Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Elections with Few Voters: Candidate Control Can Be Easy*

Jiehua Chen[1] and **Piotr Faliszewski**[2] and **Rolf Niedermeier**[1] and **Nimrod Talmon**[1]
[1]TU Berlin, Berlin, Germany
{jiehua.chen, rolf.niedermeier}@tu-berlin.de, nimrodtalmon77@gmail.com
[2]AGH University of Science and Technology, Krakow, Poland
faliszew@agh.edu.pl

Abstract

We study the computational complexity of candidate control in elections with few voters (that is, we take the number of voters as a parameter). We consider both the standard scenario of adding and deleting candidates, where one asks if a given candidate can become a winner (or, in the destructive case, can be precluded from winning) by adding/deleting some candidates, and a combinatorial scenario where adding/deleting a candidate automatically means adding/deleting a whole group of candidates. 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.

Introduction

Election control problems model the issue of affecting the result of an election by either introducing some new candidates/voters or by removing some of them from the election. We study the complexity of election control by adding and deleting candidates, for the case where the election involves a few voters only. We focus on very simple, practical voting rules such as Plurality, Veto, and t-Approval, but we also discuss some more involved ones. To analyze the effect of a small number of voters, we use the formal tools of parameterized complexity theory.

From the point of view of classical complexity theory, candidate control is NP-hard for almost all typically studied voting rules (even for the Plurality rule; though some natural examples of polynomial-time candidate control problems exist as well). It turns out that for the case of elections with few voters (i.e., for control problems parameterized by the number of voters), the landscape of the complexity of candidate control is quite varied and, indeed, sometimes quite surprising (see Table 1 for an overview of our results). In addition to the standard candidate control problems, we also study their *combinatorial* variants, where it is possible to add or delete whole groups of candidates at unit cost. In this we follow the path initiated by Chen et al. (2014), who introduced combinatorial voter control.

Motivation. There is a number of settings in which it is most natural to consider elections with few voters (and, typically, many candidates). Let us look at several examples.

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 position.

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 the possible options provided by the travel offices, hotels, airlines, etc. This example is particularly relevant to the case of multi-agent systems: One may foresee that in the future we will delegate the task of finding the most satisfying holiday location to our personal software agents that will negotiate with travel offices and other travelers on our behalf.

Meta-search engine. Dwork et al. (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 before we actually hold an election, the voters (or, some particular individual) first shrink the set of candidates. In the case of the hiring committee, most of the applications are removed from the considerations early in the evaluation process. The people planning holidays first, implicitly, remove most of possible holiday options and, then, remove those candidates that do not fit their preferences completely (e.g., too expensive offers). 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 one 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

*PF was supported by the DFG project PAWS (NI 369/10) and by AGH University grant 11.11.230.124 (statutory research). NT was supported by the DFG Research Training Group "Methods for Discrete Structures" (GRK 1408). This work has been partly supported by COST Action IC1205 on Computational Social Choice.

Problem	Plurality	Veto	t-Approval	t-Veto	Borda	Copeland$^\alpha$	Maximin
$\mathcal{R}$-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	para-NP-h (10)	para-NP-h (20) ♠	para-NP-h (10)
$\mathcal{R}$-CCDC	FPT	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	para-NP-h (10)	para-NP-h (26) ♠	P ♣
$\mathcal{R}$-DCAC	FPT	FPT	FPT	FPT	P ♡	P ◇	P ♣
$\mathcal{R}$-DCDC	FPT	FPT	FPT	FPT	P ♡	P ◇	P ♣
$\mathcal{R}$-COMB-CCAC	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	W[1]-h / XP	para-NP-h (2)	para-NP-h (3) ♠	para-NP-h (6)
$\mathcal{R}$-COMB-CCDC	para-NP-h (1)	para-NP-h (1)	para-NP-h (1)	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	para-NP-h (2)	para-NP-h (3)	P
$\mathcal{R}$-COMB-DCDC	para-NP-h (3)	para-NP-h (1)	para-NP-h (2)	para-NP-h (1)	para-NP-h (2)	para-NP-h (3)	para-NP-h (5)

Table 1: The complexity of candidate control (constructive (CC) and destructive (DC), adding candidates (AC) and deleting candidates (DC)) problems for varying 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 $0 \leq \alpha \leq 1$; notice that the results by Betzler and Uhlmann (2009) hold only for $\alpha \in \{0, 1\}$). Results marked with ♣ and ◇ are due to Faliszewski et al. (2011; 2009), 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. Question mark (?) means that the exact complexity is still open.

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, we believe that it is natural and relevant to study the 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.

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

Main contributions. Our research has shown some surprising patterns that were not (nearly as) visible in the context of classical complexity analysis of election control:

1. (Non-combinatorial) destructive candidate control is easy (either in the fixed-parameter tractability sense or via outright polynomial-time algorithms) for all our voting rules.
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 (this is especially interesting since there is no such difference for the adding candidates case).

Our results (see Table 1; formal definitions follow in the next section) are of four types (with the exception of t-Veto-Comb-DCAC which is only in XP): for each of our problems we show that it either is in P, is in FPT, is W[1]-hard but has an XP-algorithm, or is para-NP-hard (in each case the parameter is the number of voters). Naturally, the first type of results is the most positive[1] (unconditionally efficient algorithms) and the second type is quite positive too (the exponential part of the running time of an algorithm depends only on the number of voters). The third kind is less positive (W[1]-hardness precludes existence of FPT algorithms, but membership in XP means that there are algorithms that are polynomial-time if the number of voters is a constant). The last kind is the most negative (NP-hardness even for a constant number of voters; this precludes membership in XP).[2] We introduce several new proof techniques to establish our results. Due to the lack of space, we can only sketch some of our proofs.

Related Work. The complexity study of election control was introduced by Bartholdi et al. (1992), who were later followed by numerous researchers, including, e.g., Hemaspaandra et al. (2007), Meir et al. (2008), and many others (we point the reader to the survey of Faliszewski et al. (2010) and to several recent papers on the topic (Parkes and Xia 2012; Erdélyi et al. 2012; Rothe and Schend 2013)). Briefly put, it turns out that for standard voting rules, control problems are typically NP-hard.

There is a growing body of research regarding the parameterized complexity of voting problems (see, e.g., the survey of Betzler et al. (2012)), where typical parameters include the solution size (e.g., the number of candidates that can be added) and the election size (i.e., the number of candidates or the number of voters). For the solution size as the parameter, control problems usually turn out to be hard (Betzler and Uhlmann 2009; Liu et al. 2009; Liu and Zhu 2010). On the contrary, taking the number of candidates as the parameter almost always leads to FPT (fixed-parameter tractability) results (see, e.g., the papers of Faliszewski et al. (2011) and Hemaspaandra et al. (2013)).

[1] We note that we evaluate the results from the computational complexity perspective and, hence, regard computational efficiency as positive.

[2] Naturally, we use the standard complexity-theoretic assumptions that P $\neq$ NP and FPT $\neq$ W[1].

However, so far, 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 number of voters. The parameter "number of voters" also received some limited attention in other voting settings (Betzler et al. 2010; Dorn and Schlotter 2012; Bredereck et al. 2014).

The study of combinatorial control was recently initiated by Chen et al. (2014), who focused on voter control. We stress that our combinatorial view of control is different from the studies of combinatorial voting domains (Boutilier et al. 2004; Xia and Conitzer 2010; Mattei et al. 2012).

Preliminaries

Elections. An election $E = (C, V)$ consists of a set of candidates $C = \{c_1, \dots, c_m\}$ and a collection $V = (v_1, \dots, v_n)$ of voters. Each voter v_ℓ has a preference order (vote), often denoted $\succ_\ell$, which ranks the candidates from the one that v_ℓ likes most to the one that v_ℓ likes least. We sometimes write $v_\ell \colon c_i \succ c_j$ to indicate that v_ℓ prefers c_i to c_j. If A is some subset of candidates, then writing A within a preference order description (e.g., $A \succ a \succ b$, where a and b are some candidates) means listing members of A in some arbitrary, but fixed, order. Writing $\overleftarrow{A}$ means listing the candidates in the reverse of this order. Given an election $E = (C, V)$, for each two candidates $c_i, c_j \in C$, we define $N_E(c_i, c_j) := \|\{v_\ell \mid v_\ell \colon c_i \succ c_j\}\|$.

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 (i.e., 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):

t-Approval and t-Veto. Under t-Approval (where $t \geq 1$ is an integer), each candidate gets a point for each voter that ranks him or her among the top t positions. For m candidates, t-Veto is a nickname for $(m-t)$-Approval (we often view the score of a candidate under t-Veto as the number of vetoes, i.e., the number of times he or she is ranked among bottom t positions). We refer to 1-Approval and 1-Veto as the Plurality rule and the Veto rule, respectively.

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 Borda, for each voter v, gives each candidate c as many points as the number of candidates that v ranks c ahead of.) Under Maximin, each candidate $c \in C$ receives $\min_{d \in C \setminus \{c\}} N_E(c, d)$ points.

Copeland$^\alpha$ rule. Under the Copeland$^\alpha$ rule (where α is rational, $0 \leq \alpha \leq 1$), in election $E = (C, V)$ each candidate c receives $\|\{d \in C \setminus \{c\} \mid N_E(c, d) > N_E(d, c)\}\| + \alpha\|\{d \in C \setminus \{c\} \mid N_E(c, d) = N_E(d, c)\}\|$ points.

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). Following the work of Chen et al. (2014), we also consider combinatorial variants of our problems, where adding/deleting a single candidate automatically adds/deletes a whole group of other candidates. In these *combinatorial variants* (denoted with a prefix Comb), we use bundling functions κ such that for each candidate c, $\kappa(c)$ is a set of candidates that are also added if c is added (or, that 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.[3] If B is some subset of candidates, by $\kappa(B)$ we mean $\bigcup_{c \in B} \kappa(c)$. Bundling functions are encoded by explicitly listing their values for all the arguments. Formally, given a voting rule $\mathcal{R}$, our problems are defined as follows.

$\mathcal{R}$-COMB-CCAC (resp. $\mathcal{R}$-COMB-CCDC)
Input: An election (C, V), a preferred candidate $p \in C$, a bundling function κ, and a non-negative integer k (in $\mathcal{R}$-COMB-CCAC we are also given a set A of unregistered candidates and the voters have preference orders over $C \cup A$).
Question: Is there a set $A' \subseteq A$ with $\|A'\| \leq k$ such that $p \in \mathcal{R}(C \cup \kappa(A'), V)$? (resp. is there a set $C' \subseteq C$ with $\|C'\| \leq k$ such that $p \in \mathcal{R}(C \setminus C', V)$?)

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 if 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 discussions.

Our model of combinatorial candidate control is about the simplest that one can think of. Indeed, in a scenario with m candidates, there are at most m corresponding bundles of candidates that can be added/deleted. In real life, one might expect many more. However, on the one hand, even such a simple model turns out to be computationally difficult and, on the other hand, we believe that it is instructive to consider such a simplified model first.

Parameterized Complexity. A parameterized problem is in 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 $\|I\|$; for us the parameter value is the number of voters involved) decides this instance in time $f(p) \cdot \|I\|^{O(1)}$, where f is some computable function. If, instead, the algorithm requires time $O(\|I\|^{f(p)})$, then the problem is in XP. Indeed, while the problems from both complexity classes come under "polynomial-time solvable when the parameter p is a constant", it is decisive that in the case of FPT the degree of the polynomial does not depend on p. Problems in FPT are viewed as tractable, whereas the class XP is rather considered to be at a high-level of the parameterized intractability hierarchy $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{XP}$. Due to space constraints, we omit the formal definition of the $\mathsf{W}[\cdot]$ hierar-

[3] Whenever we delete candidates from an election, these candidates are also implicitly deleted from the voters' preference orders.

chy (see, e.g., the textbooks (Downey and Fellows 2013; Flum and Grohe 2006; Niedermeier 2006)). To show W[1]-hardness one can, e.g., give a parameterized reduction from the W[1]-hard MULTI-COLORED CLIQUE problem. A parameterized reduction from a parameterized problem L to a parameterized problem L' is a function that, given an instance (I, p), computes in FPT time (wrt. p) an instance (I', p') such that $p' \leq g(p)$ (where g is an arbitrary function) and $(I, p) \in L \Leftrightarrow (I', p') \in L'$; in this paper all reductions can actually be computed in polynomial time.

If a problem is NP-hard via a reduction that produces instances where the value of the parameter is a constant, then we say that such a problem is para-NP-hard. W[1]-hardness precludes an FPT algorithm for the problem and para-NP-hardness precludes an XP algorithm.

Overview of Proof Techniques

We introduce several proof techniques that can be useful in studying the complexity of election problems parameterized by the number voters. We use the following techniques (the first two are, perhaps, most interesting):

Multi-Colored Clique Technique. This is a technique used for establishing W[1]-hardness results. The idea is to give a reduction from MULTI-COLORED CLIQUE (MCC) parameterized by the clique order (a variant of the standard CLIQUE problem, better suited for the parameterized complexity results, where each vertex has one of h colors and we seek a clique of order h with each vertex of a different color): Given an MCC-instance, we introduce a candidate for each vertex and two candidates for each edge, and—in essence—we have to ensure that we add only the candidates (delete all but the candidates) that correspond to a multi-colored clique. 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 to bound the number of voters needed for the construction. See Theorem 1 for a sample proof.

Cubic Vertex Cover Technique. This is a technique used for establishing para-NP-hardness results for non-combinatorial constructive candidate controls. The crucial idea of the technique is that the edges in a cubic graph can be partitioned into four disjoint matchings, which allows one to encode all the 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. A sample proof is given in Theorem 5.

Set-Embedding Technique. This is a very simple technique for showing para-NP-hardness results for combinatorial control by adding/deleting candidates. The idea is to reduce from the standard SET COVER problem using the bundling function to encode sets. Due to the power of bundling, a constant number of voters suffices for the reduction. A sample proof is given for Theorem 2.

Signature Technique. This is a group of two very similar techniques for showing FPT results (usually for destructive control). The first technique in the group works for control by adding candidates problems and relies on the fact that 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). The second technique applies to control by deleting candidates. A sample proof using the first technique is given in Theorem 1.

Approval-Based Rules

In this section, we consider t-Approval and t-Veto rules. These are perhaps the simplest and most frequently used rules, so results regarding them are of particular interest.

We start by looking at the Plurality rule and the Veto rule. In terms of standard complexity theory, control by adding/deleting candidates (constructive and destructive) is NP-complete for both of them (Bartholdi et al., (1992); Hemaspaandra et al., (2007)). However, if we parameterize by the number of voters, the results change quite drastically. On the one hand, the results for analogous types of (non-combinatorial) control for these rules differ (for example, Plurality-CCDC is in FPT but Veto-CCDC is W[1]-hard; this is quite unexpected given the similarity and simplicity of Plurality and Veto), and, on the other hand, combinatorial and non-combinatorial control problems behave differently. For example, in combinatorial control, the *deleting* candidates case is para-NP-hard for all the rules, but the *adding* candidates case is either in FPT or W[1]-hard (but in XP).

Theorem 1. *When parameterized by the number of voters, (1) for Plurality and Veto, DCAC and DCDC are both in* FPT*, (2) Plurality-CCAC and Veto-CCAC are both* W[1]*-hard, and (3) Plurality-CCDC is in* FPT*, while Veto-CCDC is* W[1]*-hard.*

Proof sketch for Plurality-DCAC. First, we guess a candidate p which is to defeat the despised candidate d (such a candidate must exist in a "yes"-instance; if p is an unregistered candidate, then we add it and decrease k by one).

Let $m := \|A\| + \|C\|$ be the total number of candidates and n be the number of voters. For each unregistered candidate a, we define its signature to be the collection of votes restricted to candidates p, d, and a, with each occurrence of a replaced by a global symbol x. Adding a single candidate with a given signature has the same effect on the score difference of d and p as adding several candidates with the same signature. Thus, we partition the set of unregistered candidates into equivalence classes based on their signatures, and, for each signature, remove all unregistered candidates but one. We also remove all the registered candidates that do not score any points in the original election. Altogether, we are left with at most n registered candidates and at most 3^n unregistered ones (the maximum number of different signatures). We solve this instance by brute-forcing all at-most-k-sized subsets of the unregistered candidates. This gives running time of the form $O(3^{n^2} \cdot \text{poly}(m, n))$ since $k \leq n$. Finally, we remark that by using exponential space we can design a more complicated $O(2^n \cdot m \cdot n)$-time algorithm for Plurality-DCAC. □

Proof sketch for Plurality-CCAC. We give a reduction from the W[1]-hard problem MULTI-COLORED CLIQUE parameterized by the clique order. In this problem, we are given an undirected graph $G = (V(G), E(G))$ whose vertices are partitioned into h disjoint sets, $V_1(G), \dots, V_h(G)$ such that for each i, $V_i(G)$ consists of exactly n' vertices with color i. We ask if there is an order-h clique containing a vertex for each color. We rename the vertices so that for each i, $1 \le i \le h$, we have $V_i(G) = \{v_1^{(i)}, \dots, v_{n'}^{(i)}\}$. W.l.o.g., we assume that G has edges between vertices of different colors only.

We construct a Plurality-CCAC instance as follows: The registered candidates are p (the preferred one) and d. We have one unregistered candidate v for each vertex v, and two unregistered candidates, (u, v), (v, u), for each edge $\{u, v\}$.

To describe the votes, we need the following notation. Let i and j be two distinct colors. Let $E(i, j)$ denote the set of all edge candidates (u, v), where $u \in V_i(G)$ and $v \in V_j(G)$. For each vertex $v_z^{(i)} \in V_i(G)$, let $L(v_z^{(i)}, j)$ denote the set of all edge candidates $(v_z^{(i)}, v)$, where $v \in V_j(G)$. Finally, let $R(i, j)$ and $R'(i, j)$ denote the following two orders (which, indeed, are the crucial part of our construction):

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

We construct a set V of $3h + 2(h+1) \cdot \binom{h}{2}$ voters as follows.

1. For each color i, $(1 \le i \le h)$, construct one voter with orders $v_1^{(i)} \succ \cdots \succ v_{n'}^{(i)} \succ d \succ \cdots$.
2. For each pair of colors i, j, $(1 \le i \ne j \le h)$, construct $h - 1$ voters with orders $E(i, j) \succ d \succ \cdots$, and another two voters, one with orders $R(i, j) \succ d \succ \cdots$ and one with orders $R'(i, j) \succ d \succ \cdots$.
3. Construct h voters with orders $d \succ \cdots$ and h voters with orders $p \succ \cdots$.

We claim that p can become a winner by adding at most $k := h + 2\binom{h}{2}$ candidates if and only if G has an order-h multi-colored clique (i.e., a clique containing a vertex for each color). Simple calculation shows that if Q is a multi-colored clique of order h, then adding the vertex candidates and the edge candidates corresponding to Q makes p win.

Conversely, we observe that irrespective of how many candidates we add to the election, p cannot have more than h points. Thus, d and every added unregistered candidate *cannot* have more than h points in the final election. This implies that any size-at-most-$(h + 2\binom{h}{2})$ set A' of unregistered candidates that we add to the election must contain exactly one vertex candidate for each color and exactly one edge candidate for each (ordered) pair of colors. Further, if A' contains two vertex candidates u, v but not the edge candidate (u, v), then, due to the orders $R(i, j) \succ d \succ \cdots$ and $R'(i, j) \succ d \succ \cdots$, either u or an edge candidate (u', v') (where $u' \in V_i(G)$, $v' \in V_j(G)$, but $(u', v') \ne (u, v)$) receives too many points, causing p not to win. To see why, note that $R(i, j)$ and $R'(i, j)$ contain all the candidates from $V_i(G)$ and $E(i, j)$. If we restrict those two preference orders to u and (u, v), then they will become $u \succ (u, v)$ and the reverse one $(u, v) \succ u$. However, if we restrict them to u and (u', v'), then either they will both be $u \succ (u', v')$ or they will both be $(u', v') \succ u$. This completes the proof. □

The Veto-CCAC case is quite intriguing. To see why, let us consider the following voting rule: Under *TrueVeto*, a candidate c is a winner if none of the voters ranks c last. It is quite easy to show that TrueVeto-CCAC is NP-complete, but it is also in FPT (when parameterized by the number of voters; an algorithm similar to that for Plurality-DCAC works). 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 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 few spoiler candidates to add from a large group of them. If we were adding more candidates than voters, the problem would be in FPT.

In the combinatorial setting, there is a sharp difference between control by adding and by deleting candidates.

Theorem 2. *When parameterized by the number of voters, for Plurality and Veto, (1)* COMB-*DCAC is in* FPT*, (2)* COMB-*CCAC is* W[1]*-hard, and (3)* COMB-*CCDC and* COMB-*DCDC are para-*NP*-hard.*

*Proof sketch for Plurality-*COMB-*DCDC.* We reduce from SET COVER which, given a ground set $X = \{x_1, \dots, x_{n'}\}$, a family $\mathcal{S} = \{S_1, \dots, S_{m'}\}$ of subsets of X, and a non-negative integer h (taken to be the parameter), asks whether it is possible to pick at most h sets from $\mathcal{S}$ so that their union is X. Given an instance I of SET COVER, we create an instance of Plurality-COMB-DCDC as follows. We let the candidate set be $C = \{p, d\} \cup X \cup \mathcal{S}$ (note that, depending on the context, we will use the symbol S_j, $1 \le j \le m'$, to denote both the set from $\mathcal{S}$ and a set-candidate in the election). We introduce three voters with the following preference orders:

$$v_1\colon X \succ p \succ \cdots, \quad v_2\colon d \succ \cdots, \quad \text{and} \quad v_3\colon p \succ d \succ \cdots.$$

We set the bundling function κ so that for each set-candidate S_j, we have $\kappa(S_j) := \{S_j\} \cup \{x_i \mid x_i \in S_j\}$, and for every non-set candidate c, we have $\kappa(c) := \{c\}$.

We claim that the candidate d can be precluded from winning by deleting at most h bundles of candidates if and only if there are h sets from $\mathcal{S}$ whose union is X.

Prior to deleting candidates, d, p, and one of the candidates from X are tied as winners. Deleting p would make d a unique winner, so the only way to defeat d is to ensure that v_1 gives its point to p. It is easy to see that we can assume that we only delete bundles of the set-candidates. To ensure that v_1 gives a point to p, all candidates from X must be deleted and, given our bundling function, this is possible (by deleting h bundles) if and only if the union of the sets corresponding to the deleted bundles is X. □

For t-Approval and t-Veto with $t \ge 2$, there are fewer surprises and patterns are more clearly visible: In the non-combinatorial setting, constructive controls are W[1]-hard

and the destructive ones are in FPT. In the combinatorial setting, we have mostly hardness results.

Theorem 3. *When parameterized by the number of voters, for each fixed integer $t \geq 2$, for t-Approval and t-Veto, (1) (*COMB*)-CCAC, and CCDC are* W[1]*-hard, (2) DCAC and DCDC are in* FPT*, (3)* COMB*-CCDC and* COMB*-DCDC are para-*NP*-hard, and (4) t-Approval-*COMB*-DCAC is* W[1]*-hard.*

We conclude our discussion by claiming that in each of the W[1]-hard cases discussed in this section we can, indeed, provide an XP algorithm. This means that these cases cannot be strengthened to para-NP-hardness results.

Theorem 4. *For each control type $\mathcal{K} \in \{$CCAC, CCDC,* COMB*-CCAC,* COMB*-DCAC$\}$, and for each fixed integer t, $t \geq 1$, each of t-Approval-$\mathcal{K}$ and t-Veto-$\mathcal{K}$ is in* XP*, when parameterized by the number of voters.*

Other Voting Rules

We focus on the voting rules Borda, Copeland$^\alpha$, and Maximin. The results are quite different from those for the case of t-Approval and t-Veto. Instead of FPT and W[1]-hardness results, we obtain 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 in Borda (Loreggia et al. 2014), Copeland$^\alpha$ (Faliszewski et al. 2009), and Maximin (Faliszewski et al., (2011)). For constructive candidate control, para-NP-hardness was already known for Copeland^0 and Copeland^1 (Betzler and Uhlmann 2009) and we establish it for the remaining values of α and for Borda and Maximin (in the latter case, only for CCAC; CCDC is known to be in P).

Theorem 5. *When parameterized by the number of voters, for Borda and Copeland$^\alpha$ ($0 \leq \alpha \leq 1$), CCAC and CCDC are para-*NP*-hard, and Maximin-CCAC is para-*NP*-hard.*

Proof sketch for Borda-CCDC. We reduce from the NP-complete problem CUBIC VERTEX COVER that given an undirected graph G, where each vertex has degree exactly three, and a non-negative integer h, asks whether 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.

Let $I = (G, h)$ be a CUBIC VERTEX COVER instance. We decompose $E(G)$ into four disjoint matchings (this is possible due to the computational variant of the classic graph-coloring result of Vizing (Misra and Gries 1992)) and rename the edges so that for each ℓ, $1 \leq \ell \leq 4$, the ℓ'th of these matchings is $E^{(\ell)} = \{e_1^{(\ell)}, \ldots, e_{m_\ell}^{(\ell)}\}$. We set $m' = m_1 + m_2 + m_3 + m_4 = \|E(G)\|$ and $n' = \|V(G)\|$. For each edge e, 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 $EV^{(-\ell)}$ to mean the set of edges not in $E^{(\ell)}$ union the set of vertices not incident to any of the edges in $E^{(\ell)}$. For each edge e, we define the following two orders over e, $v'(e)$, and $v''(e)$:

$$P(e)\colon e \succ v'(e) \succ v''(e) \text{ and } P'(e)\colon e \succ v''(e) \succ v'(e).$$

We form an election $E = (C, V)$, where $C = \{p, d\} \cup V(G) \cup E(G)$ and the voter set includes the following voters:

1. For each ℓ, $1 \leq \ell \leq 4$, we have the following two voters ($E^{(\ell)}$ is a matching so the orders are well-defined):

$$\mu(\ell)\colon P(e_1^{(\ell)}) \succ \cdots \succ P(e_{m_\ell}^{(\ell)}) \succ EV^{(-\ell)} \succ d \succ p, \text{ and}$$
$$\mu'(\ell)\colon p \succ d \succ \overleftarrow{EV^{(-\ell)}} \succ P'(e_{m_\ell}^{(\ell)}) \succ \cdots \succ P'(e_1^{(\ell)}).$$

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

We claim that deleting at most h candidates can make p a winner if and only if there is a vertex cover of size h for G.

Initially, we have the following scores (to calculate them, note that—except for small asymmetries—our pairs of votes are reverses of each other): p has $5(n'+m')+6$ points, d has $5(n'+m')+4$ points, each edge candidate has $5(n'+m')+7$ points, and each vertex candidate has $5(n'+m')+2$ points. So, p has one point fewer than each edge candidate, but more points than the other ones.

Consider the effects of deleting candidates. Deleting d decreases the score of p by six, whereas it decreases the scores of each other candidate by five (so it is never beneficial to delete d). Further, 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. Now, let v be some vertex candidate. If we delete v, the score of each edge candidate e such that $v = v'(e)$ or $v = v''(e)$ decreases by six, and the score of each other remaining candidate decreases by five. Thus, there is a vertex cover of size h if and only if deleting vertices corresponding to the cover ensures p's victory. □

For combinatorial variants of candidate control, we only have one polynomial-time algorithm (for Maximin-COMB-DCAC); all the remaining cases are para-NP-hard. Our proofs mostly rely on the set-embedding technique. In particular, we prove that for every voting rule $\mathcal{R}$ that satisfies the unanimity principle (that is, for each voting rule $\mathcal{R}$ that chooses as the unique winner the candidate that is ranked first by all the voters), $\mathcal{R}$-COMB-CCDC is para-NP-hard.

Theorem 6. *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.*

Altogether, we have the following result.

Theorem 7. *When parameterized by the number of voters, for Borda, Copeland$^\alpha$ ($0 \leq \alpha \leq 1$), and Maximin,* COMB*-$\mathcal{K}$ is para-*NP*-hard for each control type $\mathcal{K} \in \{$CCAC, CCDC, DCDC$\}$. For Borda and Copeland$^\alpha$ ($0 \leq \alpha \leq 1$),* COMB*-DCAC is para-*NP*-hard. On the contrary, Maximin-*COMB*-DCAC is polynomial-time solvable.*

In summary, for our more involved voting rules, constructive candidate control is hard even in the non-combinatorial setting, whereas destructive candidate control is tractable in the non-combinatorial setting, but becomes para-NP-hard in the combinatorial ones (with the exception of Maximin).

Outlook

Our work motivates several research directions. A particularly interesting one is to consider game-theoretic aspects of candidate control: Tractability results motivate studying more involved settings (e.g., consider a setting where two actors try to preclude two different candidates from winning; their goals might involve both cooperation and competition). Finally, taking a more general perspective, 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.

References

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

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

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

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

Boutilier, C.; Brafman, R.; Domshlak, C.; Hoos, H.; and 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.; Harrenstein, P.; Kardel, K.; and Seedig, H. G. 2013. It only takes a few: on the hardness of voting with a constant number of agents. In *Proceedings of AAMAS-2013*, 375–382.

Bredereck, R.; Chen, J.; Faliszewski, P.; Nichterlein, A.; and Niedermeier, R. 2014. Prices matter for the parameterized complexity of shift bribery. In *Proceedings of AAAI-2014*, 1398–1404.

Chen, J.; Faliszewski, P.; Niedermeier, R.; and Talmon, N. 2014. Combinatorial voter control in elections. In *Proceedings of MFCS-2014*, 153–164.

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

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

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceedings of WWW-2001*, 613–622.

Erdélyi, G.; Fellows, M.; Rothe, J.; and Schend, L. 2012. Control complexity in Bucklin and Fallback voting. Technical Report arXiv:1103.2230 [cs.CC].

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

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2011. Multimode control attacks on elections. *Journal of Artificial Intelligence Research* 40:305–351.

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

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

Hemaspaandra, L.; Lavaee, R.; and Menton, C. 2013. Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. In *Proceedings of AAMAS-2013*, 1345–1346.

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

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

Loreggia, A.; Narodytska, N.; Rossi, F.; Venable, K.; and Walsh, T. 2014. Controlling elections by replacing candidates: Theoretical and experimental results. In *Proceedings of MPREF-2014*, 61–66.

Mattei, N.; Pini, M.; Rossi, F.; and Venable, K. 2012. Bribery in voting over combinatorial domains is easy. In *Proceedings of AAMAS-2012*, 1407–1408.

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

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

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

Parkes, D., and Xia, L. 2012. A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In *Proceedings of AAAI-2012*, 1429–1435.

Rothe, J., and 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.

Xia, L., and Conitzer, V. 2010. Strategy-proof voting rules over multi-issue domains with restricted preferences. In *Proceedings of WINE-2010*, 402–414.