurality-Comb-DCDC is NP-hard, even for elections with only three voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I :=(X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of Plurality-Comb-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$, and where V contains three voters with the following preference orders:

$$\begin{aligned}
&X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}} \succ d,\\
&d \succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and}\\
&p \succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}}.
\end{aligned}$$

We use the set-embedding bundling function. We claim that the despised candidate d can be precluded from winning by deleting at most h (bundles of) candidates if and only if there is a set cover of size h for I.

Prior to deleting any of the candidates, d, p, and one of the candidates from X are tied as winners. Since deleting candidates cannot make any candidate lose points and since deleting p will make d a unique winner, the only way of defeating d is by ensuring that the first voter gives her point to p. This means that all element candidates have to be removed from the election. By the same argument as in the previous proofs, doing so by deleting at most h candidates is possible if and only if I is a "yes"-instance of Set Cover. □

Theorem 6.7. For each fixed integer $t \geq 2$, t-Approval-Comb-DCDC is NP-hard, even for elections with only two voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I :=(X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of t-Approval-Comb-DCDC as follows. We construct an election $E = (C, V)$ with candidate set:

$$C = \{p, d\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}} \cup D \cup F,$$

where $D = \{d_1, \ldots, d_{t-2}\}$ and $F = \{f_1, \ldots, f_{t-1}\}$ are two sets of dummy candidates (note that D can be empty), and with the voter collection V containing two voters with the following preference orders:

$$\begin{aligned}
&d \succ X_{\text{cand}} \succ D \succ p \succ \mathcal{S}_{\text{cand}} \succ F \text{ and}\\
&p \succ F \succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}} \succ D.
\end{aligned}$$

We use the set-embedding bundling function. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

Initially, both d and p are winners (as well as some members of $X_{\text{cand}} \cup F$). Deleting p will make d gain one more point (from the second voter), making it impossible for d to lose. The same holds for the dummy candidates from set F. In other words, if we change the set of candidates that gain a point from the second voter, then d will obtain two points and will certainly be a winner. This implies that the only way of making d lose is to let either p or at least one candidate from F gain one point from the first voter. By construction of the first voter's preference order, this is possible only for p if and only if we delete all members of X_{cand}. As in the previous proofs, deleting them (through deleting at most h bundles of candidates) is possible if and only if I is a "yes"-instance of Set Cover. □

Theorem 6.8. For each fixed integer $t \geq 1$, t-Veto-Comb-DCDC is NP-hard, even for elections with only a single voter.

Proof. We use the same construction as used in Theorem 6.3 for t-Approval-Comb-CCDC but we reverse the preference order and replace p with d, the despised candidate:

$$\mathcal{S}_{\text{cand}} \succ d \succ X_{\text{cand}} \succ D.$$

The crucial observation here is that with only one voter, the only way of preventing d from winning is to rank her within the last t positions. This means that all element candidates have to "disappear" from the election (one could also try deleting the dummy candidates, but it is never a mistake to "make disappear" the members of X_{cand} instead, through deleting the appropriate candidates in $\mathcal{S}_{\text{cand}}$). Thus we can conclude that the set of deleted candidates contains the set candidates only. If d is to be precluded from winning by deleting at most h candidates, this set must correspond to a set cover of size h. Since we can assume that $h < |\mathcal{S}_{\text{cand}}|$, there is at least one set element not deleted, and this will be a winner. □

Theorem 6.9. Borda-Comb-DCDC is NP-hard, even for elections with only two voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I := (X, \mathcal{S}, h)$ for Set Cover, we create an instance I' of Borda-Comb-DCDC as follows. We construct an election $E = (C, V)$ where $C = \{p, d, z\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$ and where V contains two voters with the following preference orders:

$$d \succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}} \succ z \text{ and}$$
$$p \succ z \succ d \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{\mathcal{S}_{\text{cand}}}.$$

Recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X. We use the set-embedding bundling function. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to preclude d from winning by deleting at most h (bundles of) candidates.

For convenience, we calculate the scores of all candidates:

1. d has $2|\mathcal{S}_{\text{cand}}| + 2|X_{\text{cand}}| + 2$ points.

2. p has $2|\mathcal{S}_{\text{cand}}| + |X_{\text{cand}}| + 3$ points.

3. Each element candidate x_i has $2|\mathcal{S}_{\text{cand}}| + |X_{\text{cand}}| + 1$ points.

4. z has $|\mathcal{S}_{\text{cand}}| + |X_{\text{cand}}| + 1$ points.

5. Each set candidate s_j has $|\mathcal{S}_{\text{cand}}|$ points.

It follows that d has the highest number of points and, thus, is a winner.

Since both voters rank d ahead of the candidates in the set $X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$, no member of this set can have score higher than d, irrespective which other candidates we delete. Similarly, irrespective which candidates we delete, z will never have score higher than d. We conclude that candidate p is the only candidate that has a chance of defeating d.

Since deleting candidates does not increase the scores of any of the remaining candidates, to ensure that d is not a winner, we have to guarantee that he loses at least $|X_{\text{cand}}|$ points (relative to p). This means that it is possible to ensure that d is not a winner if and only if it is possible to remove all candidates from X_{cand}. However, this is possible if and only if I is a "yes"-instance of SET COVER. $\square$

Theorem 6.10. Copeland$^\alpha$-COMB-DCDC is NP-hard, even for elections with only three voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I :=$ $(X, \mathcal{S}, h)$ for SET COVER, we construct an instance for Copeland$^\alpha$-COMB-DCDC. Since our reduction will produce an instance with an odd number of voters, the particular value of α is immaterial. We form the set of candidates:

$$C = \{d, p\} \cup X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}.$$

We have three voters with the following preference orders:

$$\begin{aligned}
&p \succ d \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}}, \\
&d \succ X_{\text{cand}} \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and} \\
&\overleftarrow{X_{\text{cand}}} \succ p \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}.
\end{aligned}$$

Recall that for each subset X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X. We use the set-embedding bundling function. We claim that I is a "yes"-instance of SET COVER if and only if it is possible to preclude d's victory by deleting at most h (bundles of) candidates.

The initial scores are:

1. d receives $|\mathcal{S}_{\text{cand}}| + |X_{\text{cand}}|$ points (d wins head-to-head contests against all the other candidates but p).

2. p receives $|\mathcal{S}_{\text{cand}}|+1$ point (p wins head-to-head contests against d and all the members of $\mathcal{S}_{\text{cand}}$).

3. Each member x_i of X_{cand} receives at most $|\mathcal{S}_{\text{cand}}|+|X_{\text{cand}}|$ (from head-to-head contests with p, all members of $\mathcal{S}_{\text{cand}}$, and the other members of X_{cand}).

4. Each member s_j of $\mathcal{S}_{\text{cand}}$ receives at most $|\mathcal{S}_{\text{cand}}|-1$ points (from head-to-head contests with the other members of $\mathcal{S}_{\text{cand}}$).

Since deleting candidates cannot make any candidate gain more points, the only way of ensuring that d is not a winner is to make sure that d's score decreases relative to some other candidate. By the above list of scores, it follows that the only candidate that may end up with a score higher than d is p. This happens only if we remove all the members of X_{cand}. As in the previous proofs using the set-embedding technique, doing so by deleting at most h candidates is possible if and only if there is a set cover of size at most h for I. $\square$

Theorem 6.12. For each rational number α, $0 \leq \alpha \leq 1$, Copeland$^\alpha$-COMB-DCAC and Copeland$^\alpha$-COMB-CCAC are NP-hard, even for elections with only three voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I := (X, \mathcal{S}, h)$ for SET COVER with $n' := |X_{\text{cand}}|$, we construct an instance for Copeland$^\alpha$-COMB-DCAC. Since our reduction will produce an instance with an odd number of voters, the particular value of α is immaterial. We form the set of registered candidates:

$$C = \{d, p\} \cup D \cup F,$$

where d is the despised candidate (and we will want to ensure that p wins over d), and where $D := \{d_1, \ldots, d_{n'-2}\}$ and $F := \{f_1, \ldots, f_{n'-1}\}$ are two sets of dummy candidates. We let the set of unregistered candidates be $A = X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. Finally, we construct three voters with the following preference orders:

$$\begin{aligned}
&d \succ D \succ p \succ F \succ X_{\text{cand}} \succ \mathcal{S}_{\text{cand}},\\
&p \succ \overleftarrow{F} \succ \overleftarrow{X_{\text{cand}}} \succ \overleftarrow{D} \succ d \succ \overleftarrow{\mathcal{S}_{\text{cand}}}, \text{ and}\\
&X_{\text{cand}} \succ d \succ D \succ F \succ p \succ \mathcal{S}_{\text{cand}}.
\end{aligned}$$

Recall that for each set X, we write $\overleftarrow{X}$ to mean the reverse of some arbitrary but fixed order of X. We use the set-embedding bundling function. We claim that I is a "yes"-instance of SET COVER if and only if it is possible to preclude d's victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, we have the following scores:

1. d receives $2n'-2$ points (d wins head-to-head contests with all the remaining registered candidates).

2. p receives $n'-1$ points (p wins head-to-head contests with the members of F).

3. Every dummy candidate $d_i \in D$ receives at most $2n'-3$ points (d_i wins head-to-head contests with all the members of F, with p, and—at most—all remaining members of D).

4. Every dummy candidate $f_i \in F$ receives at most $n'-2$ points (f_i wins head-to-head contests with—at most—the remaining members of F).

Now, if there is a set cover for I of size at most h, then adding the members of $\mathcal{S}_{\text{cand}}$ that correspond to the cover ensures that d is not a winner (relative to d, p gets additional n' points).

For the reverse direction, note that adding candidates to the election cannot decrease the score of any existing candidate. Thus, in order to beat d, we must add candidates to increase (relative to d) the score of some candidate. We make several observations:

1. The candidates in $\mathcal{S}_{\text{cand}}$ themselves do not contribute to the increase of a score of any candidate relative to p because all the other candidates (including d) win head-to-head contests against them.

2. The scores of the members of D do not change relative to the score of d irrespective which other candidates join the election.

3. By the first observation in this enumeration, the maximum possible increase of a score of candidate is by n' points (if this candidate defeats all members of X_{cand} and members of X_{cand} join the election). Since all members of set F have score at most $n'-2$, none of them can obtain score higher than d, irrespective which candidates we add.

As a final conclusion, we have that the only candidate that can possibly defeat d is p, and this happens only if all members of X_{cand} join the election. It is possible to ensure that this happens by adding at most h bundles of candidates if and only if there is a set cover for I of size at most h.

We use the same construction for the case of Copeland$^\alpha$-CCAC, except that now p is the preferred candidate and we increase the size of D by one. $\square$

Theorem 6.13. Maximin-Comb-CCAC is NP-hard, even for elections with only six voters.

Proof. Let the notation be as in the introduction to Section 6. Given an instance $I := (X, \mathcal{S}, h)$ for Set Cover with $n' := |X_{\text{cand}}|$, we construct an instance for Maximin-Comb-CCAC. We let the set of registered candidates be $C := \{p\} \cup D$, where p is the preferred candidate and where $D := \{d_1, \ldots, d_{n'}\}$ is a set of dummy candidates. The unregistered candidate set is $A := X_{\text{cand}} \cup \mathcal{S}_{\text{cand}}$. We construct six voters with the following preference orders:

$$\begin{aligned}
&v_1\colon p \succ x_1 \succ d_1 \succ \cdots \succ x_{n'} \succ d_{n'} \succ \mathcal{S}_{\text{cand}},\\
&v_2\colon p \succ x_{n'} \succ d_{n'} \succ \cdots \succ x_1 \succ d_1 \succ \mathcal{S}_{\text{cand}},\\
&v_3\colon x_1 \succ \cdots \succ x_{n'} \succ d_1 \succ \cdots \succ d_{n'} \succ p \succ \mathcal{S}_{\text{cand}},\\
&v_4\colon d_{n'} \succ \cdots \succ d_1 \succ p \succ x_{n'} \succ \cdots \succ x_1 \succ \mathcal{S}_{\text{cand}},\\
&v_5\colon x_1 \succ \cdots \succ x_{n'} \succ d_1 \succ \cdots \succ d_{n'} \succ p \succ \mathcal{S}_{\text{cand}}, \text{ and}\\
&v_6\colon d_{n'} \succ \cdots \succ d_1 \succ p \succ x_{n'} \succ \cdots \succ x_1 \succ \mathcal{S}_{\text{cand}}.
\end{aligned}$$

(Note that v_3 and v_5 have the same preference order and that v_4 and v_6 have the same preference order.) We use the set-embedding bundling function. We claim that I is a "yes"-instance of Set Cover if and only if it is possible to ensure p's victory by adding at most h (bundles of) candidates.

Prior to adding any of the candidates, p has two points and each candidate in D has three points. All the voters rank the members of $\mathcal{S}_{\text{cand}}$ last, so the presence of these candidates in the election does not change the scores of p and members of D. More so, members of $\mathcal{S}_{\text{cand}}$ themselves receive zero points each. If some candidate x_i appears in the election, however, then we have the following effects:

1. This candidate's score is at most two (since only voters v_3 and v_5 prefer x_i to p).

2. The score of d_i becomes at most two (since only voters v_4 and v_6 prefer d_i to x_i).

3. The score of p does not change (since already v_1 and v_2 prefer p to x_i).

This means that if there is a set cover of size at most h for I, then adding the set candidates that correspond to this cover will bring all members of X_{cand} to the election and p will be among the winners.

First, note that it is impossible to increase the score of p by adding candidates, and that for each d_i, the only way to decrease its score to at most two is to bring x_i into the election.

For the reverse direction, notice that in order to let p win, we must add candidates to the election to decrease the score of every element candidate x_i, and the only way to achieve this by adding at most k bundles is by adding the s_j corresponding to the set cover. This means that if it is possible to ensure p's victory by adding at most h candidates, then it must be possible to add all members of X_{cand} into the election, and this means that there is a set cover of size at most h. ☐

Appendix D. Deferred Proofs for the Signature Proof Technique

Theorem 7.4. For each fixed integer $t \geq 2$, t-Approval-DCAC can be solved in time $\min\{O(m \cdot (t \cdot 3^n)^{t \cdot n}), O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})\}$, where m is the total number of candidates and n is the number of voters.

Proof. There are two means of solving our problem. We can either run the brute-force algorithm on top of Theorem 7.2, obtaining running time $O(m \cdot (t \cdot 3^n)^{t \cdot n}))$, or we can use a variant of the algorithm from Theorem 7.3. Below we describe how to adapt the algorithm from Theorem 7.3, as it allows us to achieve a better running time.

We use the same algorithm as in Theorem 7.3, but with a more involved notion of a signature and with a more involved merging operator $\oplus$. Indeed, the algorithm to be described next is a generalization of the algorithm described in Theorem 7.3.

So, if we have n voters, then we define the *unbounded* signature of a set A' of unregistered candidates to be an vector $\vec{\tau}$ with n entries, such that the ith entry is a vector τ_i with t values, defined as follows. The jth entry of τ_i, for $1 \leq j \leq t$ and $1 \leq i \leq n$, contains the number of candidates in A' that the ith voter prefers to all but $j-1$ registered candidates. Now a *bounded* signature (simply, a signature) of a set A' is its unbounded signature where all entries greater than t are replaced by t. Altogether, there are $(t+1)^{t \cdot n}$ signatures.

Given two signatures, $\vec{\tau}'$ and $\vec{\tau}''$, we define their merge, $\vec{\tau} = \vec{\tau}' \oplus \vec{\tau}''$, as follows: For each i, $1 \leq i \leq n$, vector τ_i is computed by first calculating the component-wise sum of vectors $\vec{\tau}'$ and $\vec{\tau}''$, and then replacing with t each entry greater than t. Now, if A' and A'' are two disjoint sets of candidates with signatures $\vec{\tau}_{A'}$ and $\vec{\tau}_{A''}$, then $\vec{\tau}_{A'} \oplus \vec{\tau}_{A''}$ is a signature of their union. (Note that in our algorithm we apply operator $\oplus$ to "signatures of disjoint sets of candidates" only.)

It is straightforward to verify that given a signature of a subset A' of unregistered candidates, we can compute the scores of candidates p and d. This suffices to describe our algorithm and to justify its correctness. The running time is $O(m \cdot n \cdot t \cdot (t+1)^{t \cdot n})$ (it is calculated in the same way as in the proof of Theorem 7.3, except that now we have more signatures and the components of the signatures are t-dimensional vectors). ☐

An example for the algorithm described above, for t-Approval-DCAC, is provided next.

Example 3. *Consider the following election.*

$$v_1 : \mathbf{d} \succ c \succ a \succ \mathbf{e} \succ b \succ \mathbf{p}$$
$$v_2 : b \succ c \succ \mathbf{p} \succ \mathbf{d} \succ \mathbf{e} \succ a$$
$$v_3 : a \succ c \succ \mathbf{d} \succ \mathbf{p} \succ b \succ \mathbf{e}$$

The registered candidates are $\{d, p, e\}$ and the unregistered candidates are $\{a, b, c\}$. We consider 2-Approval (that is, $t = 2$), therefore d gets 3 points, p gets 2 points, and e gets 1 point.

We have the following signatures.

$$a\text{'s signature: } \begin{pmatrix} 0 & 1 \\ 0 & 0 \\ 1 & 0 \end{pmatrix} \quad b\text{'s signature: } \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{pmatrix} \quad c\text{'s signature: } \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 0 \end{pmatrix}$$

Combining a and c together, we have the following signature.

$$a\text{'s signature } \oplus c\text{'s signature: } \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 0 \end{pmatrix}$$

Indeed, it can be computed from the above signature, that $\{a, c\}$ is indeed a solution, causing p to win the election.

Theorem 7.7. For each fixed integer $t \geq 1$, both t-Approval-DCDC and t-Veto-DCDC can be solved in time $O^*(m \cdot 4^n \cdot (3^n)^{3^n})$, where m is the total number of candidates and n is the number of voters.

We first describe our approach to proving the theorem and give the formal proof below. Let us fix a positive integer t and let $((C, V), d, k)$ be an instance of t-Approval-DCDC, where $V = (v_1, \ldots, v_n)$. We focus on the case of t-Approval and later we will argue how to adapt the results to apply to the case of t-Veto. We guess a candidate p, whose role is to defeat the despised candidate d. For each such candidate p we do the following. First, we make an initial brute-force search: For each voter, we "guess" one of at most four possible choices of how d and p would gain points after our action of deleting candidates:

1. choice one: only d receives one point,

2. choice two: only p receives one point,

ALGORITHM 1: FPT algorithm for t-Approval-DCDC.

Input:
$((C, V), d, k)$ — input: an instance of t-Approval-DCDC
p — a guessed candidate who is to defeat d

foreach $\vec{\delta} = (\delta_1, \delta_2, \ldots, \delta_n) \in [4]^n$ **with** $|\{i \mid \delta_i = 1\}| < |\{i \mid \delta_i = 2\}|$ **do**
— Run ILP for each sane $\vec{\delta}$ such that p beats d.
  foreach $i \in [n]$ **do**
    if `SanityCheck` (δ_i) = **false then**
      Next $\vec{\delta}$;
  if *p has more points than d when p and d receive points as described by $\vec{\delta}$* **and** *there is a solution for* `ILP` $(\vec{\delta})$ **then**
    accept;
reject;

`SanityCheck`(δ_i)
  if $\delta_i = 1$ **and** $(v_i : p \succ d)$ **then**
  — $\delta_i = 1$: only d gains one point.
    return false;
  if $\delta_i = 2$ **and** $(v_i : d \succ p)$ **then**
  — $\delta_i = 2$: only p gains one point.
    return false;
  return true;

`ILP`$(\vec{\delta} = (\delta_1, \delta_2, \ldots, \delta_n))$:
  Variables
  $\forall \vec{\gamma} \in [3]^n : x_{\vec{\gamma}}$ — # *deleted* candidates with $\{d, p\}$-signature $\vec{\gamma}$
  Constants
  $\forall \vec{\gamma} \in [3]^n : z_{\vec{\gamma}}$ — # *existing* candidates with $\{d, p\}$-signature $\vec{\gamma}$
  Constraints
    $\sum_{\vec{\gamma}} x_{\vec{\gamma}} \leq k$ $\forall \vec{\gamma} \in [3]^n : x_{\vec{\gamma}} \leq z_{\vec{\gamma}}$
    $\forall i \in [n]$:
    if $\delta_i = 1$ **or** $\delta_i = 2$ **then**
    — $v_i : d \succ p$ **and** only d gains one point, or
    — $v_i : p \succ d$ **and** only p gains one point
      $\sum_{\forall \vec{\gamma} : \gamma_i = 3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \leq t - 1$
      $\sum_{\forall \vec{\gamma} : \gamma_i = 3 \vee \gamma_i = 2} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t - 1$
    else if $\delta_i = 3$ **then**
    — Both d and p gain one point each
      $\sum_{\forall \vec{\gamma} : \gamma_i = 3 \vee \gamma_i = 2} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) + 2 \leq t$
    else
    — No one gains one point
      $\sum_{\forall \vec{\gamma} : \gamma_i = 3} (z_{\vec{\gamma}} - x_{\vec{\gamma}}) \geq t$

3. choice three: both candidates receive one point, and

4. choice four: neither p nor d receive a point.

We record our guesses in vector $\vec{\delta}$. For each guessed p and $\vec{\delta}$, we check whether giving the points according to our guesses in $\vec{\delta}$ guarantees that p has more points than d. If so, then we run an integer linear program to verify if it is at all possible to ensure that every voter gives points to candidates p and d as described by vector $\vec{\delta}$, and to compute the smallest number of candidates we have to delete to ensure this. The complete procedure, for the case of t-Approval-DCDC, is given as Algorithm 1.

Proof of Theorem 7.7. We start by considering the case of t-Approval-DCDC. The running time for Algorithm 1 is easy to verify: we guess a candidate p and a possible way of giving p and d points, followed by running an ILP. Therefore, the running time is $O(m \cdot 4^n)$ times the cost of running the ILP. The ILP has 3^n variables and $(3^n + 2n)$ constraints. Thus, employing the famous result of Lenstra, Jr. (1983), our algorithm runs in $O^*(4^n \cdot f(n))$ where f is a function that describes the running time of the ILP solver and solely depends on n (Kannan, 1987; Lenstra, Jr., 1983).

To prove the correctness of the algorithm, it suffices to show the correctness of the ILP program for a given guess of p and $\vec{\delta}$. First, the constraint in Line 23 ensures that we do not delete more candidates with a given $\{d, p\}$-signature $\vec{\gamma}$ than there are present in the election (c.f. the meaning of a signature in Definition 5). The remaining signatures verify that we can implement vector $\vec{\delta}$. For each i, $1 \le i \le n$, we verify whether it is possible to implement guess δ_i:

1. If $\delta_i = 1$ (that is, d gains a point from the ith voter but p does not), then according to our sanity check (`SanityCheck`) we have that v_i prefers d over p. Thus, after the candidate deletion, d must be ranked in the first t positions (Line 28) and p must be ranked behind the t'th position (Line 29).

2. If $\delta_i = 2$ which means that only p gains one point, then v_i prefers p over d. Thus, after the candidate deletion, p must be ranked in the first t positions (Line 28) and d must be ranked behind the t'th position (Line 29).

3. If $\delta_i = 3$, then both candidates gain one point each and must be ranked in the first t positions (Line 32) after the candidate deletion.

4. Otherwise, both gain zero points and must be ranked behind the t'th position (Line 35) after the candidate deletion.

This justifies the correctness of the ILP and completes the proof for the case of t-Approval.

For the case of t-Veto, it suffices to use the same approach as for t-Approval, provided that we first reverse all preference orders and consider that a candidate is a winner if this candidate's score is the lowest (in essence, this is equivalent to replacing "points" with "vetoes" in the above reasoning). ∎

References

Arrow, K., Sen, A., & Suzumura, K. (Eds.). (2002). *Handbook of Social Choice and Welfare, Volume 1*. Elsevier.

Bartholdi III, J., Tovey, C., & Trick, M. (1992). How hard is it to control an election?. *Mathematical and Computer Modeling*, *16*(8/9), 27–40.

Betzler, N., Bredereck, R., Chen, J., & Niedermeier, R. (2012). Studies in computational aspects of voting—a parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, Vol. 7370 of *LNCS*, pp. 318–363. Springer-Verlag.

Betzler, N., Guo, J., & Niedermeier, R. (2010). Parameterized computational complexity of Dodgson and Young elections. *Information and Computation*, *208*(2), 165–177.

Betzler, N., Niedermeier, R., & Woeginger, G. (2011). Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, pp. 55–60.

Betzler, N., Slinko, A., & Uhlmann, J. (2013). On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, *47*, 475–519.

Betzler, N., & Uhlmann, J. (2009). Parameterized complexity of candidate control in elections and related digraph problems. *Theoretical Computer Science*, *410*(52), 43–53.

Binkele-Raible, D., Fernau, H., Fomin, F. V., Lokshtanov, D., Saurabh, S., & Villanger, Y. (2012). Kernel(s) for problems with no kernel: On out-trees with many leaves. *ACM Transactions on Algorithms*, *8*(4), 38.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, *21*, 135–191.

Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. (Eds.). (2016). *Handbook of Computational Social Choice*. Cambridge University Press.

Brandt, F., Harrenstein, P., Kardel, K., & Seedig, H. G. (2013). It only takes a few: On the hardness of voting with a constant number of agents. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '13)*, pp. 375–382. IFAAMAS.

Brandt, F., Brill, M., Hemaspaandra, E., & Hemaspaandra, L. (2015). Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *Journal of Artificial Intellgence Research*, *53*, 439–496.

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

Bredereck, R., Faliszewski, P., Niedermeier, R., Skowron, P., & Talmon, N. (2015). Elections with few candidates: Prices, weights, and covering problems. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT '15)*, Vol. 9346 of *LNCS*, pp. 414–431. Springer-Verlag.

Bredereck, R., Faliszewski, P., Niedermeier, R., & Talmon, N. (2016a). Complexity of shift bribery in committee elections. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*, pp. 2452–2458. AAAI Press.

Bredereck, R., Faliszewski, P., Niedermeier, R., & Talmon, N. (2016b). Large-scale election campaigns: Combinatorial shift bribery. *Journal of Artificial Intelligence Research*, *55*, 603–652.

Bredereck, R., Chen, J., Faliszewski, P., Guo, J., Niedermeier, R., & Woeginger, G. J. (2014). Parameterized algorithmics for computational social choice: Nine research challenges. *Tsinghua Science and Technology*, *19*(4), 358–373.

Bulteau, L., Chen, J., Faliszewski, P., Niedermeier, R., & Talmon, N. (2015). Combinatorial voter control in elections. *Theoretical Computer Science*, *589*, 99–120.

Chen, J., Faliszewski, P., Niedermeier, R., & Talmon, N. (2015). Elections with few voters: Candidate control can be easy. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI '15)*, pp. 2045–2051. AAAI Press.

Conitzer, V., Lang, J., & Xia, L. (2009). How hard is it to control sequential elections via the agenda?. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pp. 103–108. AAAI Press.

Conitzer, V., Sandholm, T., & Lang, J. (2007). When are elections with few candidates hard to manipulate?. *Journal of the ACM*, *54*(3), 1–33.

Cygan, M., Fomin, F. V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., & Saurabh, S. (2015). *Parameterized Algorithms*. Springer-Verlag.

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

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

Dutta, B., Jackson, M., & Le Breton, M. (2001). Strategic candidacy and voting procedures. *Econometrica*, *69(4)*, 1013–1037.

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the Web. In *Proceedings of World Wide Web Conference (WWW-2001)*, pp. 613–622. ACM Press.

Elkind, E., Faliszewski, P., & Slinko, A. (2011). Cloning in elections: Finding the possible winners. *Journal of Artificial Intelligence Research*, *42*, 529–573.

Erdélyi, G., Fellows, M., Rothe, J., & Schend, L. (2015a). Control complexity in Bucklin and fallback voting: A theoretical analysis. *Journal of Computer and System Sciences*, *81*(4), 632–660.

Erdélyi, G., Fellows, M., Rothe, J., & Schend, L. (2015b). Control complexity in Bucklin and fallback voting: An experimental analysis. *Journal of Computer and System Sciences*, *81*(4), 661–670.

Erdélyi, G., Nowak, M., & Rothe, J. (2009). Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly*, *55*(4), 425–443.

Erdélyi, G., Hemaspaandra, E., & Hemaspaandra, L. (2015). More natural models of electoral control by partition. In *Proceedings of the 4th International Conference on Algorithmic Decision Theory (ADT '15)*, Vol. 9346 of *LNCS*, pp. 396–413. Springer-Verlag.

Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2010). Using complexity to protect elections. *Communications of the ACM*, *53*(11), 74–82.

Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2011). Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, *40*, 305–351.

Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, *35*, 275–341.

Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2011). The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. *Information and Computation*, *209*(2), 89–107.

Faliszewski, P., Skowron, P., Slinko, A., & Talmon, N. (2016). Multiwinner analogues of the plurality rule: Axiomatic and algorithmic views. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI '16)*, pp. 482–488.

Faliszewski, P., & Rothe, J. (2016). Control and bribery in voting. In Brandt, F., Conitzer, V., Endriss, U., Lang, J., & Procaccia, A. D. (Eds.), *Handbook of Computational Social Choice*, chap. 7, pp. 146–168. Cambridge University Press.

Fellows, M., Hermelin, D., Rosamond, F., & Vialette, S. (2009). On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, *410*(1), 53–61.

Flum, J., & Grohe, M. (2006). *Parameterized Complexity Theory*. Springer-Verlag.

Garey, M., & Johnson, D. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Garey, M. R., Johnson, D. S., & Stockmeyer, L. J. (1976). Some simplified NP-complete graph problems. *Theoretical Computer Science*, *1*(3), 237–267.

Hemaspaandra, E., & Hemaspaandra, L. (2007). Dichotomy for voting systems. *Journal of Computer and System Sciences*, *73*(1), 73–83.

Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2007). Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, *171*(5–6), 255–285.

Hemaspaandra, L., Lavaee, R., & Menton, C. (2016). Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. *Annals of Mathematics and Artificial Intelligence*, *77*(3-4), 191–223.

Kannan, R. (1987). Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, *12*(3), 415–440.

Kellerhals, L., Korenwein, V., Zschoche, P., Bredereck, R., & Chen, J. (2017). On the computational complexity of variants of combinatorial voter control in elections. In *Proceedings of the 14th Annual Conference on Computation and Logic: Theory and*

Applications of Models of Computation (TAMC '17), Vol. 10185 of *LNCS*, pp. 348–361. Springer-Verlag.

Koutecky, M., Knop, D., & Mnich, M. (2017). Voting and bribing in single-exponential time. In *Proceedings of the 34th Annual Symposium on Theoretical Aspects of Computer Science (STACS '17)*, Vol. 66, pp. 46:1–46:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

Lang, J., Maudet, N., & Polukarov, M. (2013). New results on equilibria in strategic candidacy. In *Proceedings of the 6th International Symposium on Algorithmic Game Theory (SAGT '13)*, Vol. 8146 of *LNCS*, pp. 13–25. Springer-Verlag.

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

Lin, A. (2012). *Solving Hard Problems in Election Systems*. Ph.D. thesis, Rochester Institute of Technology.

Liu, H., Feng, H., Zhu, D., & Luan, J. (2009). Parameterized computational complexity of control problems in voting systems. *Theoretical Computer Science*, *410*(27–29), 2746–2753.

Liu, H., & Zhu, D. (2010). Parameterized complexity of control problems in Maximin election. *Information Processing Letters*, *110*(10), 383–388.

Loreggia, A., Narodytska, N., Rossi, F., Venable, K., & Walsh, T. (2014). Controlling elections by replacing candidates: Theoretical and experimental results. In *Proceedings of the 8th Multidisciplinary Workshop on Advances in Preference Handling*, pp. 61–66. AAAI Press.

Magiera, K., & Faliszewski, P. (2014). How hard is control in single-crossing elections?. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI '14)*, pp. 579–584. IOS Press.

Mattei, N., Pini, M. S., Rossi, F., & Venable, K. B. (2013). Bribery in voting with CP-nets. *Annals of Mathematics and Artificial Intelligence*, *68*(1-3), 135–160.

Maushagen, C., & Rothe, J. (2016). Complexity of control by partitioning veto and maximin elections and of control by adding candidates to plurality elections. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI '16)*, pp. 277–285. IOS Press.

Meir, R., Procaccia, A., Rosenschein, J., & Zohar, A. (2008). The complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research*, *33*, 149–178.

Menton, C. (2013). Normalized range voting broadly resists control. *Theory of Computing Systems*, *53*(4), 507–531.

Menton, C., & Singh, P. (2013). Control complexity of Schulze voting. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*, pp. 286–292. AAAI Press.

Misra, J., & Gries, D. (1992). A constructive proof of Vizing's theorem. *Information Processing Letters*, *41*(3), 131–133.

Misra, N., Nabeel, A., & Singh, H. (2015). On the parameterized complexity of minimax approval voting. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pp. 97–105. IFAAMAS.

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

Obraztsova, S., Elkind, E., Polukarov, M., & Rabinovich, Z. (2015). Strategic candidacy games with lazy candidates. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, pp. 610–616. AAAI Press.

Parkes, D., & Xia, L. (2012). A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI '12)*, pp. 1429–1435. AAAI Press.

Polukarov, M., Obraztsova, S., Rabinovich, Z., Kruglyi, A., & Jennings, N. (2015). Convergence to equilibria in strategic candidacy. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI '15)*, pp. 624–630. AAAI Press.

Rothe, J., & Schend, L. (2013). Challenges to complexity shields that are supposed to protect elections against manipulation and control: a survey. *Annals of Mathematics and Artificial Intelligence*, *68*(1–3), 161–193.

Schäfer, A., Komusiewicz, C., Moser, H., & Niedermeier, R. (2012). Parameterized computational complexity of finding small-diameter subgraphs. *Optimization Letters*, *6*(5), 883–891.

Vizing, V. G. (1965). Critical graphs with a given chromatic class. *Metody Diskretnogo Analiza*, *5*(1), 9–17.

Xia, L., & Conitzer, V. (2010). Strategy-proof voting rules over multi-issue domains with restricted preferences. In *Proceedings of the 6th International Workshop on Internet and Network Economics (WINE '10)*, Vol. 6484 of *LNCS*, pp. 402–414. Springer-Verlag.

Yang, Y. (2014). Election attacks with few candidates. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI '14)*, pp. 1131–1132. IOS Press.