Ann Math Artif Intell (2016) 77:191–223
DOI 10.1007/s10472-015-9479-1

Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control

Lane A. Hemaspaandra[1] · **Rahman Lavaee**[1] · **Curtis Menton**[2]

Published online: 3 November 2015

Abstract Schulze and ranked-pairs elections have received much attention recently, and the former has quickly become a quite widely used election system. For many cases these systems have been proven resistant to bribery, control, or manipulation, with ranked pairs being particularly praised for being NP-hard for all three of those. Nonetheless, the present paper shows that with respect to the number of candidates, Schulze and ranked-pairs elections are fixed-parameter tractable to bribe, control, and manipulate: we obtain uniform, polynomial-time algorithms whose running times' degrees do not depend on the number of candidates. We also provide such algorithms for some weighted variants of these problems.

Keywords Elections · Computational social choice

Mathematics Subject Classification (2010) 91B12 · 91B14 · 91B10 · 68Q17

✉ Lane A. Hemaspaandra
lane@cs.rochester.edu
URL: www.cs.rochester.edu/u/lane

Rahman Lavaee
rlavaee@cs.rochester.edu

Curtis Menton
cgmenton@gmail.com

[1] Department of Computer Science, University of Rochester, Rochester, NY 14627, USA

[2] Google Inc., Mountain View, CA 94043, USA

1 Introduction

Schulze voting [34], though relatively recently proposed, has quickly been rather widely adopted. Designed in part to well-handle candidate cloning, its users include Wikipedia in French, Hebrew, and Russian, the Pirate Party in over a dozen countries, Debian, the Gentoo Foundation, KDE e.V., the Free Software Foundation Europe, Ubuntu, and dozens of other organizations, and in 2013 Wikipedia even asserted that "currently the Schulze method is the most widespread Condorcet method" [36].

Although the winner-choosing process in Schulze voting is a bit complicated to describe, involving minima, maxima, and comparisons of paths in the so-called weighted majority graph, Schulze [34] proved that finding who won a Schulze election nonetheless is polynomial-time computable, and Parkes and Xia [31] for the so-called destructive case and Gaspers et al. [16] for the so-called constructive case (extending a one-manipulator result for that case by Parkes and Xia [31]) proved that the (unweighted coalitional) manipulation problem for Schulze elections is polynomial-time computable. On the other hand, Parkes and Xia [31] proved that for Schulze elections bribery is NP-hard, and the work of Parkes and Xia [31] and Menton and Singh [28] established that for Schulze elections 15 of the 22 benchmark control attacks are NP-hard.

Parkes and Xia also note that, by the work of [31, 37, 38], the ranked-pairs election system, which is not widely popular but like Schulze has a polynomial-time winner-determination problem and like Schulze is based on the weighted majority graph, is resistant to (basically, NP-hard with respect to) bribery, control under each of the control types they study in their paper, and manipulation. Based on their discovery that ranked pairs is more broadly resistant to attacks than Schulze, the fact that Schulze itself "is in wide use," and the fact that there is "broad axiomatic support for both Schulze and ranked pairs," Parkes and Xia [31] quite reasonably conclude that "there seems to be good support to adopt ranked pairs in practical applications."

However, in this paper we show that the resistances-to-attack of Schulze and ranked pairs are both quite fragile.

For each of the *bribery* and *control* cases studied by Parkes and Xia, Menton and Singh, and Gaspers et al. for which they did not already prove Schulze voting to be in P, we prove that Schulze voting is fixed-parameter tractable with respect to the number of candidates. (The (unweighted) *manipulation* cases were already all put into P by these papers.) Fixed-parameter tractable (see [30]) means there is an algorithm for the problem whose running time is $f(j)I^{O(1)}$, where j is the number of candidates and I is the input's size. This of course implies that for each fixed number of candidates, the problems are in polynomial time, but it says much more; it implies that there is a global bound on the degree of the polynomial running time, regardless of what the fixed number of candidates is.

That result might lead one to even more strongly suggest the adoption of ranked pairs as an attractive alternative to Schulze. However, although for ranked pairs Parkes and Xia proved all the types of bribery, control, and manipulation they studied to be NP-hard, we show that every one of those cases is fixed-parameter tractable (with respect to the number of candidates) for ranked pairs. So even ranked pairs does not offer a safe haven from fixed-parameter tractability.

Our final results section looks at bribery and manipulation in the case of *weighted* voting, and establishes a number of results for that case. For example, for ranked pairs, we establish that weighted constructive coalitional manipulation is fixed-parameter tractable with respect to the combined parameter "number of candidates" and "cardinality of the manipulators' weight set." We give evidence that this fixed-parameter tractability result cannot be extended to a general P result, namely, we establish that weighted constructive coalitional

manipulation is NP-complete for five or more candidates. We also show that this "five" is optimal unless P = NP, by establishing that this problem is in polynomial time for four or fewer candidates.

The structure of the remainder of the paper is as follows. Section 2 presents the most important contribution of this paper: a winner-set certification framework that will work hand-in-hand with a "looping algorithm" to allow us to prove fixed-parameter tractability results for a wide range of manipulative-attack problems related to Schulze elections and ranked-pairs elections. Section 3 provides most of the definitions we need, including those of ranked-pairs elections and various types of manipulative-attack problems. Section 4 surveys related work. Section 5 is our most central results section, and presents most of our fixed-parameter tractability results for Schulze and ranked-pairs elections, obtained by looping over our winner-set certification frameworks. Sections 6 and 7 provide additional results for, respectively, the unweighted-voting and weighted-voting cases, mostly regarding alternate parameterizations and at what threshold the weighted constructive coalitional manipulation problem for ranked-pairs elections shifts from being in P to being NP-complete.

2 Presentation of the key idea

Our fixed-parameter tractability proofs are of interest in their own right, because they face a very specific challenge, which at first might not even seem possible to handle. The goal of this section is to describe in relatively high-level terms what that challenge is and how we handle it.

This section is organized as follows. Section 2.1 defines Schulze's election system, and gives an example to show how that systems works. Section 2.2 discusses why the "go-to" approach for showing fixed-parameter tractability for manipulative-attack problems seems to lack the flexibility to address Schulze and ranked-pairs elections. Finally, Section 2.3, which is the most important section of our paper, discusses our approach to obtaining fixed-parameter tractability results regarding manipulative attacks on these election systems. We present, in Section 2.3, a "winner-set certification framework" for Schulze elections (and later, in Section 5.1.2, for ranked-pairs elections) that will be extensively used in most of the rest of this paper to obtain fixed-parameter tractability results.

2.1 Schulze voting

We present the definition of Schulze voting [34]. Voters will always vote by linear orders over the candidates; in doing that, we adopt the complete, tie-free ordering version of Schulze voting that is used in the papers most related to this one [16, 28, 31]. We first define the important, standard notion of a weighted majority graph.

Definition 1 (**Weighted Majority Graph**) Given the input set of candidates and the set of votes over them (as linear orders), the weighted majority graph (WMG) is the graph that for each ordered pair of candidates c and d, $c \neq d$, has an edge from c to d having weight equal to the number of voters who prefer c to d minus the number of voters who prefer d to c.

Clearly, either all WMG edges have even weight or all WMG edges have odd weight, and the weight of the edge from c to d is negative one times the weight of the edge from d to c.

The notion of a path's strength underpins the definition of Schulze elections.

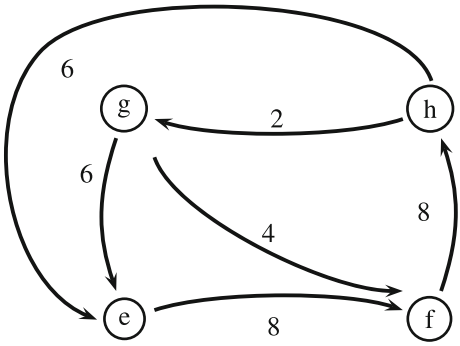

Fig. 1 WMG for the election examples, with one edge from each pair left implicit. Those edges are the reverse edges of the displayed edges, with a weight equal to negative one times the weight of their displayed counterpart, e.g., the implicit edge from candidate f to candidate g has weight -4

Definition 2 (**Path Strength**) The strength of a directed path between two nodes in the WMG is the minimum weight of all the edges along that path.

The strength (of a path) can be negative.

Definition 3 (**Schulze Elections**) The Schulze election system is that candidate c is a winner exactly if for each other candidate d it holds that there is some simple path from c to d whose strength is at least as great as that of every simple path from d to c.

A lovely result is that the set of winners, under this definition, is always nonempty [34].

We now give a small Schulze-election example, adapted from an example of Parkes and Xia [31], over the candidate set $\{e, f, g, h\}$. Although the votes are not specified here, using McGarvey's method [26] we can build a profile of votes realizing the WMG of Fig. 1. (McGarvey's method is quite standard, so we ask the reader to here take our claim on faith. However, as Appendix B of our technical report version [20], we provide a detailed explanation of McGarvey's method, especially regarding the weighted case.) In Fig. 1's example, candidate g is the sole Schulze winner, strictly beating each other candidate in best-path strength. For each other candidate i, candidate g has a path to i of strength 6, but i's strongest path to g has strength 2.

2.2 Looping over integer linear programming feasibility problems, and the challenges that Schulze and ranked-pairs elections pose for this approach

One of the most powerful tools to use in building algorithms establishing fixed-parameter tractability is a result due to Lenstra, showing that the integer linear programming feasibility problem (henceforth, ILPFP) is in P if the number of variables is fixed [23]. Lenstra's result, based on the geometry of numbers, is very deep and so strong that intuition whispers it should not even be true; yet it is.

Now, if within an appropriate-sized integer linear program with a number of variables that was bounded by some function of the number of candidates we could capture our bribery/manipulation/control challenges and the action of the election system, we would be home free. Indeed, this has been done, for control, for such systems as plurality, veto, Borda, Dodgson, and others (see the discussion on p. 338 of [11]). However, Schulze and ranked-pairs elections have such extremely demanding definitions that they seem well beyond such an approach, and we have not been able to make that approach work. So we have a challenge.

Fortunately, the literature provides a way to hope to approach even systems that are too hard to directly wedge—together with the manipulative action—into an ILPFP. That approach is to define some sort of structure associated with subcases of behavior/outcomes of an election system, such that for each fixed number of candidates the number of such structures is bounded as a function of the number of candidates (independent of the number of voters), yet such that for each such structure we can wedge into an ILPFP the question of whether the given action can be made to succeed in the system in a way that is consistent with that structure. If that can be done, then we just loop over all such structures (for the given number of candidates), and for each of them build and run the appropriate ILPFP.

This has not been done often. But it has been done for example by Faliszewski et al. [14], with respect to some control problems, for the election system known as Copeland voting. The structure they used is what they called a Copeland Output Table, which is a collection of bits associated with the outcomes of the pairwise majority contests between the candidates.

Unfortunately, such output tables do not seem to have enough information to support the case of Schulze or ranked pairs. (Appendix A of our technical report version [20] provides an alternative winner certification framework that readers may wish to look at to get more of a sense of what certification frameworks do and how they may look. That alternative certification framework is closer in flavor to Copeland Output Tables than the approach we will outline here, although as that technical-report appendix discusses the alternative approach is both quite clear and quite subtle.) The natural structure that would allow us to tackle our systems is the one the systems are based on, namely, the WMG. Looping over all of those would allow us within the loop to easily write/run an appropriate ILPFP to check the given case. However, that falls apart because the number of WMGs is *not* bounded as a function of the number of candidates; the number also grows as a function of the number of voters. The impossibility of looping over WMGs leaves us still faced with the challenge of how to tackle our problems.

2.3 Our Schulze winner certification framework

A key contribution of this paper is to show that the needle described above can be threaded—and to thread it—for Schulze and ranked-pairs elections. In particular, we need to, for each of those election systems, find a (winner-set certification) structure that on one hand is rich enough that for each structure instance we can within an ILPFP check whether the given manipulative action can lead to success in a way consistent with the case of which the particular instance of the structure is speaking. Yet on the other hand, the structure must be so restrictive that the number of such structures is bounded purely as a function of the number of candidates (independent of the number of voters). In brief, we need to find, if one exists, a "sweet spot" that meets both these competing needs.

We achieve this with structures we call Schulze winner-set certification frameworks (SWCFs) and ranked pairs winner-set certification frameworks (RPWCFs). A Schulze winner-set certification framework contains a "pattern" for how we can prove that a given set of candidates is the winner set of a Schulze election. To do that, the structure for each winner a specifies, for each other candidate b, a "strong path" γ_{ab} from a to b in the WMG (recall that victory in Schulze elections is based on having strong paths), and then—to establish that the other candidate b has no stronger path back to a—for every simple path from b back to our candidate a the structure identifies a "weak link" (a directed edge on that path) that will keep the path from being too strong. By a "weak link," we mean an edge on that path in the WMG such that its weight is less than or equal to that of every edge in our allegedly

quite strong path γ_{ab}. (Now, keep in mind that, at the time we are looping through the structure, we will not even know how strong each link is, since the manipulation/bribery/control will not yet even have happened. Rather, the structure is specifying a particular pattern of victory, and the ILPFP will have to check whether the given type/amount of manipulation/bribery/control can bring to life that victory pattern.) Additionally, for each candidate a the structure claims is not a winner, the structure will specify what rival b eliminates that candidate from the winner set and then will outline a pattern for a proof that that is the case, in particular, giving a "strong path" from b to a and for each simple path from a to b our structure will specify a "weak link," i.e., an edge on that path from a to b whose weight in the WMG we hope will be *strictly* less than the weight of all edges in the selected strong path from b to a; if all our hopes of this sort turn out to be true (and that is among what the integer linear program will be testing, for each of our certification framework's structures), this proves that b eliminates a. Crucially, the number of structures in that Schulze winner-set certification framework, though large, is clearly bounded as a function of the number of candidates. The certification framework, however, does not itself have its hands on the weights of the WMG. So the paths and edges it specifies are all given in terms of the self-loop-free graph, on nodes named $1, 2, \ldots, \|C\|$, that between each pair of distinct nodes has edges in both directions. Since the candidate names are irrelevant in Schulze voting, we can change to those canonical names, so that our Schulze structures are always in terms of those names.

Crucially, as noted above, the number of structures in our Schulze winner-set certification framework, though large, is bounded as a function of the number of candidates. Yet, also crucially, this approach provides enough structure to allow a polynomial-sized (technically, it is actually a uniform-over-all-numbers-of-candidates polynomial multiplied by a constant that may depend on the number of candidates; this does not trivialize the claim, since the ILPFP must work for all numbers of voters) ILPFP to do the "rest" of the work, namely, to see whether by a given type of attack we can bring to life the proof framework that a given instance of the structure sets out, as to who the winners/nonwinners are in the Schulze election and why.

This approach underpins most of our fixed-parameter tractability results. Our proofs of Theorems 1 and 3 give detailed examples of this approach, and those two proofs respectively leverage our proofs of Theorems 2 and 6. Many additional examples can be found in the proofs that we omit here but have included in our technical report version [20].

For ranked pairs, the entire approach is what we just described above, except the certification framework we use is completely different than that used for Schulze. Ranked pairs is a method that is defined in highly sequential terms, through successive rounds some of which add a relationship between two candidates. So our certification framework will be making extensive guesses about what happens in each round (and about a number of other things). But again, we will ensure that the number of such certification structures is bounded as a function of the number of candidates (independent of the number of voters), yet each structure will give enough information that the rest of the work can be done by an integer linear programming feasibility problem. Our notion of a ranked pairs winner-set certification framework will be given in detail in Section 5.1.2 and will first be applied in Section 5.1.3.

3 Definitions

In this section, we define the ranked-pairs election system (Section 3.1), our manipulative-attack problems and their various models (Section 3.2), and the complexity class FPT (3.3).

3.1 Ranked-pairs elections

Schulze elections were defined in the previous section. We now define the quite different system known as ranked pairs, due to Tideman (see [35]). The ranked-pairs winner is defined by a sequential process that uses the weighted majority graph (WMG). We choose the edge in the WMG of greatest weight, say from a to b, and fix in the eventual output that a must beat b; cases of ties, either regarding what edge has the greatest weight, or regarding cases where (a, b) and (b, a) both are weight zero, are handled as will be specified in footnote 1. We then remove the edges between a and b from the WMG. We then iterate this process, except if the greatest remaining edge is one between two candidates who are already ordered by earlier fixings of output ordering (this can happen, due to transitivity applied to earlier fixings), then we discard the pair of edges between those candidates. We continue until we have completely fixed a linear order.[1] The candidate at the top of this linear order is the winner under ranked pairs. Even if the first removed edge is from a to b and that edge has positive weight, it is possible that a will not be the ranked pairs winner.

We give a small example of selecting the winner under ranked pairs. We again consider the election with candidate set $\{e, f, g, h\}$ and votes such that Fig. 1 is the WMG. We will break order-of-consideration ties (due to tied edge weights) between $\{a, b\}$ and $\{c, d\}$ in favor of which pair has the lexicographically-larger larger-candidate-of-the-pair, and if they tie in that, in favor of which pair has the lexicographically-larger smaller-candidate-of-the-pair. Thus we handle the edges in the following order: $f \to h$, $e \to f$, $h \to e$, $g \to e$, $g \to f$, $h \to g$. The output ordering will be set by those $f \succ h$, $e \succ f$, etc., except with $h \to e$ discarded due to transitivity, and after $g \to e$ sets $g \succ e$ we have a completed linear

[1] There are two different types of ties that must be handled. One is when we get to a case when we are considering an edge, and we don't discard it, and the candidates tie (the edges between them are both 0); here, we break ties using some simple ordering among the candidates. By simple, we mean feasible; there is a polynomial-time machine that, given the candidates, outputs a linear ordering of them that is the ordering to use when breaking ties of this sort. The second type of tie is when there is a tie as to what is the largest edge remaining in the WMG. In ties of that sort, we use a simple—again, by simple we mean feasible, analogously to the first case—ordering among all unordered pairs of candidates to decide which pair having a highest-weight edge still left is the one to next consider. If that pair is $\{a, b\}$ and both (a, b) and (b, a) have weight zero, either edge can be chosen to consider next, since which we consider at this point among (a, b) and (b, a) makes no difference in the result of this step.

An at first seemingly tempting alternate approach to breaking ties would be to require as part of the input the two types of tie-breaking orders discussed above. But that is highly unattractive, since that would require changing the definitions of long-defined problems (manipulation, control, bribery), in order to add that extra input part. In truth, the tie-breaking is being made, by us and the earlier papers, to be a part of ranked pairs; and so it should be a feature or setting that is part of one's version of ranked pairs, and should not be built in by hacking the notions of manipulative actions. So to us, if one wants to speak about ranked pairs, one to be clear and complete must also specify the two feasible tie-breaking functions that are needed to completely define the system. However, our main results for ranked pairs, which are fixed-parameter tractability results, will all hold for all feasible tie-breaking functions.

Here is an example of a pair of feasible tie-breaking functions. One could break ties between two candidates in favor of the lexicographically larger. And one could break ties between two candidate pairs in favor of the pair with the lexicographically-larger larger-candidate-of-the-pair, and when the larger members are the same in both pairs then breaking the tie in favor of whichever pair has the lexicographically-larger smaller-candidate-of-the-pair. The suggestion to use the candidate-vs.-candidate ordering to induce an ordering on the pairs—a suggestion our example is consistent with—was made and used by the creator of ranked pairs, Nicolaus Tideman, in his book "Collective Decisions and Voting: The Potential for Public Choice" [35].

order, $g \succ e \succ f \succ h$, and so the processing stops. Thus under ranked pairs, g is the sole winner in this example.

3.2 Models and manipulative-attack problems

As mentioned earlier, our elections are specified by a set of candidates and voters (each vote is a tie-free linear ordering of the candidates). The standard (also called "nonsuccinct") approach to the votes is that each comes in separately. In the succinct approach, which is meaningful only for systems such as Schulze and ranked pairs that don't care about voters' names, each tie-free linear ordering that is cast by at least one voter comes with, as a binary integer, the number of voters that voted that way.

In our problems we will speak of making a candidate p a winner or precluding p from being a winner. This is known as the nonunique-winner model or, in some papers, the co-winner model. If one changes "a winner" into "the one and only winner," that is what is known as the unique-winner model. Which model is more natural to use depends on which of these two questions is what one's setting is focused on: being *a winner*, or being *the one and only winner*. It has been argued by Faliszewski, Hemaspaandra, and Hemaspaandra [13] that the nonunique-winner model is often the better model on which to focus. In our paper, to support comparisons with each of these models, we will typically prove our results for both models. Fortunately, this usually takes almost no additional work or discussion. In fact, since ranked-pairs in the Parkes and Xia [31] version that we use never has more than one winner, the models coincide there. However, the same cannot be said for Schulze elections.

The problem definitions we are about to give present the definitions for the nonsuccinct, nonunique-winner case. However, the above two paragraphs make clear the very slight changes needed to define the succinct, nonunique-winner case, the nonsuccinct, unique-winner case, and the succinct, unique-winner case. For consistency with the literature, the wording of many of our attack-problem definitions is taken directly from or modeled on the definitions given in [14]. In these definitions, $\mathcal{E}$ will represent the election system.

Definition 4 (**The constructive and destructive bribery problems for $\mathcal{E}$ elections** [10])

Given: A set C of candidates, a collection V of voters (specified via their tie-free linear orders over C), a distinguished candidate $p \in C$, and a nonnegative integer k.

Question (constructive): Is it possible to change the votes of at most k members of V in such a way that p is a winner of the election whose voter collection is the thus-changed V and whose candidate set is C.

Question (destructive): Is it possible to change the votes of at most k members of V in such a way that p is not a winner of the election whose voter collection is the thus-changed V and whose candidate set is C.

In the literature, the above problem is often called the unweighted, unpriced bribery problem.

Definition 5 (**The constructive manipulation problem for $\mathcal{E}$ elections and the destructive manipulation problem for $\mathcal{E}$ elections** [1, 7])

Given: A set C of candidates, a collection V of voters (specified via their tie-free linear orders over C), a voter collection W (each starting as a blank slate), and a distinguished candidate $p \in C$.

Question (constructive): Is it possible to assign (tie-free linear order) votes to the members of W in such a way that p is a winner of the election whose voters are the members of V and W, and whose candidate set is C.

Question (destructive): Is it possible to assign (tie-free linear order) votes to the members of W in such a way that p is not a winner of the election whose voters are the members of V and W, and whose candidate set is C.

In the literature, this is often called the unweighted coalitional manipulation problem. We will sometimes refer to the members of the manipulative coalition W as manipulators and to the members of V as nonmanipulators.

As benchmarks, over time eleven "standard" types of control questions have emerged [2, 14, 19], each with a constructive version and a destructive version. Four of the eleven are each of adding/deleting (at most k, with k part of the input) candidates/voters. A fifth is so-called unlimited adding of candidates. The remaining six are partition of candidates, runoff partition of candidates, and partition of voters, each in both the model where first-round ties promote and in the model where first-round ties eliminate. The benchmark types are modeled on situations from the real world. For example, constructive control by adding voters models highly targeted get-out-the-vote drives and constructive control by adding candidates models trying to draw into elections spoiler candidates who will hurt your opponents. See for example [2, 14, 19] for a detailed discussion of the motivations of the benchmark control types. We mention in passing that recent work [18] has shown that in the nonunique-winner model there are at most nine distinct destructive control types and in the unique-winner model there are at most ten distinct control types; what we mean by this is that that paper shows that some control types are identical as to which instances they can be carried out on successfully.

Due to there being many control types, giving their definitions can take up a large amount of space. In order to keep the paper self-contained for those who want more formal definitions, as Appendix A we provide in their typically stated forms the definitions of all the control types. For those who already know the definitions or at this point simply want a brief treatment (knowing Appendix A is available with formal definitions), the rest of the present paragraph gives a short overview of the flavor of the different benchmark control attacks. All these problems have as their input an election, (C, V), and a distinguished candidate $p \in C$. Constructive (destructive) control by deleting voters—for a given election system, of course—also has a nonnegative integer k in the input and asks whether there is a subset of V of cardinality at most k such that with that subset removed p is (is not) a winner. Control by adding voters is analogous, except the input is the election, k, and a set W of voters who can be added (but at most k can be added). Deleting candidates and adding candidates are analogous to the voter cases, with a k as part of the input, and the only twist is that in destructive control by deleting candidates, it is forbidden to delete p. Unlimited adding of candidates is the same except there is no limit k. Constructive (destructive) partition of voters, in the ties-promote model, asks whether there is a way of partitioning the voters into two groups so that if all winners under the election system of each of those first-round elections compete in a final election under the same election system in which all voters vote (with their votes masked down to the remaining candidates), p is (is not) a winner. In its ties-eliminate variant, only unique-winners of a first round election move forward. The runoff partition of candidates types are analogous, except in the first round it is the candidates that are partitioned and all voters vote in each of those subelections. Partition of candidates has just one side of the partition participating in the first-round election, while the others get a bye to the final round.

3.3 Fixed-parameter tractability

We now define the important class FPT.

Definition 6 (**FPT, see** [30]) A problem is said to belong to the class FPT (equivalently, is said to be fixed-parameter tractable) with respect to a parameter if there is an algorithm for the problem whose running time is $f(j)I^{O(1)}$, where j is the input's value of that parameter, f is a computable function, and I is the input's size.

Note that this means that although the algorithm for larger values of j can have a bigger multiplicative constant, the degree of the polynomial running time is uniformly bounded from above—there is some single integer k such that regardless of the fixed j the algorithm for that parameter bound runs in time $O(I^k)$. Our parameter will almost always be the most natural one—the number of candidates. In Section 7, we will have cases where our parameter is a tuple of features of the input rather than a single feature, i.e., is a "combined" parameter (see [3, Chapter 9]). We stress that FPT should not be confused with the class, of which it is a subset, XP. XP problems are in P for each fixed bound on the parameter, but their degree can grow as that bound grows. Thus XP is a far less attractive class, even though by brute force it is often clear that parameterized versions of voting problems fall into it. However, our focus is firmly on the more demanding goal of establishing FPT results.

4 Related work

The computational complexity of manipulation, bribery, and control for Schulze voting and ranked pairs has been studied previously by Parkes and Xia, Xia et al., Menton and Singh, and Gaspers et al. [16, 28, 31, 38]. The work of Xia et al. and Parkes and Xia establishes (see also the table in [31]) that for Schulze elections constructive and destructive bribery, constructive and destructive control by adding and deleting voters, and constructive control by adding candidates are NP-complete, and that ranked pairs has not only all these hardnesses but also has NP-completeness results for constructive and destructive manipulation, for destructive control by adding candidates, and for constructive and destructive control by deleting candidates. Menton and Singh (see Table 1 of [28]) studied all remaining types of constructive and most types of destructive control for Schulze voting. Their work establishes that NP-completeness holds additionally for constructive control by unlimited adding of candidates, constructive control by deleting candidates, each variant of constructive control by partition or runoff partition of candidates, and each variant of constructive and destructive control by partition of voters. For the four cases of destructive control by partition or runoff partition of candidates, they show that polynomial-time algorithms exist.[2] The (unweighted coalitional) manipulation problem is shown to be in P for the destructive case by Parkes and Xia [31] and for the constructive case by Gaspers et al. [16]. Gaspers et al. [16] prove that the *weighted* constructive coalitional manipulation problem for Schulze elections is in

[2]For Schulze, three of the 22 benchmark control cases—each clearly belonging to NP for Schulze—were left open by the abovementioned papers: destructive control by adding candidates, destructive control by deleting candidates, and destructive control by unlimited adding of candidates. Indeed, even now these cases remain open as to whether they are in P, are NP-complete, or have some other complexity. However, for each of these three control cases (and all other of the benchmark control cases), we obtain membership in FPT.

polynomial time for each fixed number of candidates. We observe that inspection of their paper immediately makes clear that they have even established the stronger claim that the weighted constructive coalitional manipulation problem for Schulze elections is in the class FPT.[3]

All the results in the two papers involving Xia are in the unique-winner model. Ranked pairs is resolute (has exactly one winner), as Parkes and Xia frame it, and we follow their framing. (For nonresolute versions of ranked-pairs elections, the handling of tie-breaking is known to strongly affect complexity issues [6].) And so, as we noted earlier, the nonunique-winner and the unique-winner models are in effect the same for ranked pairs. Schulze is not resolute, but although Parkes and Xia's results on that are in the unique-winner model, they comment that their results all also hold in the nonunique-winner model. Gaspers et al. study both the nonunique-winner model and the unique-winner model. Menton and Singh use the nonunique-winner model as their basic model, as do we in the present paper. To us the nonunique-winner model is more attractive in not requiring a tie-breaking that, especially in symmetric cases, is often arbitrary and can change the flavor of the system. However, our main FPT results are proven by a loop approach over ILPFPs (integer linear programming feasibility problems), and it is clear that a straightforward adjustment to these will also handle the unique-winner cases. The key difference between our work and all the above-mentioned work is that our work is in general looking at the complexity of these problems when parameterized by the number of candidates, and for this we give FPT algorithms. The earlier papers primarily looked at unbounded numbers of candidates and obtained both P and NP-completeness results. Our contribution is that for all their NP-complete cases, we show membership in FPT.

As to technique, the closest precursors of this paper are two papers by Faliszewski et al. [11, 14]. Those, like us, use a loop over ILPFPs. There are two main differences between that work and ours. One is that they deal with control, whereas we are concerned with control, bribery, and manipulation. The other difference is that, as explained in detail in Section 2, their type of loop-over structure isn't flexible enough for our cases, and the natural structure for one to loop over (WMGs) generates a number of objects not bounded in the number of candidates. To handle this, and in contrast, our paper finds a middle ground that allows the loop to be over a bounded-in-$\|C\|$ number of objects, yet provides enough information in the objects so as to allow the ILPFPs to complete the checking of whether success is possible. For a different type of attack known as "swap bribery" and a different election system, Dorn and Schlotter [8] have recently employed what in effect (although implicitly) is a loop over ILPFPs, and they mention in passing without details that that swap bribery approach should apply to ranked pairs (please see footnote 4 of our technical report version [20] for a detailed discussion of the relationship of their work and ours).

[3] Briefly, the reason the algorithm of Gaspers et al. [16] for the weighted constructive coalitional manipulation problem for Schulze elections is clearly even an FPT algorithm is as follows. Their algorithm is using the following fact, which was observed independently by them and Menton and Singh [27]. For Schulze elections and the weighted constructive coalitional manipulation problem in the nonunique-winner model, it holds that if one can make a given candidate a winner, then there is a set of manipulative votes *in which all manipulators vote the same way* and that candidate is selected as a winner. Once one has this, an FPT algorithm is obtained simply by cycling over all possible preference orders, for each seeing whether, if that is what all the manipulators cast as their vote, the given candidate becomes a winner. And that is precisely what their short, elegant algorithm is doing for this case.

It is well-known that a problem is fixed-parameter tractable if and only if it has what is known as a kernelization [22]. So each fixed-parameter tractability result, including those of this paper, is establishing the existence of a kernelization. The size of the smallest kernels of a problem is an interesting issue of study, although one that is not the focus of this paper (however, see [4, 8, 17]).

Taking an even broader perspective, this work is part of a line that looks at the complexity of elections in the context of bounds on the number of candidates, a study that for example has been pursued famously by Conitzer, Sandholm, and Lang [7] regarding at what candidate numbers complexity jumps from P to NP-complete. The particular focus on FPT algorithms, and maintaining a uniform degree bound over all values of bounds on the number of candidates, is part of the important field of parameterized complexity (see [30], see [4] for a survey on this approach for elections, and see [32] for a survey of an alternate approach to bypassing complexity results).

5 Results by looping over frameworks

We now present our results that are established by our looping-over-frameworks approach. We will handle in separate sections bribery, control, and manipulation, showing how to achieve FPT results for each.

In the bribery section, Section 5.1, we will first prove the bribery result for Schulze elections, so that the reader quickly gets to seeing how the proof goes without having to have first seen how the approach works for ranked pairs. We then will give our ranked pairs winner-set certification framework, and will note how to convert our proof into a proof for that case also. Then later in the control section, Section 5.2, we will state and prove together the Schulze-elections case and the ranked-pairs case. To avoid overly much repetition of proofs using the looping approach, in this journal version we will present the detailed proofs of two theorems using that approach, namely, Theorem 1, which is about bribery, and Theorem 3, which is about control. We will refer the reader to our technical report version [20] for detailed proofs of most of the remaining results that are proved by the looping framework, as well as a few other lengthy proofs.

Since (unweighted) Schulze manipulation has been shown to be in P in general, both for the constructive [16] and the destructive [31] cases, we do not need to handle Schulze elections in our manipulation section, Section 5.3. (For the nonunique-winner model, weighted constructive coalitional manipulation for Schulze elections, Gaspers et al. [16] provide what as mentioned earlier is an FPT algorithm. And in Section 7 we will establish, as part of Theorem 11, that for Schulze elections our approach provides an FPT algorithm for special cases of both weighted *destructive* coalitional manipulation and unique-winner model, weighted constructive coalitional manipulation.)

5.1 Bribery results and specification of ranked pairs winner-set certification framework

In this section, we will first state and prove the bribery result for Schulze, then will give our winner-set certification framework for ranked pairs, and then will state and prove our bribery result for ranked pairs.

5.1.1 Bribery result for Schulze

We state and prove the bribery result for Schulze. To help us avoid repetition in theorem statements, we first introduce a shorthand that we will use in most of our theorem statements. The "eight" below is because each of the three binary choices can be independently made.

Notation 1 We will use "in our eight bribery models" as a shorthand for "in both the succinct and nonsuccinct input models, for both constructive and destructive bribery, in both the nonunique-winner model and the unique-winner model." We will use the analogous shorthand for the cases of control and manipulation.

Theorem 1 *For Schulze elections, bribery is in FPT (is fixed-parameter tractable) with respect to the number of candidates, in our eight bribery models.*

Proof Our proof will rely heavily on the notion of Schulze winner-set certification frameworks (SWCFs), which we introduced in Section 2.3. Recall from that section that for each (number of candidates) j, the number of j-SWCFs is constant. That is, the number of j-SWCFs is bounded by a function of the number of candidates, and is independent of the number of voters.

We first give the proof for the constructive, nonunique-winner model case. We will handle simultaneously the succinct and nonsuccinct cases.

Our FPT algorithm works as follows. It gets as its input an instance of the bribery problem, and so gets the candidates (with a distinguished candidate noted), the votes (or for the succinct version, a list of which types of votes occur at least once, along with the multiplicities of each), and the limit k on how many voters can be bribed. Let j be the number of candidates in the input instance. To mesh with the naming scheme within our SWCFs, we immediately rename all the candidates (including within the votes) to be $1, \ldots, j$, with the distinguished candidate becoming candidate 1. Now, the top-level programming loop of the algorithm is as specified in Algorithm 1 (recall that we are showing the constructive, nonunique-winner model case in this algorithm).

Algorithm 1 Top-level loop for bribery

```
Start
for each j-SWCF K do
    if candidate 1 is a winner according to K and K is an internally consistent,
    well-formed j-SWCF then
        (1) build an ILPFP that checks whether there is a way of bribing at most k of the
        voters such that K's winner-set certification framework is realized by that bribe
        (2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying
        settings will even let us output the precise bribe that succeeds)
    end if
end for
declare that the given goal cannot be reached by using at most k bribes
End
```

All that remains is to specify the ILPFPs that we build inside the loop, for each given j-SWCF K. Suppose we are doing that for some particular K. We do it as follows.

There are $j!$ possible votes over j candidates; let us number them from 1 through $j!$ in any natural computationally simple-to-handle way. We call the ith of these the ith "vote type." We will have constants n_i, $1 \leq i \leq j!$, denoting how many voters start with vote type n_i. Our ILPFP will have integer variables (which we will ensure are nonnegative) $m_{i,\ell}$, $1 \leq i \leq j!$, $1 \leq \ell \leq j!$. $m_{i,\ell}$ is the number of voters who start with vote type i but are bribed to instead cast vote type ℓ. (Having $m_{i,i} \neq 0$ is pointless but allowed, as is having simultaneously $m_{i,\ell} > 0$ and $m_{\ell,i} > 0$.) So the number of ILPFP variables is $(j!)^2$, which is large but is bounded with respect to j, so Lenstra's algorithm [23] can be used to deliver an FPT performance overall.

Also, not as direct parts of the ILPFP but as tools to help us build it, we define two boolean predicates, $Bigger(a, b, c, d)$ and $StrictlyBigger(a, b, c, d)$, where the arguments each vary over $1, \ldots, j$. Let us use $D(a, b)$ to indicate the weight of the WMG edge (after our manipulative actions) that points from a to b. Recall, from Section 2, that what a j-SWCF does—and so in particular what our under-consideration j-SWCF, K, does—is specify, for a very large number of such quadruples (that number is bounded as a function of j, although we do not need that since our number of variables is already bounded as a function of j), that $D(a, b) \geq D(c, d)$ or that $D(a, b) > D(c, d)$. That is, it is specifying for various edge pairs in the WMG that the one edge is greater than or equal to the other edge in weight, or is strictly greater in weight. For each such specified relation that explicitly appears in K, set to true that bit in the appropriate predicate (*Bigger* or *StrictlyBigger*), and leave all the other bits set to false. We of course will have to enforce these specifications through our constraints.

Now we can specify all the constraints of our ILPFP. There will be three types of constraints. The first are the housekeeping constraints to make sure that the number of bribes and the $m_{i,\ell}$s are all reasonable. Our constraints of this sort are: For each $1 \leq i, \ell \leq j!$, we have a constraint $m_{i,\ell} \geq 0$. For each $1 \leq i \leq j!$ we have a constraint $n_i \geq \sum_{1 \leq z \leq j!} m_{i,z}$; that is, we do not try to bribe away from vote type i more votes than initially exist of vote type i. And we have the constraint $k \geq \sum_{1 \leq i', \ell' \leq j!} m_{i',\ell'}$; that is, our total number of bribes does not exceed the bribe limit k.

The second type of constraint consists of those constraints used to enforce the bits set to "true" in *StrictlyBigger*. For each such bit, we will generate one constraint: for a bit that is saying that $D(a, b) > D(c, d)$, we will enforce that with the constraint shown in Fig. 2; in that figure and for the rest of this paper, in order to make the representation of the constraints more concise, we introduce the shorthand notation $pref(a, b)$ as the set of vote types i, $1 \leq i \leq j!$, in which candidate a is preferred to candidate b. All that the bulky-looking constraint of the figure says is that after all the gains and losses due to bribing happen, the number of voters who prefer a to b minus the number who prefer b to a is strictly larger than the number of voters who prefer c to d minus the number who prefer d to c. If $StrictlyBigger(a, b, c, d)$ is set to "false," that does *not* mean we generate a constraint ensuring that $D(a, b) \not> D(c, d)$. Rather, if a given bit is set to "false," that just means that that particular bit-setting does not itself create a constraint. In contrast, bits set to "true" in *StrictlyBigger* and *Bigger* mean that we generate a constraint to enforce the stated relation.

The third type of constraint consists of those constraints used to enforce the bits set to "true" in *Bigger*. For each such bit, we will generate one constraint: for a bit that is saying that $D(a, b) \geq D(c, d)$, we will enforce that with precisely the constraint shown in Fig. 2, except with the "1+" removed from the right-hand side of the inequality.

That completes our statement of the ILPFP, which indeed captures what it seeks to capture. And using Lenstra's algorithm [23] for each of our ILPFPs, the overall loop over the

$$\left(\sum_{i\in pref(a,b)}\left(n_i-\left(\sum_{1\le\ell'\le j!}m_{i,\ell'}\right)+\left(\sum_{1\le i'\le j!}m_{i',i}\right)\right)\right)$$
$$-\left(\sum_{i\in pref(b,a)}\left(n_i-\left(\sum_{1\le\ell'\le j!}m_{i,\ell'}\right)+\left(\sum_{1\le i'\le j!}m_{i',i}\right)\right)\right)$$
$$\ge 1+\left(\sum_{i\in pref(c,d)}\left(n_i-\left(\sum_{1\le\ell'\le j!}m_{i,\ell'}\right)+\left(\sum_{1\le i'\le j!}m_{i',i}\right)\right)\right)$$
$$-\left(\sum_{i\in pref(d,c)}\left(n_i-\left(\sum_{1\le\ell'\le j!}m_{i,\ell'}\right)+\left(\sum_{1\le i'\le j!}m_{i',i}\right)\right)\right).$$

Fig. 2 Constraint enforcing that, after the bribes happen, $D(a, b) > D(c, d)$

ILPFPs has the desired running time. (Although for each fixed j the multiplicative constant is very large, the degree of the polynomial, which is uniform over all j, isn't terrible; Lenstra's algorithm uses just a linear number of arithmetic operations on linear-sized integers [30]. Still, even within the good news that we have placed the problem within FPT, there is the bad news that the multiplicative constant is so large that this FPT algorithm does not provide an algorithm for practical use.)

To be clear as to what is going on this proof, since what is a constant and what is a variable is a bit subtle here, let us say a bit more about the use of Lenstra here. What we in effect are using is that Lenstra's work ensures that there is a k such that for each fixed number of candidates and each of the (large but bounded as a function of the number of candidates) ILPFPs generated in our loop, if we view that ILPFP as an object whose running time for solution is being evaluated asymptotically as the number of voters increases without bound, the ILPFP's running time is $O(n^k)$. (That same value k holds for all numbers of candidates and for all ILPFPs that our loop generates for that number of candidates. However, for different numbers of candidates the multiplicative constant represented by the "big O" may differ.) Note that each such ILPFP object in effect has as *its* set of variables (regarding the asymptotics of its running time) the *constants* of the ILPFP; and a big part of what our looping algorithm does is to set those constants based on the votes in the election.

That was the proof for the constructive, nonunique-winner model case. To change the above proof from the constructive to the destructive case and/or from the nonunique-winner case to the unique-winner case, in the main loop we will simply create ILPFPs for only those SWCFs whose set of who wins and loses reflects a sought outcome. For example, for the destructive case in the unique-winner model, that would be having the distinguished candidate not be a unique winner, i.e., the start of Algorithm 1's "if" statement would become "if candidate 1 is not a unique winner. □

The same proof approach applies to ranked pairs. However, we first must do some work to define an appropriate winner-set certification framework for ranked pairs. We turn to that now.

5.1.2 Specification of ranked pairs winner-set certification framework

In this section, we describe the winner-set certification framework that makes our approach work for ranked pairs.

Basically, an instance of that framework will be a story that tells us what happens at each stage of the iterative process that defines ranked pairs. We could actually tell this story without fixing up front, for each pair $\{a, b\}$ of distinct candidates, whether a is preferred to

b by a majority of the voters, or whether b is preferred to a by a majority of the voters, or whether a and b exactly tie as to how many voters prefer one to the other. Not fixing that information up front would improve our multiplicative factor that depends on but is fixed for each fixed number of candidates. But we are not focused on that factor. So to make things particularly simple to describe, we are here just going to toss into our framework a fixing of all such pairwise-outcomes-in-the-WMG. As in the Schulze case, we will have changed all the names of the candidates to be 1 through $\|C\|$ (and will have remapped our tie-breaking function in the same way). So, one part of our framework is, for each (unordered) pair of distinct candidates $\{a, b\}$, a claim as to which one of these holds: the WMG edge from a to b is strictly positive, the edge from b to a is strictly positive, or both edges are 0. (We do not include any claim about the precise value of those edge weights; that would create a framework whose number of instances, for a fixed number of candidates, grew with the number of voters—something we must firmly avoid.) And an instance of the framework then goes step by step through the process the ranked-pairs algorithm goes through, but in a somewhat ghostly way in terms of what it specifies. For each step of the process, it makes a claim as to what pair of candidates is considered next. And it makes a claim as to whether that pair of candidates will be skipped permanently due to it having been already set (due to transitivity) by earlier actions of our flow through ranked pairs (that isn't an on-the-fly thing in that the instance itself has all its earlier claims and so we can even make sure to loop only over instances of the framework that are internally consistent regarding this). And if it claims that that pair of candidates is not skipped, it also makes a claim about which of the two outcomes happens (which is placed above the other in our ranked-pairs outcome; again, we can read this from those choice-of-3-possibilities settings we did up front, plus the feasible tie-breaking if needed). So that is the story the framework provides, and a given instance of the framework will (if properly formed) set an ordering over all the candidates. As before, the algorithms will loop over instances of these frameworks, doing so over only instances that have the desired outcome (e.g., "p is a unique winner") and that aren't obviously internally inconsistent. (For our partition by voter cases, there is a double-loop over such frameworks, to handle both subelections.)

As in the Schulze case, we will use the ILPFPs to see if the given kind of control can create a case where the given framework can be made to hold. All the housekeeping work in the ILPFPs as to tallying how the votes are bribed/controlled/manipulated is still needed here (so the variable sets are the same as the ones for Schulze). But note, crucially, that we now must enforce not things about paths, but rather we must enforce that the framework's guesses about whether the edge from a to b is negative, positive, or 0 after the bribery/control/manipulation are all correct (this is very natural to enforce with constraints, within the ILPFP framing), *and must also enforce that the framework's claim about which candidate pair is considered next is what would actually happen under the votes that emerged from the bribery/control/manipulation*. But that latter claim, for each step in the story, can be checked by appropriate, carefully built constraints, written with close attention paid to the tie-breaking rule among pairs. These constraints will be pretty much our favorite sort of constraint—seeing whether a WMG edge is greater than or equal to another, or seeing whether it is strictly greater than another. This will be made clearest by an example. Suppose our candidates are named *1*, *2*, *3*, and *4*. And suppose the tie-breaking order on unordered pairs is $\{4, 3\} > \{4, 2\} > \{4, 1\} > \{3, 2\} > \{3, 1\} > \{2, 1\}$, and on candidates is $4 > 3 > 2 > 1$. (This is exactly a case of the sample feasible rule pair we gave in footnote 1, under which, recall, tied pairs are tie-broken in favor of the pair with the lexicographically-larger larger-candidate-of-the-pair, and when the larger members are the same in both pairs then the tie is broken in favor of whichever pair has the lexicographically-larger

smaller-candidate-of-the-pair. We've written our unordered pairs, in the tie-breaking order above, with the lexicographically larger element first simply to make it clear why footnote 1's example tie-breaking rule would put them in the order shown above.) Suppose the RPWCF says the first pair to be compared is $\{1, 4\}$ and that the outcome is $4 \succ 1$. Let $D(a, b)$ be defined as before. To check that $4 \succ 1$ is the right outcome, since $4 > 1$ in the tie-breaking function we need to check that $D(4, 1) \geq D(1, 4)$ (if $1 \succ 4$ in the tie-breaking order, we'd check that $D(4, 1) \geq 1 + D(1, 4)$); we can read this right off the 3-way-claim as to how *1* and *4* compare in their head-to-head contest, which itself we'll enforce in constraints. And as to the claim that $(1, 4)$ was the first pair to be compared, in light of the tie-breaking order, that can be enforced using 10 constraints: 6 saying that our pair ties or beats those below us in the tie-breaking ordering ($D(4, 1) \geq D(a, b)$ for the (a, b) values $(3, 2)$, $(2, 3)$, $(3, 1)$, $(1, 3)$, $(2, 1)$, $(1, 2)$), and 4 saying that our pair strictly beats those above us in the tie-breaking ordering ($D(4, 1) \geq D(a, b)$ for the (a, b) values $(4, 3)$, $(3, 4)$, $(4, 2)$, $(2, 4)$). We could cut those $6 + 4$ constraints to $3 + 2$ if we wish, by using the value of the 3-way-claim for each of those 5 other pairs. Note that all these comparisons are about post-bribe/manipulation/control vote numbers—things we do know how to easily put into an ILPFP constraint, basically, by appropriate summations. Moving on, if the framework says the next unordered pair after $\{1, 4\}$ to be considered is $\{1, 3\}$ and that the outcome is $1 \succ 3$, we of course will not need to enforce any comparisons with $D(4, 1)$ or $D(1, 4)$; we will generate and put into the ILPFP just the needed/appropriate comparisons. If the framework after that says the third pair to consider is $\{3, 4\}$ but it also says that (due to $4 \succ 1 \succ 3$ already being set) the pair $\{3, 4\}$ gets skipped, we under our RPWCF framework still must generate the constraints to check that $\{3, 4\}$ truly under our votes as they now are *did* deserve to come up next (we mention in passing that we could skip that check as long as we adjust our ILPFP to not check anything regarding pairs that are already related, even transitively, under the $\succ$'s so far—a slightly different approach than ours but also quite fine), but the generated comparisons-to-check-that will not do comparisons against things the existing order so far ($4 \succ 1 \succ 3$) takes out of play. This completes our description of our RPWCF notion.

5.1.3 Bribery result for ranked pairs

Having specified the ranked pairs winner-set certification framework, the bribery case for ranked pairs can now be stated and justified.

Theorem 2 *For ranked pairs (with any feasible tie-breaking functions), bribery is in FPT (is fixed-parameter tractable) with respect to the number of candidates, in our eight bribery models.*

Proof We use the programming loop of Algorithm 1 to now loop over not j-SWCFs, but instead over j-RPWCFs, with j again being the number of candidates. The variables of the ILPFPs will be the same as those for Schulze. As for the constraints of the ILPFPs, all the housekeeping constraints for bribery remain intact. Additionally an RPWCF, as described in Section 5.1.2, guesses for each edge whether it is positive, negative, or zero in weight. We can easily handle these possibilities with the constraints $D(a, b) \geq 1$, $D(b, a) \geq 1$, and $D(a, b) = 0$ respectively. The other constraints enforced by an RPWCF are those between pairs of edges, and they can be successfully captured by the *StrictlyBigger* and *Bigger* predicates defined in the proof of Theorem 1. Thus we clearly can build an ILPFP encoding an RPWCF and the constraints of the bribery problem

in the same way we did for Schulze elections, and bribery is in FPT for ranked pairs as well. □

5.2 Control results

In this section, we establish our results for control gained through the looping-over-frameworks approach. The proofs for these cases will closely follow our general looping-over-frameworks structure, and we will just have to appropriately build the constraints of our ILPFPs to handle the details of the control problems.

Theorem 3 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by adding voters is in FPT with respect to the number of candidates, in our eight control models.*

Proof We will handle together all the cases of this theorem.

Again, let the candidates in the control problem be $1, \ldots, j$ with the distinguished candidate being 1. The control problem has as its input a set of initial votes V and a set of additional votes W (or for the succinct version, one list for each of these two sets describing which types of votes occur at least once in that set, along with the multiplicities of each), the latter of which contains the votes that can be added by the control action. Also there is a limit k on the number of votes that can be added from W.

The top-level programming loop is as described in Algorithm 2 (with WCF being either SWCF for Schulze or RPWCF for ranked pairs).

Algorithm 2 Top-level loop for control by adding voters

Start
for each j-WCF K **do**
  if candidate 1 (is/is not) (a winner/a unique winner) according to K and K is an internally consistent, well-formed j-WCF **then**
    (1) build an ILPFP that checks whether there is a set of votes $W' \subseteq W$, with $\|W'\| \leq k$, such that K's winner-set certification framework is realized by the set of votes $V \cup W'$
    (2) run that ILPFP and if it can be satisfied then halt and accept (note: the satisfying settings will even let us output the precise added set that succeeds)
  end if
end for
declare that the given goal cannot be reached by adding at most k voters
End

The "if" line at the start of the algorithm should be set to "is" ("is not") for the constructive (destructive) case, and to "a winner" ("a unique winner") for the nonunique-winner model (the unique-winner model).

All we have to do now is show how we build the ILPFP inside the loop. Again, as with manipulation, we will have two groups of constraints: those corresponding to the WCF-enforcing predicates and those corresponding to the structure of the control problem. For every i, $1 \leq i \leq j!$, we will have a variable v_i representing the number of votes of type i in

W' (i.e., the W' sought by Algorithm 2), a constant n_i representing the number of votes of type i in V, and a constant h_i representing the number of votes of type i in W.

As we described in the proof of Theorem 6, if we can represent $D(a, b)$ as a linear expression in terms of our constants and variables, we can implement all the WCF-enforcing constraints (those of the *StrictlyBigger* and *Bigger* predicates and those of the 3-way possibilities for ranked pairs) as linear constraints in our ILPFP. Thus, using the shorthand notation $pref(a, b)$ as described in Section 5.3, we express $D(a, b)$ as follows:

$$\sum_{i \in pref(a,b)} (n_i + v_i) \quad - \quad \sum_{i' \in pref(b,a)} (n_{i'} + v_{i'}).$$

As for the constraints ensuring the validity of the control action, we first need to ensure that for every type of vote, the number of votes of that type in W' is bounded by the number in W. For every vote type i, $1 \leq i \leq j!$, the constraint $v_i \leq h_i$ will enforce this in our ILPFP. We also make the following constraint to enforce the adding bound:

$$\sum_{1 \leq i \leq j!} v_i \leq k. \tag{5.1}$$

Here again all of our variables have to be nonnegative, and thus we have the constraint $v_i \geq 0$ for every i, $1 \leq i \leq j!$.

This suffices to describe how we can build a WCF-enforcing ILPFP for the control by adding voters problem in a way appropriate for our looping-over-frameworks technique. Thus we have an algorithm that will run in uniform polynomial time for every fixed parameter value, putting this problem in FPT. □

Theorem 4 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by deleting voters is in FPT with respect to the number of candidates, in our eight control models.*

Theorem 5 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control by partition of voters is in FPT with respect to the number of candidates, in both the ties-eliminate and ties-promote models, in our eight control models.*

The proofs of Theorems 4 and 5 are provided in our technical report version [20].[4]

5.3 Manipulation results

Unweighted manipulation in Schulze elections has recently been shown to be in P by Gaspers et al. [16] for the constructive case. So, since the destructive case was itself handled even earlier by Parkes and Xia [31], there is no need to cover (unweighted) manipulation for Schulze elections in this paper.

We now present our FPT result for (unweighted) manipulation in ranked pairs elections. Although for this case we could use the looping-over-framework approach to obtain

[4]For readers who refer to those proofs, we mention that in those proofs and a few of the other proofs for which we refer the reader to that report, the proof says something of the form, "The two binary selections in the algorithm's 'if' statement are made as in the proof of Theorem 5.6." Yet Theorem 5.6 of the technical report in fact does not discuss the issue. What is meant in each case is: "The 'if' line at the start of the algorithm should be set to 'is' ('is not') for the constructive (destructive) case, and to 'a winner' ('a unique winner') for the nonunique-winner model (the unique-winner model)."

an FPT algorithm, we instead obtain an FPT algorithm here simply as a consequence of Theorem 3.

Very loosely put, the idea behind this proof is that, in a certain sense, for fixed numbers of candidates manipulation nicely reduces—not just regarding ranked pairs but in fact as a general matter—to control by adding voters, simply by putting into the set of potential voters to add enough copies of every possible vote. Unfortunately, this approach doesn't quite work, due to the standard definition of control by adding candidates bounding rather than exactly setting the number of additions, which is what one needs as one's reduction's target to make this connection work. However, this worry is easy enough to smooth over that we still can prove the manipulation case here as a quick consequence of groundwork done for control.

In the proof of Theorem 6, we will use the notion of fpt-reductions (by which we will always mean many-one fpt-reductions). This notion will be further used in Section 6.4. We now give the standard definition of fpt-reductions for the case of reductions from a problem, call it Q, with respect to a parameter j, to a problem, call it Q', with respect to a parameter j'. Let $j(x)$ be the function that given an input to Q gives the value of parameter j on input x. Let $j'(x)$ be the function that given an input to Q' gives the value of parameter j' on input x. A function R is an fpt-reduction from Q to Q' if the following three conditions hold [15]: (i) for each x, $x \in Q$ if and only if $R(x) \in Q'$, (ii) there exist a polynomial p and a computable function f such that R is computable in time $f(j(x))p(|x|)$, and (iii) there exists a computable function g such that, for all x, $j'(R(x)) \leq g(j(x))$.

Theorem 6 *For ranked pairs (with any feasible tie-breaking functions), manipulation is in FPT with respect to the number of candidates, in our eight manipulation models.*

Proof We will handle together all the cases of this theorem. Let us again assume without loss of generality that the candidates in the manipulation problem are $1, \ldots, j$ with the distinguished candidate being 1. The manipulation problem has as its input a set of nonmanipulative votes V (or for the succinct version, a list of which types of votes occur at least once, along with the multiplicities of each) and the set of manipulators W.

We give an fpt-reduction from this problem to a slightly modified version of control by adding voters. The difference is that instead of having a bound on the number of votes which we can add, we will need an exact number of votes to be added. This modification will change only one of the constraints in our ILPFP (the constraint labeled 5.1 in the proof of Theorem 3) from being an inequality to an equality, and so it is clear that the thus-altered control by adding voters problems remain in FPT (by the thus-altered version of Theorem 3 and its proof). So an fpt-reduction to that problem will establish our desired FPT result for manipulation.

Our (modified) control instance will have the same set of candidates, and the same distinguished candidate. The set of initial votes in the control instance will be identical to the set of nonmanipulative votes in the manipulation instance. As for the additional vote set in the control instance, it will include $\|W\|$ copies of each possible vote (out of all $j!$ votes) (or for the succinct version, a list of all $j!$ vote types, each with multiplicity of $\|W\|$). Finally, the "exact" number of votes to be added is equal to $\|W\|$. This completes the specification of the reduction.

Clearly, a manipulation instance is a Yes instance of manipulation if and only if this construction maps to a Yes instance of the modified control problem. Furthermore, the mapped-to control instance will have the same number of candidates as the mapped-from manipulation instance. Finally, the reduction runs in time $|x|(j!)$, where $|x|$ is the manipulation

problem's input size. Thus the reduction meets the requirement of an fpt-reduction. This proves that manipulation is in FPT with respect to the number of candidates. □

6 Other results for the unweighted case

This section presents some additional results regarding the unweighted case. Section 7 will discuss the weighted case,

Our FPT control results so far have been about voter control, parameterized on the number of candidates. It is natural to wonder about other cases. Are Schulze elections and ranked-pairs elections also in FPT for candidate control, parameterized on the number of candidates? Are Schulze elections and ranked-pairs elections in FPT for voter control, parameterized on the number of voters? In this section, we establish "yes" answers to both these questions.

Although parameterizing on the number of candidates is by far the most commonly studied parameterization for election problems, it is easy to imagine situations where the focus parameter is the number of additions/deletions allowed. This section proves that constructive control by adding voters, deleting voters, and adding candidates are each W[2]-hard for Schulze elections, when parameterized on the addition/deletion bound. This is strong evidence that these problems do not belong to FPT. We also observe that it follows from an examination of the proofs of Menton and Singh [28] that many control problems for Schulze elections are NP-complete even when, for example, one severely limits the number distinct weight values that can appear in the WMG.

Parameterizing by number of candidates is certainly the most natural approach to parameterization for election-manipulation problems. Indeed, it was the only parameterization we studied in Section 5. However, we feel that this section's results, many involving other parameterizations, provide a broader and more complete picture of the control complexity of Schulze elections. See the survey of Betzler et al. [4] for many examples of interesting results under parameterizations other than the number of candidates.

6.1 Candidate control parameterized on the number of candidates

Under our primary parameterization of interest, parameterizing on the number of candidates, and when considering manipulation, bribery, and voter control, we achieved FPT results using the looping-over-frameworks technique, and thus involving Lenstra's algorithm. In contrast, when considering candidate control problems parameterized on the number of candidates, we need not use such a powerful technique. Instead it is sufficient to brute-force search over all possible control solutions to see if any of them are successful. At every possible value for the parameter, there are only a constant number of possible solutions to any of the candidate control problems, and checking the success of the possible solution will require only a simple polynomial-time task. Thus we have algorithms that at each fixed parameter value will have a running time that is a (large, parameter-value-dependent) constant times a small uniform polynomial. This puts these problems in FPT. We mention that for the adding/unlimited adding of candidates cases, the parameter can even be taken to just be the size of the pool of potential additional candidates, regardless of how many original candidates there were.

Theorem 7 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control is in FPT with respect to the number of candidates, for all standard types of candidate control (adding/unlimited adding/deleting candidates and, in both the ties-eliminate and ties-promote first-round promotion models, partition and runoff partition of candidates), in our eight control models.*

Proof In the case of adding candidates, at most all the $2^{||D||}$ possible subsets of the auxiliary candidate set D need be considered. In the case of deleting candidates, at most all subsets of C that contain the distinguished candidate p need be considered, and so we need look at at most $2^{||C||-1}$ subsets. (In the destructive case, the definition of this control type forbids trivially satisfying the goal by deleting p. In the constructive case, deleting p would make success impossible and so it need not be considered.) In the case of runoff partition, there are $2^{||C||-1}$ interestingly distinct partitions, while in the partition case there are $2^{||C||}$. The difference between these two different types of partition cases is because in runoff partition of candidates case the two parts of the partition are handled symmetrically, and so the partitions (A, B) and (B, A) are not interestingly distinct from each other, and we need consider just one among them. However, in the partition of candidates case, where one side of the partition is getting a bye, no such general symmetry can be claimed. For each of these cases, for each setting of its item that we are cycling through above, we have to call the voting system's winner problem between one and three times. So all these cases will be in FPT for any voting system with a polynomial-time winner problem. □

6.2 Voter control parameterized on the number of voters

Although we feel that the number of candidates is the most natural parameterization for manipulative action problems, it is natural to ask about parameterizing on the number of voters. We do not exhaustively handle this case for all manipulative action problems, but we note that voter control problems parameterized on the number of voters can be shown to be in FPT through simple brute-force. Again, as in the case of candidate control problems parameterized on the number of candidates, we note that the number of possible solutions to these problems is bounded by a constant for each parameter value, and checking each solution is easily done in polynomial time, giving us an FPT algorithm for each of the voter control problems. We mention that for the adding of voters cases, the parameter can even be taken to just be the size of the pool of potential additional voters, regardless of how many original voters there were.

Theorem 8 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, control is in FPT with respect to the number of voters, for all standard types of voter control (adding/deleting voters and, in both the ties-eliminate and ties-promote first-round promotion models, partition of voters), in our eight control models.*

Proof In the case of adding voters we will have to try at most all of the $2^{||W||}$ possible subsets of the auxiliary voter set W. In the case of deleting voters we will have to consider at most all the $2^{||V||}$ subsets of the voter set V. And in the partition of voters cases we will have to consider the $2^{||V||-1}$ interestingly distinct partitions of the voter set (again, we need consider just one among the partitions (A, B) and (B, A)). In all of these cases the large exponential term of the complexity will be constant with fixed parameter values. And

beyond that term, we will just have to perform a few iterations of the voting system's winner function along with a few other simple checks, putting all these cases in FPT for any voting system with a polynomial-time winner problem. □

6.3 WMG edge bound parameterization

Let us return to considering the case of parameterization by number of candidates. What drove us to our approach of looping over the winner "frameworks" we defined, rather than just looping over all WMGs? It was the fact that even for fixed numbers of candidates, the number of WMGs blows up as the number of voters increases. We mention in passing, though, that if one, *in addition* to parameterizing on the number of candidates, requires that the absolute value of the edge weights in the WMG be bounded by some fixed constant independent of the number of voters, then for that particular special case, one could loop over all WMGs. Is this natural and important? We would tend to say "no," because assuming that all edges in the WMG have weights bounded by k is to assume that even as the number of voters grows, every single head-on-head contest between pairs of candidates is very evenly matched. That simply is not the case in most natural elections.

On the other hand, perhaps surprisingly, there is something theoretical to be gained from the strange approach just mentioned of considering elections in which all edges of the WMG turn out to have relatively low weights. In particular, we observe that in the NP-hardness-establishing reductions *to* Schulze control problems used by Menton and Singh [28], all edges in the WMG have absolute value at most 6 (and for some types of control, at most 4 or 2). That, along with that fact that all weights of edges in the WMG have the same parity, gives us Corollaries 1 and 2.

Corollary 1 (Corollary to the proofs of Menton and Singh [29])

1. *Even when restricted to instances having all pairwise contests so equal that each WMG edge*[5] *has absolute value at most 2, Schulze elections are NP-complete (in the nonunique-winner model) for constructive control by deleting candidates.*
2. *The same claim as in part 1 holds for constructive control by adding candidates, unlimited adding of candidates, partition of candidates in the ties-eliminate model, and runoff partition of candidates in the ties-eliminate model, except with a bound of 4.*
3. *The same claim as in part 1 holds for constructive control by partition of candidates in the ties-promote model and runoff partition of candidates in the ties-promote model, except with a bound of 6.*

Clearly, this immediately implies the following result (still keeping in mind that the values of all edge weights are of the same parity).

[5] In Corollaries 1 and 2, when we speak of restricting a WMG, the WMG we are speaking of as having to obey the restriction is the WMG involving *all* candidates involved in the problem. So for adding candidates, the WMG this is speaking of is, using the votes in the problem instance, the WMG whose nodes are all the initial candidates and all the candidates in the pool of candidates that can potentially be added. Since membership in NP for all the problems discussed is obvious, what is most interesting in Corollaries 1 and 2 is NP-hardness. And the fact that our restriction is applying to the broadest WMG involved in these control-by-candidates problems makes the restriction harsher than if it were applying to some sub-WMG, and so makes the results stronger.

Corollary 2 (Corollary to the proofs of Menton and Singh [29])

1. *Even when restricted to instances having all pairwise contests so equal that the total cardinality of the set of absolute values of WMG edges (in the WMG involving all candidates in the instance) is at most 2, Schulze elections are NP-complete (in the nonunique-winner model) for constructive control by deleting candidates. Even when restricted to instances having all pairwise contests so equal that the total cardinality of the set of values of WMG edges is at most 3, Schulze elections are NP-complete (in the nonunique-winner model) for constructive control by deleting candidates.*
2. *The same claims as in part 1 hold for constructive control by adding candidates, unlimited adding of candidates, partition of candidates in the ties-eliminate model, and runoff partition of candidates in the ties-eliminate model, except with the 2 and 3 above replaced by 3 and 5.*
3. *The same claims as in part 1 holds for constructive control by partition of candidates in the ties-promote model and runoff partition of candidates in the ties-promote model, except with the 2 and 3 above replaced by 4 and 7.*

6.4 Adding/deleting bound parameterization

For those control problems having as part of their inputs a limit on how many candidates or voters can be added/deleted, it is natural to consider parameterizing on that limit. This parameterization has been studied in some voting systems and the relevant problems have often been found to be W[1]-hard or W[2]-hard, and thus very unlikely to be fixed-parameter tractable. For example, under this parameterization, Betzler and Uhlmann [5] showed, for what are known as Copeland$^{\alpha}$ elections, that constructive control by adding candidates and constructive control by deleting candidates are W[2]-complete, Liu and Zhu [25] proved, for maximin elections, that constructive control by adding candidates is W[2]-hard, and Liu and Zhu also achieved W[1]-hardness results for the relevant voter control problems. For additional control results parameterized on the problem's internal addition/deletion limit, see Table 8 of Betzler et al. [4].

Hardness for these classes is defined in terms of fpt-reductions (whose definition can be found in Section 5.3). Thus one typically shows a problem is, for instance, W[2]-hard by providing such a reduction from a known W[2]-hard problem.

We observe that, with respect to parameterizing on the internal addition/deletion bound, for Schulze elections it holds that constructive control by adding voters, constructive control by deleting voters, and constructive control by adding candidates are all W[2]-hard.

Theorem 9 *Parameterized on the adding/deleting bound, each of*

1. *constructive control by adding voters,*
2. *constructive control by deleting voters, and*
3. *constructive control by adding candidates*

is W[2]-hard for Schulze elections, in the nonunique-winner model.

These problems have already been shown to be NP-complete by Parkes and Xia [31] and Menton and Singh [27] through reductions from exact cover by 3-sets (X3C). In order to prove W[2]-hardness, we need to reduce from a parameterized problem that is known to be W[2]-hard. One such problem is hitting set. Thus we modify the NP-hardness proofs given by Parkes and Xia to reduce from hitting set instead of from X3C.

All that is necessary is to replace the role of candidates modeling X3C elements with the role of candidates modeling hitting set sets, and to replace the role of voters modeling the X3C sets with the role of voters with the same preferences modeling hitting set elements. For the sake of completeness, we give the complete reduction for one of the above three cases, namely, control by adding candidates. First, we give the definition of hitting set.

Definition 7 (**Hitting Set**) Given a set of elements U, a collection $\mathcal{F}$ of subsets of U, and a positive integer k, does there exist $H \subseteq U$, with $\|H\| \leq k$, such that for every $S \in \mathcal{F}$, we have $S \cap H \neq \emptyset$ (i.e., H hits every set in $\mathcal{F}$)?

Proof of part 3 of Theorem 9 The standard NP-complete problem hitting set is also known to be W[2]-complete (see p. 464 of [9]). We give an fpt-reduction from hitting set to our problem.

Given a hitting set instance $(U, \mathcal{F}, k)$ as described in the definition, we will construct a control instance (C, D, V, p, k) where C is a set of original candidates, D is a set of auxiliary candidates, V is a set of voters, p is a distinguished candidate, and k is an adding bound. The original candidate set C will contain the following candidates.

- The distinguished candidate p.
- A candidate S for every $S \in \mathcal{F}$.

The auxiliary candidate set D will contain the following:

- A candidate u for every $u \in U$.

The voter set V will be as follows. We will not explicitly construct the entire voter set, but rather we will specify the weight of the WMG edges between the candidates and let the voter set be as constructed by McGarvey's method [26].

- For every $S \in \mathcal{F}$, $D(S, p) = 2$ (and so $D(p, S) = -2$).
- For every $u \in U$, $D(p, u) = 2$ (and so $D(u, p) = -2$).
- For every $u \in U$, and for every $S \in \mathcal{F}$ such that $u \in S$, $D(u, S) = 2$ (and so $D(S, u) = -2$).
- All WMG edges not set above will be of weight 0.

The distinguished candidate will be p and the adding limit will be the same limit k as in the hitting set instance. This completes the specification of the reduction. Note that initially—i.e., before any adding of candidates—the winners are exactly the "S" candidates (unless $\mathcal{F} = \emptyset$, in which case p is initially a winner in the control problem, and the hitting set instance is a positive instance, so in the case we are already done).

If we map from a positive hitting set instance, we claim that we will have a positive instance of the control problem. Why? Let $H \subseteq U$, $\|H\| \leq k$, be a solution to the hitting set instance. We will show that the set of candidates D' corresponding to the elements from H will be a solution to the control instance. First, $\|D'\| \leq k$, so we are within the adding bound. Also, since the hitting set solution includes members of every set in $\mathcal{F}$, there will be a path of strength two from p to each candidate corresponding to those sets, as including a candidate $u \in U$ creates paths from p to every "S" candidate hit by u. It is easy to see that p will have paths to every other candidate just as strong as they have back to p, and so p will be a Schulze winner.

If our reduction maps to a successful control instance, we claim it must have mapped from a positive hitting set instance. Why? Recall that we earlier handled the case $\|\mathcal{F}\| = 0$.

Initially, each "S" candidate will have a path of strength two to p, but p's strongest path to each of them is of strength negative two. Adding "u" candidates will leave those strength-two paths from each "S" candidate to p intact. Since all edges have weight at most two, regardless of how many candidates we add, no path can have a strength of more than two. So for a control instance to succeed it must give p a strength-two path to each "S" candidate. In our setting, that means adding a set of k "u" (auxiliary) candidates such that each "S" candidate is pointed to (with a weight-two edge) by at least one of them. However, given our current setup, that itself says that the hitting set instance being mapped from is a positive instance.

Note also that our reduction clearly runs in polynomial time in the size of the entire input, easily meeting the running-time limit for an fpt-reduction. Additionally, the parameter in the mapped-to instance will always be bounded by—in fact, identical to—the parameter in the mapped-from instance.

So this clearly is an fpt-reduction from hitting set to our problem. Thus we have established that constructive control by adding candidates, parameterized on the adding bound, is W[2]-hard for Schulze elections, in the nonunique-winner model. □

7 Weighted case

So far this paper has studied just manipulative-action problems without weights. However, weighted voting is quite common. In such settings as the US Electoral College, the US House of Representatives (on local issues such as water rights on which each state's representatives tend to vote as a block), and stockholder elections, the voting is weighted. Even beyond the human world, weighted voting can be very important. For example, in aggregating a number of subsystems' inputs as to what the best recommendation is on a certain query, some subsystems might be given more weight, perhaps based on a belief in their higher quality or track record.

So, although up to now in this paper our manipulation problems have been without weights and our bribery problems have been without weights or prices, we mention that (keep in mind we still are also parameterizing on the number of candidates, and that when weights and prices are used in problems, they are typically taken to be—and here we do take them to be—nonnegative integers) if one parameterizes by also bounding the maximum voter weight (if there are weights) and the maximum voter price (if there are prices), our main theorems hold even in the context of weights and prices. That is because when weights and prices are bounded, one can clearly still carry out the approach we use.

In fact, we can go slightly further, although at the outer edge of things doing so will require some surgery on our approach. Not just for the cases of bounded weights and prices, but even for the case where there is a bound on the *cardinality* of the set of weights (if there are weights) and there is a bound on the *cardinality* of the set of prices (if there are prices), all theorems (again, still parameterizing also on number of candidates) of Section 5 in bribery and manipulation still hold (as also do our Section 5 theorems on control, if one looks at control in the context of weighted votes; studying control in the context of weighted votes has only very recently been generally proposed [12], see also [24, 33]). To give an example of how we can handle this, we now state the result for bribery.

We note that both Theorem 1 and 2 follow from Theorem 10, which is not surprising as Theorem 1 in fact is building on and generalizing their approach. We include in this journal version those earlier proofs, since they provide a clearer, simpler setting in which to present

our proof approach. The detailed proof of the more flexible, difficult result that is Theorem 10 is provided in our technical report version [20], as is a detailed proof of Theorem 11.

Theorem 10 *For Schulze elections and for (with any feasible tie-breaking functions) ranked pairs, weighted bribery, priced bribery, and weighted, priced bribery each are in FPT with respect to the combined parameter "number of candidates" and "cardinality of the set of voter weights" (for cases with weights) and "cardinality of the set of voter prices" (for cases with prices), in our eight bribery models.*

Pushing beyond this, for the case of manipulation (still parameterized by the number of candidates), we can handle even the case of there not being any bound on the cardinality of the weight set of all the voters, but rather there simply being a bound on the cardinality of the set of weights over all *manipulative* voters.

Theorem 11 1. *For (with any feasible tie-breaking functions) ranked pairs, it holds that weighted coalitional manipulation is in FPT with respect to the combined parameter "number of candidates" and "cardinality of the manipulators' weight set," in our eight manipulation models.*
2. *The same claim holds for Schulze elections, except limited to the destructive case (in both the succinct and nonsuccinct input models, and in both the nonunique-winner model and the unique winner model) and the unique-winner constructive case (in both the succinct and nonsuccinct input models).*[6]

This still is all a valid framework for our many-uses-of-Lenstra-based approach (see our technical report version [20], which contains the omitted proofs from this paper). Note that for the case of the cardinality of the set of weights being 1, that gives the case of weighted noncoalitional manipulation mentioned as an aside by Dorn and Schlotter [8], though here we're handling even any fixed-constant number of manipulators (since any fixed-constant number have at most a fixed-constant cardinality of their weight set), and indeed, even a number of manipulators whose cardinality isn't bounded but who among them in total have a fixed-constant cardinality of occurring weights.

Recall that we already were proving our unweighted manipulation result (Theorem 6) nearly "for free," by drawing on a related control problem's groundwork (and in doing so, we were pointing out an interesting connection between manipulation and control). The strength of Theorem 11 is such that it gives us an alternative and even more "for free" path to seeing that Theorem 6 holds: Theorem 6 follows from Theorem 11 simply by setting all the weights to 1.

[6]The nonunique-winner model, constructive case for Schulze elections—which the above theorem is carefully avoiding claiming—also holds. But as noted in Section 4, a construction of Gaspers et al. [16] establishes that the weighted constructive coalitional manipulation problem for Schulze elections is in FPT, with respect to the parameter "number of candidates," in the nonunique-winner model. That is a broader result for nonunique-winner model, weighted constructive coalitional manipulation than would be the nonunique-winner model, constructive, Schulze version of Theorem 11, and so it would not make sense to include the nonunique-winner model, constructive Schulze case in the statement of Theorem 11. And the second paragraph of Footnote 6 of our technical report version [20] provides a detailed discussion of why Theorem 11's claim about the destructive Schulze case (and similarly, its unique-winner model, constructive Schulze case) seems not to be subsumed by or implicit in existing results, and mentions an interesting related open issue.

It seems intuitively necessary to bound the cardinality of the manipulator weight set to achieve results like the above. For ranked pairs we show that that is in some sense necessary: We prove as Theorem 12 that weighted constructive coalitional manipulation (with no bound on the number of manipulator weights) is NP-complete in ranked pairs for each fixed number of candidates starting at five. The significance of this result is that it shows that there cannot be any algorithm that is polynomial for any fixed number of (at least five) candidates, unless P = NP. As a weaker consequence, this will also block the existence of an FPT algorithm for this problem parameterized solely on the number of candidates.

It is natural to want to know at what number of candidates a problem becomes hard. This is an issue that has been intensely studied for a number of election systems, for example in the famous work of Conitzer, Sandholm, and Lang [7]. Similarly, one may naturally wonder whether the constant five is tight for the NP-hardness result we just mentioned. Theorem 13 establishes that five indeed is tight here, by showing that for each number of candidates smaller than five, we can solve the manipulation problem under consideration in polynomial time. That is, Theorems 12 and 13's significance is that they show the precise number of candidates where hardness appears, for the weighted constructive coalitional manipulation problem for ranked-pairs. The key idea behind the proof of Theorem 13 is to show that, for each ranked-pairs instance with fewer than five candidates, it holds that if there exists a successful manipulation, then for that same instance there exists a successful manipulation in which all the manipulators vote the same. Detailed proofs of Theorems 12 and 13 are provided in our technical report version [20].

Theorem 12 *For any feasible tie-breaking functions, and for each fixed number of candidates, j, $j \geq 5$, weighted constructive coalitional manipulation is NP-complete for ranked pairs (under the given feasible tie-breaking functions), in both the nonunique-winner model and the unique-winner model. Even without the assumption that the tie-breaking functions are feasible, NP-hardness will still hold.*

Theorem 13 *For any feasible tie-breaking functions, and for each number of candidates, j, $j < 5$, weighted constructive coalitional manipulation is solvable in polynomial time for ranked pairs (under the given feasible tie-breaking functions), in both the nonunique-winner model and the unique-winner model.*

8 Open problems

For all cases of (unweighted) bribery, control, and manipulation, including many where general-case hardness results exist, this paper proves that Schulze elections and ranked-pairs elections are fixed-parameter tractable with respect to the number of candidates.

In order to concisely represent our unweighted results, we list in Table 1 the best currently known tractability and intractability results regarding the unweighted case. All the figure's FPT results (and those results are parameterized with respect to the number of candidates) hold for both the constructive and destructive cases, in both the succinct and nonsuccinct input models, and for both the unique-winner and nonunique-winner models. Due to some of the other NPC cases not having been explored in the literature, though, Table 1 limits itself to speak just about the constructive case in the nonsuccinct input model and in the nonunique-winner model.

Table 1 Best current tractability and intractability results regarding the unweighted case of Schulze and ranked pairs, for constructive attacks in the nonsuccinct input model and for the nonunique-winner model. Our FPT results that are achieved by the looping-over-framework approach are shown in boldface. The FPT results' parameterization is with respect to the number of candidates

		Voting Rules	
		Schulze	Ranked Pairs
Bribery		**FPT** [Thm. 1], NPC [37]	**FPT** [Thm. 2], NPC [37]
Manipulation		P [16]	**FPT** [Thm. 6], NPC [38]
Control by	adding cand.	FPT [Thm. 7], NPC [31]	FPT [Thm. 7], NPC [31]
	deleting cand.	FPT [Thm. 7], NPC [28]	FPT [Thm. 7], NPC [31]
	partition of cand.	FPT [Thm. 7], NPC [28]	FPT [Thm. 7]
	adding voters	**FPT** [Thm. 3], NPC [31]	**FPT** [Thm. 3], NPC [31]
	deleting voters	**FPT** [Thm. 4], NPC [31]	**FPT** [Thm. 4], NPC [31]
	partition of voters	**FPT** [Thm. 5], NPC [28]	**FPT** [Thm. 5]

The most striking remaining open direction regards the weighted cases. For example, Gaspers et al. [16] proved that the constructive, nonunique-winner case of weighted coalitional manipulation is in FPT. Can their result be extended to the constructive, unique-winner case, or the destructive, unique-winner case, or the destructive, nonunique-winner case? These all remain open questions, although in Theorem 11 we give fixed-parameter tractability results for special cases of all three of these issues. The analogous issues are not open for ranked pairs: Theorem 12 shows that unless P = NP, the ranked-pairs analogue of the Gaspers et al. result cannot hold.

Acknowledgments An extended abstract of this paper appeared in the Proceedings of AAMAS 2013 [21]. We thank anonymous conference and journal referees for many helpful suggestions that much improved this paper, including the suggestion to replace our direct proof of Theorem 6 with one that establishes the result as a consequence of Theorem 3. This work was done in part while Curtis Menton was at the University of Rochester's Department of Computer Science. This work was supported in part by NSF grants CCF-0915792, CCF-1101479, CCF-1116104, and CNS-1319617.

Appendix

Control definitions

In this section, we provide more formal definitions of the benchmark control types. Since such papers as [2, 14, 19] provide a detailed discussion of the motivations of and real-world inspirations for the benchmark control types, we don't repeat the discussion here. The statements below are taken, sometimes identically, from Faliszewski et al. [14]. We first define the voter-control types, since those are more central in this paper. Then we define the candidate-control types, which in this paper appear only in Section 6. All definitions below are stated for the nonsuccinct, nonunique-winner model. Section 6's comments specify how to modify these to define the succinct case and/or the unique-winner case.

The adding-voters and deleting-voters control problems are the following.

Name: The constructive control by adding voters problem for $\mathcal{E}$ elections and the destructive control by adding voters problem for $\mathcal{E}$ elections.

Given: A set C of candidates, two disjoint collections of voters, V and W (each voter's vote is a tie-free linear order), a distinguished candidate p, and a nonnegative integer k.
Question (constructive): Is there a subset Q, $\|Q\| \leq k$, of voters in W such that the voters in $V \cup Q$ jointly elect $p \in C$ as a winner according to system $\mathcal{E}$?
Question (destructive): Is there a subset Q, $\|Q\| \leq k$, of voters in W such that the voters in $V \cup Q$ do not elect p as a winner according to system $\mathcal{E}$?

Name: The constructive control by deleting voters problem for $\mathcal{E}$ elections and the destructive control by deleting voters problem for $\mathcal{E}$ elections.
Given: A set C of candidates, a collection V of voters (each voter's vote is a tie-free linear order), a distinguished candidate $p \in C$, and a nonnegative integer k.
Question (constructive): Is it possible to by deleting at most k voters ensure that p is a winner of the resulting $\mathcal{E}$ election?
Question (destructive): Is it possible to by deleting at most k voters ensure that p is not a winner of the resulting $\mathcal{E}$ election?

As mentioned in Section 3.2, each partition problem has two variants, based on how ties are handled in the preliminary (i.e., first) round. In the ties-eliminate tie-handling model, if a first-round election does not have a unique winner then no one from that subelection moves forward to the second round. In the ties-promote tie-handling model, if a first-round election does not have a unique winner then all winners (if any) from that subelection move forward to the second round. This issue of which tie-handling rule is used is what is referred to in the definition below (and later, in the definitions of candidate partitioning) by the phrase "that survive the tie-handling rule."

Name: The constructive control by partition of voters problem for $\mathcal{E}$ elections and the destructive control by partition of voters problem for $\mathcal{E}$ elections.
Given: A set C of candidates, a collection V of voters (each voter's vote is a tie-free linear order), and a distinguished candidate $p \in C$.
Question (constructive): Is there a partition of V into V_1 and V_2 such that p is a winner of the two-stage election where the winners of election (C, V_1) that survive the tie-handling rule compete against the winners of (C, V_2) that survive the tie-handling rule? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.
Question (destructive): Is there a partition of V into V_1 and V_2 such that p is not a winner of the two-stage election where the winners of election (C, V_1) that survive the tie-handling rule compete against the winners of (C, V_2) that survive the tie-handling rule? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.

The unlimited-adding-candidates, adding-candidates, and deleting-candidates control problems are the following.

Name: The constructive control by adding an unlimited number of candidates problem for $\mathcal{E}$ elections and the destructive control by adding an unlimited number of candidates problem for $\mathcal{E}$ elections.
Given: Disjoint sets C and D of candidates, a collection V of voters (each voter's vote is a tie-free linear order over the candidates in the set $C \cup D$), and a distinguished candidate $p \in C$.
Question (constructive): Is there a subset E of D such that p is a winner of the $\mathcal{E}$ election with voters V and candidates $C \cup E$?
Question (destructive): Is there a subset E of D such that p is not a winner of the $\mathcal{E}$ election with voters V and candidates $C \cup E$?

Name: The constructive control by adding candidates problem for $\mathcal{E}$ elections and the destructive control by adding candidates problem for $\mathcal{E}$ elections.
Given: Disjoint sets C and D of candidates, a collection V of voters (each voter's vote is a tie-free linear order over the candidates in the set $C \cup D$), and a distinguished candidate $p \in C$, and a nonnegative integer k.
Question (constructive): Is there a subset E of D such that $\|E\| \leq k$ and p is a winner of the $\mathcal{E}$ election with voters V and candidates $C \cup E$?
Question (destructive): Is there a subset E of D such that $\|E\| \leq k$ and p is not a winner of the $\mathcal{E}$ election with voters V and candidates $C \cup E$?

Name: The constructive control by deleting candidates problem for $\mathcal{E}$ elections and the destructive control by deleting candidates problem for $\mathcal{E}$ elections.
Given: A set C of candidates, a collection V of voters (each voter's vote is a tie-free linear order), a distinguished candidate $p \in C$, and a nonnegative integer k.
Question (constructive): Is it possible to by deleting at most k candidates ensure that p is a winner of the resulting $\mathcal{E}$ election?
Question (destructive): Is it possible to by deleting at most k candidates other than p ensure that p is not a winner of the resulting $\mathcal{E}$ election?

The runoff-partition candidate control problems are the following. In runoff-partition candidate control problems, each candidate must participate in precisely one of the two first-round preliminary elections.

Name: The constructive control by runoff partition of candidates problem for $\mathcal{E}$ elections and the destructive control by runoff partition of candidates problem for $\mathcal{E}$ elections.
Given: A set C of candidates, a collection V of voters (each voter's vote is a tie-free linear order), and a distinguished candidate $p \in C$.
Question (constructive): Is there a partition of C into C_1 and C_2 such that p is a winner of the two-stage election where the winners of subelection (C_1, V) that survive the tie-handling rule compete against the winners of subelection (C_2, V) that survive the tie-handling rule? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.
Question (destructive): Is there a partition of C into C_1 and C_2 such that p is not a winner of the two-stage election where the winners of subelection (C_1, V) that survive the tie-handling rule compete against the winners of subelection (C_2, V) that survive the tie-handling rule? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.

The partition candidate control problems are the following. In partition candidate control problems, the partition is between candidates who participate in a first-round preliminary election and candidates who get a "bye" through the first round.

Name: The constructive control by partition of candidates problem for $\mathcal{E}$ elections and the destructive control by partition of candidates problem for $\mathcal{E}$ elections.
Given: A set C of candidates, a collection V of voters (each voter's vote is a tie-free linear order), and a distinguished candidate $p \in C$.
Question (constructive): Is there a partition of C into C_1 and C_2 such that p is a winner of the two-stage election where the winners of subelection (C_1, V) that survive the tie-handling rule compete against all candidates in C_2? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.

Question (destructive): Is there a partition of C into C_1 and C_2 such that p is not a winner of the two-stage election where the winners of subelection (C_1, V) that survive the tie-handling rule compete against all candidates in C_2? Each subelection (in both stages) is conducted using election system $\mathcal{E}$.

References

1. Bartholdi III, J., Tovey, C., Trick, M.: The computational difficulty of manipulating an election. Soc. Choice Welf. **6**(3), 227–241 (1989)
2. Bartholdi III, J., Tovey, C., Trick, M.: How hard is it to control an election? Math. Comput. Model. **16**(8/9), 27–40 (1992)
3. Betzler, N.: A multivariate complexity analysis of voting problems. Ph.D. thesis, Friedrich-Schiller-Universität Jena, Jena, Germany (2010)
4. Betzler, N., Bredereck, R., Chen, J., Niedermeier, R.: Studies in computational aspects of voting—A parameterized complexity perspective. In: The Multivariate Algorithmic Revolution and Beyond, 318–363. Springer-Verlag Lecture Notes in Computer Science #7370 (2012)
5. Betzler, N., Uhlmann, J.: Parameterized complexity of candidate control in elections and related digraph problems. Theor. Comput. Sci. **410**(52), 43–53 (2009)
6. Brill, M., Fischer, F.: The price of neutrality for the ranked pairs method. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, pp. 1299–1305. AAAI Press (2012)
7. Conitzer, V., Sandholm, T., Lang, J.: When are elections with few candidates hard to manipulate? J. ACM **54**(3) (2007). Article 14
8. Dorn, B., Schlotter, I.: Multivariate complexity analysis of swap bribery. Algorithmica **64**(1), 126–151 (2012)
9. Downey, R., Fellows, M.: Parameterized Complexity. Springer-Verlag (1999)
10. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: How hard is bribery in elections? J. Artif. Intell. Res. **35**, 485–532 (2009)
11. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Multimode control attacks on elections. J. Artif. Intell. Res. **40**, 305–351 (2011)
12. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Weighted electoral control. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, 367–374. International Foundation for Autonomous Agents and Multiagent Systems (2013)
13. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: The complexity of manipulative attacks in nearly single-peaked electorates. Artif. Intell. **207**, 69–99 (2014)
14. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Llull and Copeland voting computationally resist bribery and constructive control. J. Artif. Intell. Res. **35**, 275–341 (2009)
15. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer-Verlag (2006)
16. Gaspers, S., Kalinowski, T., Narodytska, N., Walsh, T.: Coalitional manipulation for Schulze's rule. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, 431–438. International Foundation for Autonomous Agents and Multiagent Systems (2013)
17. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. SIGACT News **38**(1), 31–45 (2007)
18. Hemaspaandra, E., Hemaspaandra, L., Menton, C.: Search versus decision for election manipulation problems. In: Proceedings of the 30th Annual Symposium on Theoretical Aspects of Computer Science, 377–388. Leibniz International Proceedings in Informatics (LIPIcs) (2013)
19. Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Anyone but him: The complexity of precluding an alternative. Artif. Intell. **171**(5–6), 255–285 (2007)
20. Hemaspaandra, L., Lavaee, R., Menton, C.: Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. Tech. Rep. arXiv:1301.6118 [cs.GT], Computing Res. Rep., arXiv.org/corr/ Revised June 2014 (2012)
21. Hemaspaandra, L., Lavaee, R., Menton, C.: Schulze and ranked-pairs voting are fixed-parameter tractable to bribe, manipulate, and control. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, 1345–1346. International Foundation for Autonomous Agents and Multiagent Systems (2013)

22. Cai, L., Chen, J., Downey, R., Fellows, M.: Advice classes of parameterized tractability. Annals of Pure and Applied Logic **84**(1), 119–138 (1997)
23. Lenstra Jr., H.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983)
24. Lin, A.: Solving hard problems in election systems. Ph.D. thesis, Rochester Institute of Technology, Rochester, NY (2012)
25. Liu, H., Zhu, D.: Parameterized complexity of control problems in maximin election. Inf. Process. Lett. **110**(10), 383–388 (2010)
26. McGarvey, D.: A theorem on the construction of voting paradoxes. Econometrica **21**(4), 608–610 (1953)
27. Menton, C., Singh, P.: Manipulation and control complexity of Schulze voting. Tech. Rep. arXiv:1206.2111v1 (version 1) [cs.GT], Computing Research Repository, arXiv.org/corr/ (2012)
28. Menton, C., Singh, P.: Control complexity of Schulze voting. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 286–292. AAAI Press (2013)
29. Menton, C., Singh, P.: Manipulation and control complexity of Schulze voting. Tech. Rep. arXiv:1206.2111 (version 4) [cs.GT], Computing Research Repository, arXiv.org/corr/ (2013)
30. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, London (2006)
31. Parkes, D., Xia, L.: A complexity-of-strategic-behavior comparison between Schulze's rule and ranked pairs. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, 1429–1435. AAAI Press (2012)
32. Rothe, J., Schend, L.: Challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey. Ann. Math. Artif. Intell. **68**(1–3), 161–193 (2013)
33. Russell, N.: Complexity of control of Borda count elections. Master's thesis, Rochester Institute of Technology (2007)
34. Schulze, M.: A new monotonic, clone-independent, reversal symmetric, and Condorcet-consistent single-winner election method. Soc. Choice Welf. **36**, 267–303 (2011)
35. Tideman, T.: Collective Decisions and Voting: The Potential for Public Choice. Ashgate Publishing (2006)
36. Wikipedia: Schulze method en.wikipedia.org/wiki/Schulze_method (2013)
37. Xia, L.: Computing the margin of victory for various voting rules. In: Proceedings of the 13th ACM Conference on Electronic Commerce, 982–999 (2012)
38. Xia, L., Zuckerman, M., Procaccia, A., Conitzer, V., Rosenschein, J.: Complexity of unweighted manipulation under some common voting rules. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence, 348–353 (2009)