Ann Math Artif Intell (2016) 77:225–250
DOI 10.1007/s10472-015-9478-2

The complexity of priced control in elections

Tomasz Miąsko[1] · **Piotr Faliszewski**[1]

Published online: 3 September 2015

Abstract We study the complexity of priced control in elections. Naturally, if a given control type is NP-hard for a given voting system $\mathcal{E}$ then its priced variant is NP-hard for this rule as well. It is, however, interesting what effect introducing prices has on the complexity of those control problems that without prices are tractable. We show that for four prominent voting rules (plurality, approval, Condorcet, and Copeland) introducing prices does not increase the complexity of control by adding/deleting candidates/voters. However, we do show an example of a scoring rule for which such an effect takes place.

Keywords Social choice · Voting · Control · Prices

Mathematics Subject Classifications (2010) 68Q17 · 91B14

1 Introduction

We consider the complexity of election control [2, 32] for the case where different control actions can have possibly different prices. Our main motivation comes from the fact that different types of control actions allowed in multimode control reflect a wide range of ways in which elections can be influenced through political campaigns, and prices reflect the fact that the cost of different actions varies. Our main finding is that introducing prices in control problems, typically, does not change their complexity. Specifically, we show that for several well-known voting rules (plurality, approval, Condorcet, and Copeland) the complexity of control problems with prices remains the same as for the unpriced variants (however, showing this requires more care). On the other hand, we show that for scoring protocols

✉ Piotr Faliszewski
faliszew@agh.edu.pl

[1] AGH University of Science and Technology, Krakow, Poland

destructive voter control is often easy, yet there is a scoring protocol for which destructive priced voter control is NP-hard. Our results stand in sharp contrast to those for control in weighted elections [23]. On one hand, allowing weighted votes often increases the complexity of control problems, and on the other, destructive weighted voter control for scoring protocols is always easy.

The individuals participating in an election, to whom we will refer as voters, might be, for example, members of parliaments, a jury, all adult citizens of a country, or even elements of distributed software systems [27, 35], or algorithms in various areas of computer science (we point to an application of voting related to natural language processing [34]). Voters select among possible alternatives, i.e., candidates taking part in the election. In the most frequently used, ordinal, model, a vote is a linear order over all the candidates, ranking them from the most to the least desirable one. However, under approval voting voters simply indicate which candidates they do and do not approve of. Once all the votes are gathered, we use a voting rule to determine the winner(s). There are many different voting rules to choose from, each with its own advantages and faults. For example, under the plurality rule each candidate receives a point from each voter that ranks him or her first, and the candidate(s) with most points win. Under Copeland elections, for each two candidates we form a head-to-head contest (that is, we check which of the two is preferred by a majority of the voters), the winner receives a point, and whoever has most points in the end is the winner. We formally introduce all the voting systems studied in this paper in Section 2. We focus on four rules, plurality, approval, Condorcet, and Copeland, that are widely studied from the point of view of the complexity of control. This makes it easy to compare our results with other ones in the literature.

Since elections are used to decide on matters of great importance among individuals with conflicting preferences, it is no surprise that many agents are interested in influencing their outcomes. There are two basic goals that such agents may be willing to attain: either they try to ensure that a preferred candidate is the winner of the election (a constructive action) or they try to preclude a despised candidate from achieving a victory (a destructive action). Further, there are many ways in which voters, candidates, and election organizers can influence elections results. These ways range from strategic voting [29, 47] (see the survey of Faliszewski and Procaccia [26] for an AI-focused overview), through bribery [20], to running political campaigns [15, 16] and performing control attacks [2, 32]. We focus on the latter two and we merge the ideas behind election control and campaign management.

By election control we mean actions that change the structure of an election. The most typical examples of control actions are adding/deleting candidates or voters. For example, it is easy to imagine settings where supporters of a particular candidate run a campaign to promote participation in an election, targeting the voters who are likely to vote for their candidate. Similarly, one can imagine actions discouraging opponent voters from casting their votes. A more difficult, yet possible, way of controlling an election is to fund a campaign of an additional candidate that would not otherwise take part in the election. Doing so could be motivated by a hope that such a candidate would steal votes away from our opponents.

The complexity of election control was first studied by Bartholdi, Tovey, and Trick [2] (we discuss related work in more detail in Section 1.1). However, they assumed that adding or deleting each candidate or voter has the same unit cost, which is not reasonable in the context of campaign management. Indeed, it might be very expensive to convince some candidate to join the race (e.g., because one would have to fund him or her completely), whereas convincing some other one might be quite cheap (e.g., because he or she already wants to join the election and is mostly prepared). Similarly, convincing some voters to vote

may be more expensive than convincing others (e.g., because for some we would have to pay for their transportation to the voting stations, whereas for others we would simply have to drop some leaflets in their neighborhood). Thus, in this paper we extend the model of control introduced by Bartholdi, Tovey, and Trick by allowing that different control actions have different prices. To simplify and shorten many of our proofs, we apply the multimode control framework of Faliszewski, Hemaspaandra, and Hemaspaandra [22].

1.1 Related work

Computational study of election control was initiated by Bartholdi, Tovey, and Trick [2], who defined the problems of constructive election control by adding/deleting/partitioning candidates or voters for plurality and Condorcet elections. Later Hemaspaandra, Hemaspaandra, and Rothe [32] extended their work by considering destructive variants of these problems, and by also studying the approval voting rule. Since these two papers, many researchers studied the complexity of control problems for various voting rules and in various other models. For example, Faliszewski et al. [24] considered the Copeland rule, Erdélyi et al. [18] studied Bucklin and fallback rules, and Parkes and Xia [44] studied the Schulze rule. This list, of course, is not exhaustive and is meant to present just a few examples (the reader may wish to consult the survey of Faliszewski, Hemaspaandra and Hemaspaandra for some more details [21]).

In addition to studying election control for different voting rules, researchers extended the standard model of election control in many different directions. For example, Meir et al. [42] studied election control in multiwinner voting and introduced a model that generalizes the idea of constructive and destructive control. Faliszewski, Hemaspaandra, and Hemaspaandra [22] studied multimode control, where it is possible to perform several different types of control actions at the same time (e.g., it is possible to add some candidates, and delete some voters; the standard control problems allow one to either only add candidates or only delete voters, etc.). Faliszewski, Hemaspaandra, and Hemaspaandra [23] were the first to study control in weighted elections (however the work of Baumeister et al. [5] is related to this topic). Other authors took a different perspective and, for example, studied parametrized complexity of control problems [6, 36, 37, 50], or considered the complexity of control in elections where votes come from some restricted domains (e.g., the single-peaked domain [8, 25] or the single-crossing domain [38]). On the other hand, Wojtas and Faliszewski [48] studied counting variants of control problems, where instead of asking if someone can become a winner we ask for the probability that someone becomes a winner, given that a random control action is taken. This counting variant of control can be used to predict election winners and, thus, has similar applications as the research presented in this paper; it aims to guide the election campaigning process. Recently, Bulteau et al. [11] and Chen et al. [12] considered combinatorial control, where adding each candidate or voter may result in adding some other ones as well. This variant of control also is useful from the point of view of campaign management. After all, if one convinces some person to join the election (either as a candidate or as a voter), this person might convince some more people.

While traditionally election control problems are limited to adding/deleting/partitioning candidates and voters, there are many problems that are very close in spirit to election control. For example, Chevaleyre et al. [13] studied a different setting where new candidates can appear, and Elkind, Faliszewski, and Slinko [17] studied candidate cloning. Research on election control has also affected research on related fields. For example, Baumeister et al. [3] studied control in judgment aggregation.

Our contribution to the computational study of control is to consider priced variants of control problems. Yet, the idea of studying priced variants of election problems has appeared earlier and is due to Faliszewski et al. [20], who introduced the problem of bribery in elections. In the bribery problem, we are given an election where each voter v has some price $\Pi(v)$, and we have available budget B. Upon paying the voter his or her price, we can change the voter's vote as we like. The question is if it is possible to ensure a given preferred candidate's victory without exceeding the available budget. Many algorithmic ideas of Faliszewski et al. [20] are useful in our setting as well. Bribery was later studied by various researchers [8, 22, 24, 39, 49], sometimes in settings other than those regarding elections [3, 40, 45] (these references are meant as examples only, see the survey of Faliszewski, Hemaspaandra, and Hemaspaandra [21] for some more discussion). In particular, Faliszewski [19] and Elkind et al. [15, 16] introduced the idea of voter prices that depend on the extent to which a given vote is affected. Among other motivations, they thought of using the bribery framework to model campaign management problems: A campaign manager working for some candidate can spend some effort on a particular voter to affect his or her preference order, but more significant changes require more effort than the less significant ones. Our view of modeling campaign management through priced control problems is inspired by these works on bribery. Bribery problems with prices dependent on the extent of the bribery were later studied by various other researchers [4, 9, 14, 41], also in the setting where a single bribery action can affect multiple voters [10].

1.2 Results

Given the above, very rough, overview of the literature on the complexity of election control, we see that there are three main lines of research regarding the topic. First, researchers seek complexity results for more and more different voting rules. Second, researchers seek to extend the election control model (e.g., by introducing weights, studying restricted domains, generalizing the notions of constructive/destructive control actions). Third, researchers apply the ideas from election control in other settings (e.g., in judgment aggregation).

Our paper follows the second line of research: We extend the model of election control by assuming that different control actions have possibly different costs. We focus on plurality, Condorcet, and approval voting rules, but we also include some other ones (e.g., Copeland and certain scoring protocols). We show that for each of these rules the complexity of control by adding/deleting candidates or voters is the same irrespective if we assume that all control actions have the same or possibly different costs. This result, however, is not trivial. Of course hardness proofs for the unit-cost model translate directly to hardness results in the model with varying costs, but easiness results do not. Indeed, sometimes we have to replace very simple greedy algorithms with more involved ones, sometimes using dynamic programming. We summarize our results in Table 1.

Our results for plurality, Condorcet, approval, and Copeland yield the question if adding prices can ever increase the complexity of control problems? We give an affirmative answer by showing a scoring protocol for which destructive voter control is easy, but for which destructive priced voter control is hard. This result answers our question, but leaves a bit more to be desired: Is there a natural voting rule with such a property? We leave this as an open problem.

The current paper, in some sense, complements that of Faliszewski, Hemaspaandra, and Hemaspaandra [23] regarding the complexity of control in weighted elections. However, our results are very different. They show that adding weights to control problems often increases the complexity of control, whereas this is not the case for adding prices. They

Table 1 Summary of priced control vulnerabilities. Our results are typeset in boldface. In the table, we write I to indicate that the voting rule is *immune* to a given control attack, i.e., that it is impossible to affect the result by exerting this type of control. By R we mean that the rule is *resistant*, i.e., that it is not immune and the election control problem is NP-hard. By V we mean that the voting rule is *vulnerable*, i.e., that it is not immune and that there is a polynomial-time algorithm for the election control problem

Control type	Adding Candidates		Deleting Candidates		Adding Voters		Deleting Voters	
Election system	Const. Control	Dest. Control	Const. Control	Dest. Control	Const. Control	Dest. Control	Const. Control	Dest. Control
Approval	I	**V**	**V**	I	R	**V**	R	**V**
Condorcet	I	**V**	**V**	I	R	**V**	R	**V**
Copeland	R	**V**	R	**V**	R	R	R	R
Plurality	R	R	R	R	**V**	**V**	**V**	**V**

show that destructive weighted voter control under scoring protocols is always easy, whereas we show that destructive priced voter control can sometimes be NP-hard (however, see the discussions in Section 5 for a possible explanation of this difference).

2 Preliminaries

In this section we review the voting rules that we study and formally define (priced) election control problems.

Elections and Election Rules. An election $E = (C, V)$ consists of a set of candidates $C = \{c_1, \ldots, c_m\}$ and a set of voters $V = \{v_1, \ldots, v_n\}$. (However, occasionally in our algorithms we will treat C and V as lists rather than sets. In such cases either the orders of the candidates/voters on the lists will be irrelevant or will be clearly specified.) Each voter is associated with this voter's preferences. The preferences are represented as strict total orders over the set of candidates. For example, if $C = \{a, b, c, d\}$ then some voter v_i might have preference order $d \succ c \succ b \succ a$, meaning that this voter likes d best, then c, then b, and finally he or she likes a least. For each election $E = (C, V)$ and each two candidates $c, d \in C$, we define $N_E(c, d)$ to be the number of voters in V who rank c above d.

An election rule (voting rule, election system, voting system) $\mathcal{E}$ is a function which given an election $E = (C, V)$ maps it to the set of winners $\mathcal{E}(E) \subseteq C$. In this paper we focus on polynomial-time computable voting rules but, in general, determining election winners can be a much more computationally demanding problem.[1] There can be more than one winner of an election. In such situations, to emphasize this fact, we refer to the winners of the election as the nonunique winners. Similarly, if there is only one winner of an election we refer to him or her as the unique winner. Finally, we allow situations where an election has no winners.

[1] For example in a voting scheme suggested by Lewis Caroll checking if a distinguished candidate is a winner of an election is NP-hard [1] and, indeed, is complete for parallel access to NP [31]. The same holds for the systems of Young [46] and Kemeny [33] (for the latter one, NP-hardness was first shown by Bartholdi, Tovey, and Trick [1]).

There are many different voting rules. For example, under the plurality rule we give a point to each candidate that is ranked first, and choose as winners those candidates that have the highest number of points. More generally, a scoring rule is defined through a family of scoring vectors, one for each candidate-set cardinality, that define how many points a candidate receives for being ranked at a given position by a voter. Formally, a scoring vector (for an m-candidate election) is an m-tuple $\alpha = \langle \alpha_1, \ldots, \alpha_m \rangle$ of nonnegative integers given in nonincreasing order. For each vote where a candidate is ranked i'th, the candidate receives α_i points. The candidates that have the highest number of points are the winners. For example, plurality is defined through a family of scoring vectors of the form $\langle 1, 0, \ldots, 0 \rangle$. Other interesting scoring protocols include, e.g., the veto rule (defined through vectors of the form $\langle 1, \ldots, 1, 0 \rangle$), the k-approval rule (defined through vectors that start with k ones and then continue with zeros), the k-veto rule (defined through vectors that end with k zeros, preceded by ones), and the Borda rule (defined through vectors of the form $\langle m-1, m-2, \ldots, 0 \rangle$).

In the Condorcet method, a candidate c is a winner if he or she beats all the other candidates in head-to-head contests (i.e., if $N_E(c, d) > N_E(d, c)$ for all candidates d different from c). It is possible that there is no winner under Condorcet rule, but if there is one, he or she is unique.

The Copeland rule is an extension of the Condorcet rule in the sense that it elects the Condorcet winner whenever it exists, and otherwise picks those candidates that are closest to being Condorcet winners in a certain way. Formally, for each rational α, $0 \leq \alpha \leq 1$, in the Copeland$^\alpha$ voting rule, candidate c receives one point for each candidate d, $d \neq c$, such that $N_E(c, d) > N_E(d, c)$, and α points for each candidate d, $d \neq c$, such that $N_E(c, d) = N_E(d, c)$. Candidates with the highest number of points are the winners. Naturally, there are many other rules that can be seen as extensions of the Condorcet rule (for example, the maximin rule, the Young rule, the Kemeny rule, the Dodgson rule; see, e.g., the overview of Brams and Fishburn [7]). However, among this type of rules, in this paper we focus on the Copeland rule.

Finally, we also consider the approval voting rule. Under approval, voters' preferences are represented differently. Instead of ranking the candidates, each voter provides a set of candidates that he or she approves of. A candidate receives a point for each voter that approves of him or her. As before, the candidates with the highest number of points are the winners.

We denote the score of a candidate c in election E by $\text{score}_E(c)$ (the actual voting rule will always be clear from the context). When the election E is clear from the context, we sometimes write $\text{score}(c)$ instead of $\text{score}_E(c)$. Further, for an election $E = (C, V)$ and candidates $c, d \in C$, we write $\text{diff}_E(c, d)$ to mean the difference between the score of candidate c and the score of candidate d. For the case of the Condorcet rule, by $\text{diff}_E(c, d)$ we mean $N_E(c, d) - N_E(d, c)$.

Election Control Problems. We consider priced multimode control problems. In control problems an attacker tries to execute a basic control action such as candidate addition, candidate deletion, voter addition, or voter deletion to change the result of an election. In priced multimode control problems several different types of basic control actions can be combined into a single attack. Moreover, each such action has associated price and the person exercising control over the election has a limited budget.

We assume that there is a price tag for each voter and candidate that we add or delete. That is, for each voter v that can be added or deleted, we have a nonnegative integer $\Pi(v)$, the price of adding/deleting v. For each candidate c that can be added or deleted, we have

a nonnegative integer $\Pi(c)$, the price of adding/deleting c. To simplify notation, if we consider, e.g., some set W of voters, we write $\Pi(W)$ to mean $\sum_{v \in W} \Pi(v)$. We use analogous notation for candidate sets. Unless stated otherwise, we assume that the prices are encoded in binary.

With this notation available, we define the most general form of our control problem.

Name: constructive/destructive $\mathcal{E}$-AC+DC+AV+DV-priced-control.
Given: An election (C, V), a candidate $c \in C$, a set of additional candidates D, such that $C \cap D = \emptyset$, a set of additional voters W, prices Π for candidates in $C \cup D$ and voters in $V \cup W$, and a natural number K (the budget).
Question (constructive): Are there subsets $C' \subseteq C$, $D' \subseteq D$, $V' \subseteq V$, $W' \subseteq W$, such that candidate c is the unique winner of $\mathcal{E}$ election $\big((C \setminus C') \cup D', (V \setminus V') \cup W'\big)$ and $\Pi(C' \cup D') + \Pi(V' \cup W') \leq K$.
Question (destructive): Are there subsets $C' \subseteq C \setminus \{c\}$, $D' \subseteq D$, $V' \subseteq V$, $W' \subseteq W$, such that candidate c is not the unique winner of $\mathcal{E}$ election $\big((C \setminus C') \cup D', (V \setminus V') \cup W'\big)$ and $\Pi(C' \cup D') + \Pi(V' \cup W') \leq K$.

This definition calls for some comments. We note that subset C' is the set of candidates to be removed from the election and subset D' is the set of candidates to be added to the election. Analogously, V' is the set of deleted voters and W' is the set of added voters. The total price of such a control action is the sum of the prices of added/deleted candidates and voters. That is, the total price is $\Pi(C' \cup D') + \Pi(V' \cup W')$. For the case where all the prices are equal to one, we refer to the above problem as $\mathcal{E}$-AC+DC+AV+DV-control (omitting the word "priced").

We point out that even though we follow the idea of multimode control of Faliszewski, Hemaspaandra, and Hemaspaandra [22], we slightly differ from their approach. Indeed, they have a separate "budget" for each control type, whereas we have a single parameter K that models the total budget. This matches our motivating example of campaign management better. If one is running a campaign, there is a single budget that can be partitioned between various activities in any convenient way.

We use the unique-winner model. That is, to be successful, a candidate has to be the only winner of the election. Both the unique-winner model and the nonunique-winner model are frequently studied in election control literature. While occasionally the choice of the particular model matters heavily, this is not the case for this work. Nonetheless, for some of our results we cite papers that focus on the unique winner model and, thus, it is more convenient for us to focus on this model as well.

In the constructive cases, we will often speak of the distinguished candidate c as the *preferred* candidate and thus we will often denote him or her with p rather than with c. For the destructive cases, we will refer to this candidate as the *despised* one and often use d to denote him or her.

We are often interested in subproblems of $\mathcal{E}$-AC+DC+AV+DV-priced-control where only some nonempty subset of basic control actions, AC (adding candidates), DC (deleting candidates), AV (adding voters), and DV (deleting voters), is available. We denote such subproblems by leaving only relevant parts of the input and appropriately modifying the question part of the problem. (Intuitively, one could also think that all the disallowed control actions have prices higher than the available budget.) Names of such control problems are formed from the name of the voting system $\mathcal{E}$, followed by the permitted basic control actions, where all parts are separated with the "+" character. For example, if we studied

priced control by adding candidates and deleting voters, then the problem's name would be $\mathcal{E}$-AC+DV-priced-control.

We say that a voting system is *susceptible* to constructive control problem $\mathcal{C}$ if for some instance of this control problem with an election E, a preferred candidate is not a unique winner of the election E, but it is possible to exercise control $\mathcal{C}$ over election E in such a way as to make the preferred candidate the unique winner. Similarly, in the case of destructive control, a voting system is susceptible to control if we can prevent the despised candidate from being the unique winner (and he or she had not been the unique winner before). A voting system is said to be *immune* to control if it is not susceptible to it. If a voting system is susceptible to control and the associated decision problem is in P, then we say that the voting system is *vulnerable* to this type of control. If a voting system is susceptible to control and the associated decision problem is NP-hard, we say it is *resistant* to this type of control.

The main goal of this paper is to establish the complexity of priced control by adding or deleting candidates or voters for the plurality rule, the approval rule, the Condorcet rule, and the Copeland rule. We focus on the problems where only a single type of control is allowed, but we use the expressive power of multimode control to simplify and compress our proofs.

Computational Complexity. We assume that the reader is familiar with basic notions of complexity theory such as classes P and NP, polynomial-time many-one reductions, and the notions of NP-hardness and NP-completeness (see, e.g., the textbook of Papadimitriou [43]). However, most of the proofs in this paper present polynomial-time algorithms.

3 Sometimes prices do not affect the complexity of control

In this section we study priced control under the plurality, approval, Condorcet and Copeland rules, using the multimode control framework. Naturally, introducing prices cannot make our control problems easier and, indeed, the following easy proposition holds.

Proposition 3.1 For each voting rule $\mathcal{E}$ and each control type $\mathcal{C}$, it holds that constructive (destructive) $\mathcal{E}$-$\mathcal{C}$-control polynomial-time many-one reduces to constructive (destructive) $\mathcal{E}$-$\mathcal{C}$-priced-control.

Proof Given an instance of constructive (destructive) $\mathcal{E}$-$\mathcal{C}$-control, we output an instance of $\mathcal{E}$-$\mathcal{C}$-priced-control that is identical except that, in addition, all the candidates/voters that can be added/deleted are associated with the same unit price. □

Thus all the hardness results for the unpriced control problems hold in the priced setting. The main message of this section is that all the existing vulnerability results for adding/deleting candidates or voters for our four voting rules do carry through to the setting with prices as well. In particular, we show that plurality is vulnerable to constructive and destructive AV+DV-priced-control, approval and Condorcet are vulnerable to destructive AC+AV+DV-priced-control and to constructive DC-priced-control, and Copeland is vulnerable to destructive AC+DC-priced-control. Is it the case that adding prices never increases the complexity of control problems? In Section 4 we show that there are scoring protocols for which considering prices does make a difference in terms of the complexity of control problems.

In the following sections we present our results for the plurality, approval, Condorcet, and Copeland rules. We remark that we phrase our results in full generality, using the multimode

control framework. However, of course, our results that regard several control types at the same time carry through to settings with fewer control types. For example, the fact that the Condorcet rule is vulnerable to destructive AC+AV+DV-priced-control means that it is also vulnerable to destructive variants of each of AC-priced-control, AV-priced-control, and DV-priced-control (this follows naturally from the definition because we can set up the prices for the disallowed control actions to be above the allowed budget; a very similar result is given as Proposition 4.9 by Faliszewski, Hemaspaandra, and Hemaspaandra [22]).

3.1 Plurality rule

We start by considering the plurality rule. Bartholdi, Tovey, and Trick [2] have shown that plurality is resistant to constructive control by adding/deleting candidates, but that it is vulnerable to constructive control by adding/deleting voters. Hemaspaandra, Hemaspaandra, and Rothe [32] have shown that the same results hold for the destructive variants of these problems. We extend the vulnerability results to the priced case by showing that Plurality-AV+DV-priced-control is in P both in the constructive and in the destructive case (for the case without prices, this has already been done by Faliszewski et al. [22]).

Plurality is a very simple rule. If a new vote is added to the election, then the score of the candidate who is ranked first in this vote is increased by one, while the scores of all the other candidates remain intact. Similarly, when deleting a single vote from the election, only the score of one candidate is affected. This locality property makes it possible to construct greedy algorithms for Plurality-AV+DV-priced-control. Our algorithms are natural extensions of those for the unpriced setting.

Theorem 3.2 *Constructive Plurality-*AV+DV*-priced-control is in P.*

Proof Input to the constructive Plurality-AV+DV-priced-control problem consists of an election $E = (C, V)$, a preferred candidate $p \in C$, a set of additional voters W, a list of prices Π associated with the voters from $V \cup W$, and a natural number K (the available budget). We give a greedy algorithm which in each step decreases the value of $\max_{c \in C \setminus \{p\}} \mathrm{diff}_E(c, p)$ by one, and halts either when we make p the unique winner of the election, or the available budget is exceeded, or there are no more votes to add/remove.

Our algorithm proceeds as follows. If p is already the unique winner, then accept. Otherwise keep executing one of the following actions, until either p becomes the unique winner (and then accept), there are no more actions that can be executed (and then reject), or we exceed the budget (and then reject). There are two possible actions, the one with lower cost is selected and executed:

1. From the set of additional votes that rank p first, pick a vote w which has not been added to the election yet and which has minimal price $\Pi(w)$. The cost of this action is $\Pi(w)$. If this action is executed, add w to the election.
2. For each candidate c in $\arg\max_{c \in C} \mathrm{diff}_E(c, p)$, pick a vote $v \in V$ that ranks c first, that has not already been deleted from the election, and that has minimal price among such votes. Let U be the collection of the picked votes. The cost of this action is $\Pi(U)$. If this action is executed, all the votes from U are deleted from the election.

The algorithm's pseudocode is presented on Fig. 1. The correctness is straightforward to see, and it is also easy to see that it runs in polynomial time. Let N be the total number of voters in V and W. The while-loop in the algorithm executes at most $O(N)$ times (because in every iteration p decreases the difference between its score and the score of the current

PLURALITY-AV+DV-CONSTRUCTIVE-CONTROL$((C, V), p, W, \Pi, K)$

$E \leftarrow (C, V)$
$V' \leftarrow W' \leftarrow \emptyset$
while p is not the unique winner of E and $(V \setminus V') \cup (W \setminus W') \neq \emptyset$ **do**
  $w \leftarrow$ a vote for p with minimal price from $W \setminus W'$ (if it exists)
  $U \leftarrow$ votes for candidates in $\arg\max_{c \in C} \mathrm{diff}_E(c, p)$ with minimal price, one for each candidate (if this set exists)
  if w is defined **and** (U is undefined **or** $\Pi(w) \leqslant \Pi(U)$) **then**
    $W' \leftarrow W' \cup \{w\}$
    $K \leftarrow K - \Pi(w)$
  else if U is defined **then**
    $V' \leftarrow V' \cup U$
    $K \leftarrow K - \Pi(U)$
  else reject
  $E \leftarrow (C, (V \setminus V') \cup W')$
  if $K < 0$ **then** reject
accept

Fig. 1 The algorithm for constructive Plurality-AV+DV priced control problem

election winners by one; in the worst case p initially has score 0 and the original election winners have score N). Each iteration requires polynomial time to compute w and U (a single linear scan through V and W suffices to achieve this, provided that throughout the execution of the algorithm we maintain the scores of those candidates who are ranked first by at least one voter each). □

In constructive Plurality-AV+DV-priced-control we have to ensure that the preferred candidate's score is higher than the scores of all the remaining candidates. On the other hand, in destructive control we only have to ensure that there exists at least one candidate with score equal to or higher than the score of the despised candidate. This suggests a simple algorithm enumerating all candidates and checking if one of them can beat or tie the despised one. As before, this is a natural extension of the algorithms for the unpriced setting.

Theorem 3.3 *Destructive plurality-*AV+DV*-priced-control is in P.*

Proof Input to the destructive Plurality-AV+DV-priced-control instance consists of an election $E = (C, V)$, a despised candidate $d \in C$, a set of additional voters W, a list of prices associated with voters $V \cup W$, and an available budget $K \in \mathbb{N}$. For each candidate $c \in C \setminus \{d\}$ we create a list of votes from V where d is ranked first, which we could delete from the election, and a list of votes in W where c is ranked first, which we could add to the election. We merge these lists together, sort them in the order of increasing prices, and take up to the first $\max(0, \mathrm{diff}_E(d, c))$ votes, i.e., the number of votes that creates a tie between candidate d and candidate c. If there are sufficiently many votes and their total price does not exceed the available budget K, then we accept. Otherwise if this condition is not fulfilled for any candidate $c \in C \setminus \{d\}$, we reject. Pseudocode for this algorithm is presented on Fig. 2. It is straightforward to see that the algorithm runs in polynomial time; for each candidate c one has to perform a simple set of polynomial-time operations. □

PLURALITY-DESTRUCTIVE-CONTROL$((C, V), d, W, \Pi, K)$

```
1  E ← (C, V)
2  if d is not the unique winner of E then
3  |  accept
4  for c ∈ C \ {d} do
5  |  V' ← list of voters from V who rank d first
6  |  W' ← list of voters from W who rank c first
7  |  L ← list V' ∪ W' sorted in the order of increasing prices
8  |  truncate L to contain at most first max(0, diff_E(d, c)) votes
9  |  if diff_E(d, c) ⩽ ‖L‖ and Π(L) ⩽ K then
10 |  |  accept
11 reject
```

Fig. 2 The algorithm for destructive Plurality-AV+DV-priced-control problem

Thus, for the case of plurality, introducing prices is seamless; we can adjust the existing greedy algorithms in a simple way. We believe that this is a very positive result. Priced control problems are more realistic and it is convenient that considering prices comes at essentially no additional cost in terms of computational complexity.

3.2 Approval and condorcet rules

Let us now move on to the case of approval and Condorcet. We extend the results of Faliszewski, Hemaspaandra and Hemaspaandra [22] (who themselves relied on the results of Bartholdi, Tovey, and Trick [2] and Hemaspaandra, Hemaspaandra, and Rothe [32]), who have shown that the approval and Condorcet rules are vulnerable to destructive AC+AV+DV-control, to apply to priced control. We show that approval and Condorcet are vulnerable to destructive AC+AV+DV-priced-control as well. We do not consider destructive priced control by deleting candidates because approval and Condorcet are immune to this type of control (and adding prices makes no difference with respect to immunity). On the other hand, we show that approval and Condorcet are vulnerable to constructive DC-priced-control. Naturally, through Proposition 3.1 and the results of Bartholdi, Tovey, and Trick [2] and Hemaspaandra, Hemaspaandra, and Rothe [32], approval and Condorcet are resistant to constructive voter priced control problems, and are immune to constructive priced control by adding candidates.

The reader may wonder why we consider the approval and Condorcet rules jointly. The reason is that both approval elections and Condorcet elections can be understood in terms of the results of head-to-head contests between candidates. By head-to-head contests we mean elections where only two candidates are present. To facilitate this approach we adopt the following convention: we say that candidate c is preferred to candidate d in an election with voter set V if and only if $\mathrm{diff}_{(\{c,d\},V)}(c, d) > 0$ (recall that for the approval rule $\mathrm{diff}_E(c, d)$ means the difference of scores of candidates c and d in election E, whereas for Condorcet it means the value $N_E(c, d) - N_E(d, c)$).

With this notation available, we see that a candidate is the unique winner of an approval election or of a Condorcet election if and only if he or she is the unique winner of all the head-to-head contests with the other candidates. Thus to prevent a despised candidate d from being a unique winner, we have to ensure that another candidate beats or ties the

despised candidate in their head-to-head contest. The despised candidate's loss to some candidate c can be achieved by introducing voters who prefer c to d, or by deleting voters who prefer d to c. Each such added or deleted voter introduces the same change in the score difference (difference in number of approvals) between candidate c and the despised candidate. This observation suggest a simple algorithm based on enumeration of candidates who might ensure the despised candidate's defeat, combined with a greedy approach to selecting the votes relevant to the head-to-head contest with the despised candidate.

Theorem 3.4 *Approval voting and Condorcet voting are vulnerable to destructive* AC+AV+DV*-priced-control.*

Proof In a destructive AC+AV+DV-priced-control instance we are given an election $E = (C, V)$, a despised candidate $d \in C$, a set of additional candidates D, a set of additional voters W, prices Π associated with the candidates in D and the voters in $V \cup W$, and an available budget $K \in \mathbb{N}$.

If candidate d already is not a unique winner of election E then control is successful and we accept. Otherwise, for each candidate $c \in (C \cup D) \setminus \{d\}$, we create a list L containing those voters from V, where d is preferred to c, and those voters from W, where c is preferred to d. We sort L in the order of increasing prices, and limit it to the first up to $\mathrm{diff}_E(d, c)$ votes. In our control action we delete the votes from V that are in L, and add the votes from W that are in L. Therefore the total price of control action is the sum over:

1. The prices associated with the voters added to and removed from the election.
2. The price of adding c to the election, if $c \in D$.

If there are enough votes to create a tie between candidate c and candidate d, and the total cost does not exceed K, accept. Otherwise, repeat this procedure for all the remaining candidates. If control is not possible for any candidate $c \in (C \cup D) \setminus \{d\}$, reject. The final algorithm is presented on Fig. 3.

The algorithm runs in polynomial time. The outer loop is executed at most $O(\|C\|)$ times and each iteration is easily seen to be polynomial-time computable. □

In the constructive setting, approval and Condorcet are vulnerable to control by deleting candidates only. (They are immune to control by adding candidates and resistant to voter control [2, 32].) We extend this result to the priced setting. Clearly, to make the preferred candidate win, all the candidates that defeat him or her in their head-to-head contest should be deleted. Furthermore, as deleting candidates does not affect in any way the results of head-to-head contests, it is a necessary and sufficient condition. There is a single optimal control action.

Theorem 3.5 *Approval voting and Condorcet voting are vulnerable to constructive* DC*-priced-control.*

Proof In constructive DC-priced-control instance we are given an election $E = (C, V)$, a preferred candidate $p \in C$, a list of prices associated with the candidates in C, and an available budget K. Candidate p is the unique winner of the election E if and only if he or she beats all remaining candidates in their head-to-head contests. Therefore, if the total price necessary to delete all the candidates who tie or beat the preferred candidate p in their head-to-head contests is within budget, then accept, otherwise reject. Pseudocode is presented in Fig. 4. The algorithm runs in polynomial time; computing $\mathrm{diff}_E(p, c)$ requires

```
APPROVAL-CONDORCET-DESTRUCTIVE-AC+AV+DV-CONTROL((C, V), d, D, W, Π, K)
E ← (C, V)
if d is not the unique winner of E then
    accept
for c ∈ (C ∪ D) \ {d} do
    V' ← list of those voters from V who prefer d to c
    W' ← list of those voters from W who prefer c to d
    L ← list V' ∪ W' sorted in the order of increasing prices
    truncate L to contain at most first diff_E(d, c) votes
    K' ← Π(L)
    if c ∈ D then
        K' ← K' + Π(c)
    if diff_E(d, c) ⩽ ‖L‖ and K' ⩽ K then
        accept
reject
```

Fig. 3 The algorithm for destructive AC+AV+DV-priced-control in Approval voting and Condorcet voting

a single scan over the whole profile (provided one computes $\mathrm{diff}_E(p, c)$ simultaneously for all the candidates from C) and the rest of the algorithm is straightforward. □

Again, introducing prices does not make the control problems significantly harder for the approval and Condorcet rules. It is easy and natural to extend existing greedy algorithms to take prices into account. As we will see in the next section, the case of Copeland is somewhat more involved.

3.3 Copeland rule

Faliszewski et al. [24] have studied Llull and Copeland voting rules and have shown that Copeland$^\alpha$ fully resists constructive control, resists destructive AV and DV control, but is vulnerable to both destructive AC and DC control. These vulnerability results have been combined into destructive AC+DC control vulnerability by Faliszewski, Hemaspaandra, and Hemaspaandra [22]. Here we extend this result to the priced control framework.

In the following theorem we extend the algorithm of Faliszewski, Hemaspaandra, and Hemaspaandra [22, Theorem 4.10] to the case of destructive AC+DC-priced-control, for the case of Copeland^0 and Copeland^1 rules. Then we explain why it does not work for Copeland$^\alpha$ for all rational α values, α, $0 < \alpha < 1$. Finally, in Theorem 3.8, we provide an algorithm which does work for all rational values of α.

```
APPROVAL-CONDORCET-CONSTRUCTIVE-DC-CONTROL((C, V), p, Π, K)
E ← (C, V)
C' ← {c ∈ C \ {p} | diff_E(p, c) ⩽ 0}
if Π(C') ⩽ K then
    accept
else
    reject
```

Fig. 4 The algorithm for constructive DC-priced-control in Approval voting and Condorcet voting

Theorem 3.6 *Destructive* AC+DC*-priced-control is in P for Copeland*^{0} *and Copeland*^{1} *voting.*

Proof In a destructive Copeland$^\alpha$-AC+DC-priced-control instance, where α is in $\{0, 1\}$, we are given an election $E = (C, V)$, a despised candidate $d \in C$, a set of additional candidates D, a list of prices Π associated with the candidates in $C \cup D$, and an available budget $K \in \mathbb{N}$. To preclude despised candidate d from being the unique winner of the election, we need to ensure that the score of another candidate from $C \cup D$, call him or her p (we try each possible choice of p), is higher or equal to the score of d, i.e., $\mathrm{diff}_E(d, p) \leq 0$. The score of a candidate in a Copeland$^\alpha$ election is the sum of his or her scores in head-to-head contents with the remaining candidates:

$$\mathrm{score}_{(C,V)}(d) = \sum_{c \in C \setminus \{d\}} \mathrm{score}_{(\{c,d\},V)}(c).$$

For each candidate c, define $\mathrm{gain}_{p,d}(c)$ to be the score difference that candidate p gains relative to the despised candidate d, if candidate c is part of the control action (assuming that p participates in the election):

$$\mathrm{gain}_{p,d}(c) = \begin{cases} \mathrm{score}_{(\{c,d\},V)}(d) - \mathrm{score}_{(\{c,p\},V)}(p), & c \in C, \\ \mathrm{score}_{(\{c,p\},V)}(p) - \mathrm{score}_{(\{c,d\},V)}(d), & c \in D. \end{cases}$$

It is easy to see that for Copeland^0 and Copeland^1, for each candidate c, $\mathrm{gain}_{p,d}(c)$ is either -1, 0 or 1. Moreover, as our goal is to decrease d's advantage over p, we are only interested in candidates with positive gain. Consequently, the following greedy approach can be used to select candidates to add or delete. From $C \setminus \{d, p\}$ and $D \setminus \{p\}$ select a list L of candidates with positive gain. Sort L in the order of increasing prices. Take first up to $\max(0, \mathrm{diff}_E(d, p))$ candidates from L and if there was a sufficient number of them, then $A = L \cup (D \cap \{p\})$ describes a successful control action. If the total price of control action A is within budget K then accept, otherwise repeat this procedure for another choice of candidate p. The final algorithm is presented on Fig. 5.

The algorithm runs in polynomial time. Computing the scores of all the candidates and all the gain values requires a scan over every vote for each pair of candidates. The main loop executes $O(\|C \cup D\|)$ times and each iteration requires simple polynomial-time operations. □

The above algorithm relies on the fact that all the candidates that we add or remove from the election introduce the same score difference between the despised candidate and our chosen candidate p. This is a crucial element ensuring correctness of the greedy approach. In Copeland$^\alpha$ for some rational α, $0 < \alpha < 1$, the score difference could be 1, α or $1 - \alpha$. This makes the candidates incomparable and this greedy approach infeasible.

To facilitate our dynamic programming solution, we reformulate Copeland$^\alpha$ into voting system Copeland$^{x,y}_{\mathbb{N}}$ that admits only natural numbers as scores.

Definition 3.7 Let x, y be two nonnegative integers. We define voting rule Copeland$^{x,y}_{\mathbb{N}}$ as follows. Given an election $E = (C, V)$, each candidate c receives the following score:

$$\begin{aligned} \mathrm{score}_E(c) = {} & x \ \|\{d \in C \setminus \{c\} \mid N_E(c, d) > N_E(d, c)\}\| \\ & + y \ \|\{d \in C \setminus \{c\} \mid N_E(c, d) = N_E(d, c)\}\| \end{aligned}$$

The candidates with the highest score are the winners.

COPELAND01-DESTRUCTIVE-AC+DC-CONTROL$((C, V), d, D, \Pi, K)$

```
1  E ← (C, V)
2  if d is not a unique winner of E then
3  |  accept
4  for p ∈ (C ∪ D) \ {d} do
5  |  C' ← {c ∈ C \ {p, d} | gain_{p,d}(c) > 0}
6  |  D' ← {c ∈ D \ {p} | gain_{p,d}(c) > 0}
7  |  L ← list C' ∪ D' sorted in the order of increasing prices
8  |  truncate L to contain at most first diff_E(d, c) candidates
9  |  K' ← Π(L)
10 |  if p ∈ D then
11 |  |  K' ← K' + Π(p)
12 |  if diff_E(d, p) ⩽ ‖L‖ and K' ⩽ K then
13 |  |  accept
14 reject
```

Fig. 5 The algorithm for destructive AC+DC-priced-control in Copeland^0 and Copeland^1 voting

It is easy to see that for each rational α, $0 \leq \alpha \leq 1$, Copeland$^\alpha$ election is equivalent to the Copeland$^{x,y}_{\mathbb{N}}$ election where $\alpha = y/x$, for some $x, y \in \mathbb{N}$.

Theorem 3.8 *For each $x, y \in \mathbb{N}$, destructive Copeland$^{x,y}_{\mathbb{N}}$-AC+DC-priced-control is in P.*

Proof The input to the destructive Copeland$^{x,y}_{\mathbb{N}}$-AC+DC-priced-control problem consists of an election $E = (C, V)$, a set of additional candidates D, a despised candidate $d \in C$, a list Π of prices associated with candidates $C \cup D$, and available budget $K \in \mathbb{N}$. If d is already not a unique winner of the election then accept. Otherwise for each candidate $p \in C \cup D$ distinct from d, we check if it is possible to ensure that $\mathrm{diff}(p, d) \geq 0$ by executing some control action within budget. Let $A = (C \cup D) \setminus \{d, p\} = \{a_1, \ldots, a_n\}$ and define $m(i, g)$ to be the minimal price of a control action necessary to achieve a total gain (as defined in the previous proof) of at least g by adding/deleting candidates from the set $\{a_1, \ldots, a_i\}$ only. It is easy to see that the following recursive relation holds:

$$m(i,g) = \begin{cases} 0 & \text{if } i = 0 \text{ and } g = 0 \\ \infty & \text{if } i = 0 \text{ and } g \neq 0 \\ \min\left(m(i-1,g), \Pi(a_i)\right) & \text{if } i > 0 \text{ and } g \leq \mathrm{gain}_{p,d}(a_i) \\ \min\left(m(i-1,g), m(i-1, g - \mathrm{gain}_{p,d}(a_i)) + \Pi(a_i)\right) & \text{if } i > 0 \text{ and } g > \mathrm{gain}_{p,d}(a_i) \end{cases}$$

Indeed, $m(0, g)$ is the price of achieving gain g without adding or deleting candidates and, so, $m(0, g) = 0$ if $g = 0$ and $m(0, g) = \infty$ otherwise. Now, the value $m(i, g)$ for $i > 0$ can be computed as follows: To achieve gain g by adding/deleting candidates from the set $\{a_1, \ldots, a_i\}$ we either add/delete a_i or we do not. If we do not, then $m(i, g) = m(i - 1, g)$. If we do, then again we consider two cases. If adding/deleting a_i alone guarantees gain g (i.e., if $g \leq \mathrm{gain}_{p,d}(a_i)$) then it suffices to only add/delete a_i and so we have $m(i, g) = \Pi(a_i)$. Otherwise, if $g > \mathrm{gain}_{p,d}(a_i)$, then we add/delete a_i and some candidates from the set $\{a_1, \ldots, a_{i-1}\}$. In effect, $m(i, g) = m(i - 1, g - \mathrm{gain}_{p,d}(a_i)) + \Pi(a_i)$. Since, while

computing $m(i, g)$, we do not know if we add/delete a_i or not, we compute both options and take the one with lower cost.

When the values $m(i, g)$ are computed, it is easy to solve our problem. If p is in C, then p can beat or tie the despised candidate d if and only if $m(n, \text{diff}_E(d, p)) \leq K$. If p is in D (i.e., if we need to add p to the election), then p can beat or tie d if and only if $m(n, \text{diff}_E(d, p)) + \Pi(p) \leq K$.

Let us now argue that the algorithm runs in polynomial time. Computing the scores of all the candidates and all the necessary gain values requires a scan over every vote, for every pair of candidates. For a given choice of p, computing the $m(i, g)$ values also requires a polynomial number of steps. This is so because the values of i are between 0 and $\|C \cup D\|$ and the maximum value of g that we may need to consider is $\|C \cup D\| \max(x, y)$. Using standard dynamic-programming techniques we can compute all the $m(i, g)$ values in polynomial time. Finally, there are polynomially many choices of p. □

Corollary 3.9 For each rational α, $0 \leq \alpha \leq 1$, destructive Copeland$^\alpha$-AC+DC-priced-control is in P.

Proof For each rational α, $0 \leq \alpha \leq 1$, this follows directly from Theorem 3.8. □

The above discussion shows that algorithms for priced control are not always simple extensions of those for the unpriced cases, and indeed can require new ideas. In the next section we show that it is possible that introducing prices moves control problems from being solvable in polynomial time to being NP-hard.

4 Prices can increase the complexity of control

In the previous section, we have shown that for several prominent voting rules introducing prices does not affect the complexities of our control problems. Now we will show that, nonetheless, there are rules for which it is not the case. First, we show that destructive AV-priced-control and DV-priced-control problems are polynomial-time solvable for scoring rules, provided that either the prices or the entries of the used scoring vectors are encoded in unary.[2] In particular, it means that for every scoring protocol, destructive control by adding/deleting voters (without prices) is in P. Second, we show an example of a scoring protocol, whose entries are encoded in binary, for which destructive AV-priced-control and DV-priced-control problems are NP-hard.

It is interesting to compare our results to those of Faliszewski, Hemaspaandra, and Hemaspaandra [23] regarding control problems in weighted elections. While they mostly consider constructive cases, they remark that destructive voter control for scoring protocols in weighted elections is in P (in weighted elections for each voter v there is a natural number w_v, his or her weight, and we treat the vote of v as if it were cast by w_v voters with the same preference order). Our results show that the complexity of destructive voter control for the case of priced elections behaves in much more intricate ways.

[2] Prior to our work, Faliszewski at al. [20] already observed that some NP-complete priced bribery problems can be solved in polynomial time if the prices are encoded in unary.

4.1 Vulnerability results

We first provide our vulnerability results. To simplify the proofs, we define the following head-to-head priced control problem in which we ask if some specific candidate can tie or beat the despised candidate by adding voters to the election.

Name: Scoring head-to-head priced control.
Given: An election (C, V), a scoring vector $\alpha = \langle \alpha_1, \ldots, \alpha_{\|C\|} \rangle$, a despised candidate $d \in C$, a preferred candidate $p \in C$ distinct from d, a set of additional voters W, a list of natural numbers Π describing prices associated with voters W, and available budget $K \in \mathbb{N}$.
Question: Is there a subset $W' \subseteq W$ such that $\mathrm{diff}_{(C, V \cup W')}(p, d) \geq 0$ and $\Pi(W') \leq K$.

In scoring head-to-head priced control, candidates' scores in the election are calculated using a given scoring vector α.

Lemma 4.1 *Scoring head-to-head priced control is in P if scoring vectors entries are represented in unary.*

Proof In scoring head-to-head priced control we are given an election $E = (C, V)$, a scoring vector α, a despised candidate d, a preferred candidate $p \in C \setminus \{d\}$, a set of additional voters W with their prices Π, and available budget $K \in \mathbb{N}$. We assume that $\mathrm{diff}_E(d, p) > 0$; otherwise the problem is trivial. Define $\mathrm{gain}_E(v)$ to be $\mathrm{diff}_{(C,\{v\})}(p, d)$, i.e., the score difference that p gains relative to d if we add voter v to the election E. We observe that it is of no use to add voters with nonpositive gain if we try to increase the score difference between p and d. Let $W' = \{w_1, \ldots, w_n\}$ be the set of voters from W with positive gain. Let $m(i, g)$ be the minimal price of a control action necessary to achieve a total gain (summed over all votes we decided to add) equal to or higher than g, using the first i voters from W'. If it is not possible to achieve such gain, define $m(i, g)$ to be infinite. The value $m(i, g)$ can be computed using the following recursive definition:

$$m(i,g) = \begin{cases} 0 & \text{if } i = 0 \text{ and } g = 0 \\ \infty & \text{if } i = 0 \text{ and } g > 0 \\ \min\left[m(i-1,g),\ \Pi(w_i)\right] & \text{if } i \geq 1 \text{ and } g \leq \mathrm{gain}_E(w_i) \\ \min\left[m(i-1,g),\ m(i-1, g-\mathrm{gain}_E(w_i)) + \Pi(w_i)\right] & \text{if } i \geq 1 \text{ and } g > \mathrm{gain}_E(w_i) \end{cases}$$

(The explanation and justification of this function is, in essence, the same as for the analogous function in the proof of Theorem 3.8, but considering the voters instead of the candidates.) Candidate p can tie or beat d if and only if $m(\|W'\|, \mathrm{diff}_E(d, p)) \leq K$. This completes the description of the algorithm.

The algorithm runs in polynomial time. Computing all the scores and all the gain values requires a single scan over the whole profile (and the votes that can be added). Computing all the necessary $m(i, g)$ values, using standard dynamic-programming techniques, also requires only polynomially many steps because there are $O(\|W\|)$ choices of i and $O(\alpha_1 \|W\|)$ choices of g, where α_1 is the number of points that our scoring protocol assigns to the top-ranked candidate (recall that we have assumed the scoring protocol to be encoded in unary). □

Lemma 4.2 *Scoring head-to-head priced control is in P if prices are represented in unary.*

Proof We give a proof similar in spirit to the proof of Lemma 4.1. In scoring head-to-head priced control we are given an election $E = (C, V)$, a distinguished candidate d, a preferred candidate $p \in C \setminus \{d\}$, a set of additional voters W and their prices Π. Assume that $\mathrm{diff}_E(d, p) > 0$; otherwise the problem is trivial. Define $\mathrm{gain}_E(v)$ to be the score difference that p gains relative to d if we add voter v to the election E. Let $W' = \{w_1, \ldots, w_n\}$ be the set of voters from W with positive gain. Let $g(i, \pi)$ be the maximal total gain, summed over all votes we decided to add, that can be achieved, using first i votes from W' with total price of control not exceeding π. The value $g(i, \pi)$ can be computed using the following recursive formulation:

$$g(i,\pi) = \begin{cases} 0 & \text{if } i = 0 \\ g(i-1,\pi) & \text{if } i \geq 1 \text{ and } \Pi(w_i) > \pi \\ \min\left[g(i-1,\pi),\ g(i-1,\pi-\Pi(w_i)) + \mathrm{gain}_E(w_i)\right] & \text{if } i \geq 1 \text{ and } \Pi(w_i) \leq \pi \end{cases}$$

(The explanation and justification of this function is, in essence, the same as for the analogous function in the proof of Theorem 3.8, but considering the voters instead of the candidates.) Candidate p can tie or beat d if and only if $g(\|W'\|, K) \geq \mathrm{diff}_E(d, p)$. This completes the description of the algorithm.

Using the same argument as in the preceding lemma, it is easy to see that the algorithm runs in polynomial time. In particular, computing all the necessary $g(i, \pi)$ values, using standard dynamic-programming techniques, requires polynomially many steps because there are $O(\|W\|)$ choices of i and $O(P\|W\|)$ choices of π, where P is the price of the most expensive voter in W (recall that prices are encoded in unary). □

Now we are ready to combine results from Lemma 4.1 and Lemma 4.2 and state the following result.

Theorem 4.3 *Destructive* AV*-priced-control is in P for scoring protocols if either the scoring vector entries or the prices are represented in unary, and if the scoring vectors for each number of candidates are computable in polynomial time with respect to the required number of candidates.*

Proof In the destructive AV-priced-control problem we are given an election $E = (C, V)$, a distinguished candidate $d \in C$, a set of additional voters W with their prices Π, and available budget $K \in \mathbb{N}$. If the despised candidate d is already not a unique winner of election E then accept. Otherwise check using the procedure from Lemma 4.1 (when the scoring vector entries are represented in unary) or Lemma 4.2 (when the prices are represented in unary) if there exists a candidate $p \in C \setminus \{d\}$ such that p can tie or beat candidate d after addition of some voters from W, within available budget K, and accept or reject accordingly. This requires at most $\|C\| - 1$ executions of algorithms from the mentioned theorems, therefore this procedure runs in polynomial time. □

The same approach can be used in the case of destructive control by deleting voters. No significant changes to the above proofs are required. It is simply a matter of updating the definition of gain to reflect that we are deleting voters instead of adding them, and running the dynamic programming over the voters already in the election instead of those that can be added.

Corollary 4.4 Destructive DV-priced-control is in P for scoring protocols if either scoring vector entries or prices are represented in unary, and if the scoring vectors for each number of candidates are computable in polynomial time in the required number of candidates.

For most natural classes of scoring protocols, such as, e.g., the Borda rule, the assumptions of the above theorems hold. We have the following corollary.

Corollary 4.5 Plurality, veto, k-veto, k-approval and Borda are vulnerable to destructive AV and DV priced control.

As a side comment, we mention that for some rules there is an interesting connection between the complexity of DV-priced-control and the complexity of AV-priced-control. Indeed, for some rules such as Borda, Condorcet, or Copeland, we can reduce the former to the latter.

Theorem 4.6 *Constructive (destructive)* DV*-priced-control is reducible in polynomial time to constructive (destructive)* AV*-priced-control for those voting rules for which it is possible to determine the winners in election E based on the following information only: (1) the set of all the candidates, and (2) for each two candidates c and d, the value $N_E(c, d) - N_E(d, c)$.*

Proof Let I_{DV} be an instance of constructive (destructive) DV-priced-control with election $E = (C, V)$, a distinguished candidate $c \in C$, a list of prices associated with voters Π, and available budget $K \in \mathbb{N}$. We reduce I_{DV} to instance I_{AV} of constructive (destructive) AV-priced-control, consisting of:

1. election $E = (C, V)$,
2. distinguished candidate c,
3. the set of additional voters W which consists of the voters from V with their preference orders reversed, and
4. available budget K.

Further, the price of adding voter $w \in W$ is the same as the price of deleting the voter $v \in V$ (in I_{DV}) from which w was created.

To see why the reduction works, it suffices to make the following simple observation. Let v be some voter in V and let w be the corresponding voter from W. Consider some arbitrary candidates a and b. Since w's preference order is the reverse of that of v, the following holds:

$$N_{(C,V\setminus\{v\})}(a, b) - N_{(C,V\setminus\{v\})}(b, a) = N_{(C,V\cup\{w\})}(a, b) - N_{(C,V\cup\{w\})}(b, a).$$

That is, as far as the values $N_E(a, b) - N_E(b, a)$ are concerned, the effect of deleting voter v is the same as the effect of adding voter w. By the assumption that winners depend only on the values $N_E(a, b) - N_E(b, a)$ (for all the candidates $a, b \in C$), we have that the reduction is correct. Polynomial running time is straightforward. □

To see that the above result applies to Borda, we note that the Borda score of a candidate c in election $E = (C, V)$ can be expressed as $\text{score}_E(c) = \sum_{d\in C\setminus\{c\}} N_E(c, d)$ (candidate c receives a point for each candidate d and each voter that ranks c above d). For each $d \in C \setminus \{c\}$, we have that $N_E(c, d) + N_E(d, c) = \|V\|$ and, so, $(N_E(c, d) - N_E(d, c)) + \|V\| = 2N_E(c, d)$. This means that the Borda score of candidate c is equal to $\text{score}_E(c) =$

$\frac{1}{2}\sum_{d\in C\setminus\{c\}}(N_E(c,d) - N_E(d,c) + \|V\|)$. So, Borda satisfies the conditions of the above theorem.

Theorem 4.6 is quite interesting since there are relatively few relations known between the complexities of various election-related problems. Some similar results were given by Faliszewski, Hemaspaandra, and Hemaspaandra [23] for the case of voter control under k-approval and k-veto, by Hemaspaandra, Hemaspaandra, and Menton [30] for the case of destructive control-by-partition problems, by Faliszewski, Hemaspaandra, and Hemaspaandra [20] for a relation between manipulation and priced bribery, and by Elkind, Faliszewski, and Slinko [16] for the case of the possible winner problem and the swap bribery problem.

4.2 Resistance results

We will now show a scoring protocol whose entries are computable in polynomial time and for which both destructive priced control by adding voters and destructive priced control by deleting voters are NP-hard. By our previous results, we know that the entries of our scoring protocol cannot be polynomially bounded in the number of candidates.

We design our scoring protocol to facilitate an NP-hardness proof based on a reduction from the X3C (eXact 3 Cover) problem. X3C is a well-known NP-complete problem [28]. We are given a set X and a family $\mathcal{S}$ of three-element subsets of X. We ask if there is an exact cover of X using sets from S. Formally, we define the X3C problem as follows:

Name: X3C
Given: Nonempty set $X = \{0, \ldots, 3k-1\}$, family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of three-element subsets of X.
Question: Is there $I \subseteq \{1, \ldots, n\}$ such that $\|I\| = k$ and $\bigcup_{i\in I} S_i = X$?

We now move on to defining our scoring protocol. For a given positive integer n and three integers $0 \leq x < y < z < n$, let $f_n(x, y, z)$ be the number of 3-element subsequences of $\langle n-1, \ldots, 0\rangle$ that are greater or equal to $\langle z, y, x\rangle$ in lexicographical order. For example, if we consider all 3-element subsequences of $\langle 5, \ldots, 0\rangle$, the greatest tuple in lexicographical order is $\langle 5, 4, 3\rangle$, therefore $f_6(3, 4, 5) = 1$. On the other hand $\langle 2, 1, 0\rangle$ is the least subsequence and we have that $f_6(0, 1, 2) = \binom{6}{3} = 20$. More generally, we have that $f_{3k}(3k-3, 3k-2, 3k-1) = 1$ and $f_{3k}(0, 1, 2) = \binom{3k}{3}$.

Definition 4.7 Define scoring protocol $\mathrm{SP_H}$ as follows. If the number of candidates in an election is equal to $m = \binom{3k}{3} + 1$ for some $k \in \mathbb{N}$, then we use scoring vector $\langle \alpha_1, \ldots, \alpha_m\rangle$ such that:

1. $\alpha_{f_{3k}(x,y,z)} = \binom{3k}{3}^x + \binom{3k}{3}^y + \binom{3k}{3}^z$ where $\langle z, y, x\rangle$ is a subsequence of $\langle 3k-1, \ldots, 0\rangle$.
2. $\alpha_m = 0$.

Otherwise, we use the Borda scoring vector.

Note that if $f_{3k}(a, b, c) < f_{3k}(x, y, z)$, where tuple $\langle z, y, x\rangle$ and tuple $\langle c, b, a\rangle$ are subsequences of $\langle 3k-1, \ldots, 0\rangle$, then $\langle c, b, a\rangle >_{\text{lex}} \langle z, y, x\rangle$. Applying the definition of lexicographical order, we have that $(c > z)$ or $(c = z \wedge b > y)$ or $(c = z \wedge b = y \wedge a > x)$ and in each of these cases it is easy to verify that $\alpha_{f_{3k}(a,b,c)} > \alpha_{f_{3k}(x,y,z)}$. Therefore, indeed, scoring vectors in the $\mathrm{SP_H}$ protocol are defined properly.

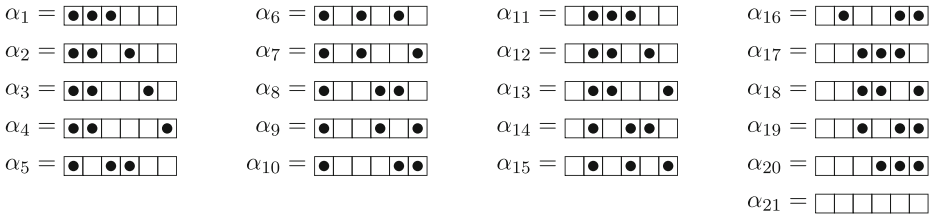

Fig. 6 SP_H scoring vector entries for election with 21 candidates. Values of scoring vector are represented in 20-ary numeral system, ones are depicted by cells with black dot inside, and zeros are depicted by empty cells.

Before presenting our main resistance results of this section, we give some intuition regarding SP_H scoring protocol in the following example.

Example 4.8 Let us consider the SP_H scoring vector for the case where the number of candidates is of the form $\binom{3k}{3}+1$ for some $k \in \mathbb{N}$. The entries of the scoring vector, for $k = 2$, are presented on the Fig. 6. The entries of our scoring vector, in base $\binom{3k}{3}$ encoding, are either all zeros or are all zeros with three ones. The first important property of SP_H that we will use in our proofs is a one-to-one correspondence between the entries $\alpha_1, \ldots, \alpha_{\binom{3k}{3}}$ and three element subsets of $\{0, \ldots, 3k-1\}$. The second one is the fact that to cause overflow when adding numbers base $\binom{3k}{3}$ whose digits are only zeros and ones, we need to add at least $\binom{3k}{3}$ numbers. By contrapositive, when we sum fewer than $\binom{3k}{3}$ such numbers, there is no overflow.

Now we can state and prove our main results of this section.

Theorem 4.9 SP_H *is resistant to destructive* AV*-priced-control.*

Proof SP_H is clearly susceptible to AV control. To show NP-hardness we give a reduction from X3C. Let $(X, \mathcal{S})$ be an X3C instance, where $X = \{0, \ldots, 3k-1\}$ and $\mathcal{S} = \{S_1, \ldots, S_n\}$. We assume that $k < n < \binom{3k}{3}$, as otherwise there is a trivial solution. Let $m = \binom{3k}{3}$. (We assume that $m \geq 3$; otherwise the instance can be solved trivially). We create a destructive AV-priced-control instance in the following way:

1. Candidate set C consists of $\{d, p\} \cup B$ where $B = \{b_j \mid 1 \leq j \leq m-1\}$ and d is the despised candidate.
2. The voter set V contains the following voters (when writing $a \succ B$ we mean that $(\forall b \in B)[a \succ b]$, and the order among candidates in B is arbitrary unless further specified, similarly for $B \succ a$):
   (a) $(n+k)m^5$ voters with preference order $d \succ p \succ B$.
   (b) $(n+k)m^5$ voters with preference order $p \succ d \succ B$.
   (c) k voters with preference order $B \cup \{d\} \succ p$, where candidate d is placed in position $f_{3k}(3i-3, 3i-2, 3i-1)$ for i, $1 \leq i \leq k$.
3. The set W of additional voters consists of one additional vote w_i for each set S_i in $\mathcal{S}$. In vote w_i, candidate d is least preferred and candidate p is placed in such a way that the addition of w_i to the election increases p's score by $\sum_{j \in S_i} m^j$ (e.g.,

if $S_i = \{7, 9, 13\}$ then candidate p is placed in position $f_{3k}(7, 9, 13)$). The remaining candidates are placed arbitrarily in the vote. The cost of adding w_i is equal to $\sum_{j \in S_i} m^j$.

4. The total available budget K is equal to $\sum_{i=0}^{3k-1} m^i$.

In election $E = (C, V)$, prior to adding any of the voters from W, the candidates have the following scores:

$$\begin{aligned}
\text{score}_E(d) &= \left[(n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + m^{3k-3} + m^{3k-4}\right] + \sum_{i=0}^{3k-1} m^i. \\
\text{score}_E(p) &= \left[(n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + m^{3k-3} + m^{3k-4}\right]. \\
\text{score}_E(b_i) &\leq \left[(n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + 2m^{3k-5}\right] \\
&\quad + k\left[m^{3k-1} + m^{3k-2} + m^{3k-3}\right].
\end{aligned}$$

Candidate d is the unique winner of this election with score advantage of $\sum_{i=0}^{3k-1} m^i$ (equal to the available budget K) over the candidate p, and score advantage of more than $n(m^{3k-1} + m^{3k-2} + m^{3k-3})$ over the candidates in B. To see why this is the case, note that for each $b_i \in B$ it holds that (recall that $m \geq 3$):

$$\begin{aligned}
\text{score}_E(d) - \text{score}_E(b_i) &\geq \left[(n+k)m^5\right]\left[m^{3k-3} + m^{3k-4} - 2m^{3k-5}\right] \\
&\quad + \sum_{i=0}^{3k-1} m^i - k\left[m^{3k-1} + m^{3k-2} + m^{3k-3}\right] \\
&> nm^{3k} + km^{3k} + \sum_{i=0}^{3k-1} m^i - k\left[m^{3k-1} + m^{3k-2} + m^{3k-3}\right] \\
&> nm^{3k} > n(m^{3k-1} + m^{3k-2} + m^{3k-3}).
\end{aligned}$$

If the input X3C instance has a solution, then adding votes from W that correspond to the sets S_i that constitute an exact cover of X increases the score of candidate p by K and requires budget of K. The score of the despised candidate d remains the same and destructive control is successful.

For the reverse direction, if control is possible then it must be a result of a tie between d and p. This is so because the scores of candidates from B can be increased by no more than $n(m^{3k-1} + m^{3k-2} + m^{3k-3})$, which is not sufficient to tie or beat d. Moreover, the score of candidate p must be increased by at least $\text{diff}_E(d, p) = K$, and by no more than the available budget K. Thus the sets S_i that correspond to the added voters must form an exact cover of X (recall the second property from Example 4.8). □

Destructive priced control by deleting voters also is NP-hard for our scoring rule.

Theorem 4.10 SP_H *is resistant to destructive* DV*-priced-control.*

Proof We give a reduction from X3C. Let $(X, \mathcal{S})$ be an X3C instance, where $X = \{0, \ldots, 3k-1\}$, and $\mathcal{S} = \{S_1, \ldots, S_n\}$. We assume that $k < n < \binom{3k}{3}$ (otherwise there is a trivial solution). Let $m = \binom{3k}{3}$ (we assume that $m \geq 3$; otherwise the input instance is trivial

to solve). Destructive priced control by deleting voters instance is created in the following way:

1. Candidate set C consists of $\{d, p\} \cup B$ where $B = \{b_i \mid 1 \leq i \leq m-1\}$, and d is the despised candidate.
2. The available budget K is $\sum_{i=0}^{3k-1} m^i$.
3. Voters set V contains the following voters:
   (a) $(3n+k)m^5$ voters with preference order $d \succ p \succ B$ and cost $K+1$.
   (b) $(3n+k)m^5$ voters with preference order $p \succ d \succ B$ and cost $K+1$.
   (c) k voters with preference orders of the form $B \cup \{d\} \succ p$, where candidate d is placed in position $f_{3k}(3i-3, 3i-2, 3i-1)$ for each i in $\{1, \ldots, k\}$, the cost of each voter is $K+1$.
   (d) n voters with preference orders of the form $B \cup \{p\} \succ d$. For each $S_i \in \mathcal{S}$, there is a vote in which p is placed in such a way as to receive score $\sum_{j \in S_i} m^j$ from this vote. The cost of each voter is $K+1$.
   (e) n voters, one for each S_i in $\mathcal{S}$, with preference orders of the form $B \cup \{d\} \succ p$, where in the vote corresponding to set S_i, candidate d is placed in such a way as to receive score of $\sum_{j \in S_i} m^j$. For each i, the cost of the vote corresponding to S_i is $\sum_{j \in S_i} m^j$.

Candidates receive the following scores in election $E = (C, V)$:

$$\begin{aligned}
1\text{score}_E(d) &= \left[(3n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + m^{3k-3} + m^{3k-4}\right] \\
&\quad + \sum_{i=1}^{n} \sum_{j \in S_i} m^j + \sum_{i=0}^{3k-1} m^i. \\
\text{score}_E(p) &= \left[(3n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + m^{3k-3} + m^{3k-4}\right] + \sum_{i=1}^{n} \sum_{j \in S_i} m^j. \\
\text{score}_E(b_i) &\leq \left[(3n+k)m^5\right]\left[2m^{3k-1} + 2m^{3k-2} + 2m^{3k-5}\right] \\
&\quad + (2n+k)\left[m^{3k-1} + m^{3k-2} + m^{3k-3}\right].
\end{aligned}$$

Candidate d is the unique winner of election E with score advantage of $\sum_{i=0}^{3k-1} m^i$ (equal to the available budget K) over p, and with score advantage of more than $n(m^{3k-1} + m^{3k-2} + m^{3k-3})$ over the candidates from B.

If input X3C instance has a solution, then deleting votes from V of type (33e) that correspond to sets S_i that constitute an exact cover of X decreases the score of d by K and requires a budget of K. The score of candidate p is unchanged and, thus, p and d tie and so destructive control is successful.

In the other direction, if control is possible it must be a result of a tie between d and p. This is so due to the existing score differences between the candidates in B and d, the fact that only voters of type (33e) can be deleted, and their prices. The score of candidate d must be therefore decreased by at least K to ensure that d ties or loses with p. But at the same time it cannot be decreased by more than K because the introduced score difference between d and p is equal to the price of the control action. Thus the total price must be exactly K and the deleted voters directly correspond to sets S_i constituting an exact cover of X. □

We believe that the above results are quite intriguing. While our scoring protocol SP_H is not likely to be used in any real-life election, it is not completely unnatural either. It is interesting if one can show that destructive control by adding/deleting voters is NP-hard for scoring vectors of the form $\langle 2^{m-1}, 2^{m-2}, \ldots, 2^1, 2^0 \rangle$. We leave this as an interesting open problem.

5 Summary

In this work we examined the computational complexity of election control for the case where different control actions (such as adding/deleting different candidates or voters) may come at different prices. We argued that such problems are useful ways of modeling problems that arise in planning political campaigns.

We examined the plurality, approval, Condorcet, and Copeland rules and we have shown that introducing prices does not affect the complexity of control problems for these rules. On the other hand, we have shown that there are scoring protocols for which unpriced destructive control is polynomial-time solvable, but for which introducing prices moves the problem to be NP-hard. This is quite interesting when we compare the complexity of priced control with the complexity of unpriced control for weighted elections. For the latter, Faliszewski, Hemaspaandra, and Hemaspaandra [23] show that destructive voter control is polynomial-time solvable for all scoring protocols, whereas we show a scoring protocol for which priced destructive voter control is NP-hard. This is interesting because in the bribery setting (which is relatively similar to the control setting) adding prices has a smaller impact on the complexity of the problem than adding weights. While one might have expected the same behavior for the case of control, we show that this is not the case. One possible reason for this difference between the complexity of the weighted and the priced variants of control is due to a subtle difference in the interpretation of their input. For example, the input for weighted voter control problems contains the total number of voters that can be added/deleted and this number can always be seen as encoded in unary. Instead, in priced control problems the input contains the budget that we can spend on adding/deleting voters. In our hardness proofs we rely on this budget being encoded in binary. (Yet, we should mention that the same input-interpretation issue applies to the bribery setting.)

Our work opens several interesting research directions. For example, one could seek if there are natural voting rules for which introducing prices increases the complexity of control problems. It is also interesting to consider the complexity of priced control in restricted domains, such as the single-peaked domain or the single-crossing domain. Another potential research direction is to consider approximation algorithms for the priced control problems.

Acknowledgments We would like to thank the reviewers for very helpful comments. The authors were supported by AGH University grant 11.11.230.124 (statutory research).

References

1. Bartholdi III, J., Tovey, C., Trick, M.: Voting schemes for which it can be difficult to tell who won the election. Soc. Choice Welf. **6**(2), 157–165 (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. Baumeister, D., Erdélyi, G., Erdélyi, O., Rothe, J.: Complexity of manipulation and bribery in judgment aggregation for uniform premise-based quota rules. Math. Soc. Sci. **76**, 19–30 (2015)
4. Baumeister, D., Faliszewski, P., Lang, J., Rothe, J.: Campaigns for lazy voters: Truncated ballots. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, pp. 577–584. International Foundation for Autonomous Agents and Multiagent Systems (2012)
5. Baumeister, D., Roos, M., Rothe, J., Schend, L., Xia, L.: The possible winner problem with uncertain weights. In: Proceedings of the 20th European Conference on Artificial Intelligence, pp. 133–138. IOS Press (2012)
6. Betzler, N., Uhlmann, J.: Parameterized complexity of candidate control in elections and related digraph problems. Theor. Comput. Sci. **410**(52), 43–53 (2009)
7. Brams, S., Fishburn, P.: Voting procedures. In: Arrow, K., Sen, A., Suzumura, K. (eds.) Handbook of Social Choice and Welfare, vol. 1, pp. 173–236. Elsevier (2002)
8. Brandt, F., Brill, M., Hemaspaandra, E., Hemaspaandra, L.: Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 715–722. AAAI Press (2010)
9. Bredereck, R., Chen, J., Faliszewski, P., Nichterlein, A., Niedermeier, R.: Prices matter for the parameterized complexity of shift bribery. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence, pp. 1398–1404. AAAI Press (2014)
10. Bredereck, R., Faliszewski, P., Niedermeier, R., Talmon, N.: Large-scale election campaigns: Combinatorial shift bribery. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, pp. 67–75 (2015)
11. Bulteau, L., Chen, J., Faliszewski, P., Niedermeier, R., Talmon, N.: Combinatorial voter control in elections. Theor. Comput. Sci. **589**, 99–120 (2015)
12. Chen, J., Faliszewski, P., Niedermeier, R., Talmon, N.: Elections with few voters: Candidate control can be easy. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, pp. 2045–2051 (2015)
13. Chevaleyre, Y., Lang, J., Maudet, N., Monnot, J., Xia, L.: New candidates welcome! possible winners with respect to the addition of new candidates. Math. Soc. Sci. **64**(1), 74–88 (2012)
14. Dorn, B., Schlotter, I.: Multivariate complexity analysis of swap bribery. Algorithmica **64**(1), 126–151 (2012)
15. Elkind, E., Faliszewski, P.: Approximation algorithms for campaign management. In: Proceedings of the 6th International Workshop On Internet And Network Economics, pp. 473–482. Springer-Verlag Lecture Notes in Computer Science #6484 (2010)
16. Elkind, E., Faliszewski, P., Slinko, A.: Swap bribery. In: Proceedings of the 2nd International Symposium on Algorithmic Game Theory, pp. 299–310. Springer-Verlag Lecture Notes in Computer Science #5814 (2009)
17. Elkind, E., Faliszewski, P., Slinko, A.: Cloning in elections: finding the possible winners. J. Artif. Intell. Res. **42**, 529–573 (2011)
18. Erdélyi, G., Fellows, M., Rothe, J., Schend, L.: Control complexity in Bucklin and fallback voting: A theoretical analysis. J. Comput. Syst. Sci. **81**(4), 632–660 (2015)
19. Faliszewski, P.: Nonuniform bribery (short paper). In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 1569–1572. International Foundation for Autonomous Agents and Multiagent Systems (2008)
20. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: How hard is bribery in elections J. Artif. Intell. Res. **35**, 485–532 (2009)
21. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Using complexity to protect elections. Commun. ACM **53**(11), 74–82 (2010)
22. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Multimode control attacks on elections. J. Artif. Intell. Res. **40**, 305–351 (2011)
23. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Weighted electoral control. J. Artif. Intell. Res. **52**, 507–542 (2015)
24. 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)
25. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: The shield that never was: Societies with single-peaked preferences are more open to manipulation and control. Inf. Comput. **209**(2), 89–107 (2011)
26. Faliszewski, P., Procaccia, A.: AI's war on manipulation: Are we winning AI Mag. **31**(4), 52–64 (2010)
27. Garcia-Molina, H.: Elections in a distributed computing system. IEEE Trans. Comput. **C-31**(1), 48–59 (1982)

28. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
29. Gibbard, A.: Manipulation of voting schemes. Econometrica **41**(4), 587–601 (1973)
30. 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, pp. 377–388. Leibniz-Zentrum für Informatik (2013)
31. Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. J. ACM **44**(6), 806–825 (1997)
32. Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Anyone but him: The complexity of precluding an alternative. Artif. Intell. **171**(5–6), 255–285 (2007)
33. Hemaspaandra, E., Spakowski, H., Vogel, J.: The complexity of Kemeny elections. Theor. Comput. Sci. **349**(3), 382–391 (2005)
34. Kuta, M., Kitowski, J.: Benchmarking high performance architectures with natural language processing algorithms. Comput. Sci. **12**, 19–31 (2011)
35. Lamport, L.: Paxos made simple. SIGACT News **32**(4), 51–58 (2001)
36. Liu, H., Feng, H., Zhu, D., Luan, J.: Parameterized computational complexity of control problems in voting systems. Theor. Comput. Sci. **410**(27–29), 2746–2753 (2009)
37. Liu, H., Zhu, D.: Parameterized complexity of control problems in maximin election. Inf. Process. Lett. **110**(10), 383–388 (2010)
38. Magiera, K., Faliszewski, P.: How hard is control in single-crossing elections? In: Proceedings of the 21st European Conference on Artificial Intelligence, pp. 579–584. IOS Press (2014)
39. Magrino, T., Rivest, R., Shen, E., Wagner, D.: Computing the margin of victory in IRV elections. Presented at 2011 Electronic Voting Technology Workshop/Workshop on Trushworthy Elections (2011)
40. Mattei, N., Goldsmith, J., Klapper, A.: On the complexity of bribery and manipulation in tournaments with uncertain information. In: Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference, pp. 549–554 (2012)
41. Mattei, N., Pini, M., Rossi, F., Venable, K.: Bribery in voting over combinatorial domains is easy. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, pp. 1407–1408. International Foundation for Autonomous Agents and Multiagent Systems (2012)
42. Meir, R., Procaccia, A., Rosenschein, J., Zohar, A.: The complexity of strategic behavior in multi-winner elections. J. Artif. Intell. Res. **33**, 149–178 (2008)
43. Papadimitriou, C.: Computational Complexity. Addison-Wesley (1994)
44. 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, pp. 1429–1435. AAAI Press (2012)
45. Rey, A., Rothe, J.: Bribery in path-disruption games. In: Proceedings of the 2nd International Conference on Algorithmic Decision Theory, pp. 247–261. Springer-Verlag Lecture Notes in Computer Science #6992 (2011)
46. Rothe, J., Spakowski, H., Vogel, J.: Exact complexity of the winner problem for Young elections. Theory Comput. Syst. **36**(4), 375–386 (2003)
47. Satterthwaite, M.: Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. J. Econ. Theory **10**(2), 187–217 (1975)
48. Wojtas, K., Faliszewski, P.: Possible winners in noisy elections. In: Proceedings of the 26th AAAI Conference on Artificial Intelligence, pp. 1499–1505 (2012)
49. Xia, L.: Computing the margin of victory for various voting rules. In: Proceedings of the 13th ACM Conference on Electronic Commerce, pp. 982–999. ACM Press (2012)
50. Xia, L.: Fixed-parameter tractability of integer generalized scoring rules. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, pp. 1599–1600 (2014)