CHAPTER 7

Control and Bribery in Voting

Piotr Faliszewski and Jörg Rothe

7.1 Introduction

In this chapter we study control and bribery, two families of problems modeling various ways of manipulating elections. Briefly put, control problems model situations where some entity, usually referred to as the chair or the election organizer, has some ability to affect the election structure. For example, the chair might be able to encourage new candidates to join the election, or might be able to prevent some voters from casting their votes. On the other hand, bribery models situations where the structure of the election stays intact (we have the same candidates and the same voters), but some outside agent pays the voters to change their votes. Naturally, such manipulative actions, dishonestly skewing election results, are undesirable. Thus it is interesting to know if there are so-called *complexity shields* against these attacks (see also Chapter 6 on manipulation and, relatedly, Section 4.3.3 in the book chapter by Baumeister and Rothe (2015)). That is, it is interesting to know the computational complexity of recognizing whether various forms of such attacks are possible or not. However, there are also other interpretations of control and bribery, many of them quite positive.

In this chapter we survey results on the complexity of control and bribery in elections, providing an overview of the specific problems studied, sketching sample proofs, and reviewing some approaches to dealing with the computational hardness of these control and bribery problems (see also Sections 4.3.4 and 4.3.5 in the book chapter by Baumeister and Rothe (2015)). Seeking ways of dealing with the computational hardness of control and bribery may seem surprising at first. However, on one hand, if we interpret control and bribery as modeling attacks on elections, then we would like to know the limitations of our complexity shields. On the other hand, if we take other interpretations of control and bribery, then we simply would like to know how to solve these problems. We survey some classical results on control in Section 7.3, on bribery in Section 7.4, and then briefly discuss their various applications in Section 7.5.

Table 7.1. *Three types of preference profiles required by different voting rules*

(a) A Borda election

points:	5	4	3	2	1	0
voter 1:	a	c	b	f	e	d
voter 2:	b	a	f	c	e	d
voter 3:	c	d	b	a	f	e
voter 4:	e	d	b	f	c	a
voter 5:	e	d	c	b	f	a
winner:	b with score 16					

(b) An approval election

	a	b	c	d	e	f
voter 1:	(1,	0,	0,	0,	0,	0)
voter 2:	(1,	1,	0,	0,	0,	0)
voter 3:	(1,	1,	1,	1,	0,	0)
voter 4:	(0,	0,	0,	1,	1,	0)
voter 5:	(0,	0,	0,	0,	1,	0)
AV score:	**3**	2	1	2	2	0

(c) A fallback election

level:	1	2	3	4
voter 1:	a			
voter 2:	b	a		
voter 3:	c	d	b	a
voter 4:	e	d		
voter 5:	e			
winner:	a on level 4			

7.2 Preliminaries

We start by recalling various voting rules, including preference-based voting rules and (variants of) approval voting. For the former, an *election* (A, R) is given by a set A of m *alternatives* (or *candidates*) and a *preference profile* $R = (\succ_1, \ldots, \succ_n)$ over A that collects n votes, each expressing a linear preference order over A. That is, letting $N = \{1, \ldots, n\}$ be the set of voters, $\succ_i$ gives voter i's preference order of the alternatives. For example, the ranking $a \succ_1 b \succ_1 c$ says that voter 1 (strictly) prefers alternative a to alternative b, and b to c. From now on we omit stating "$\succ_i$" explicitly and simply rank the alternatives in a vote from left (most preferred) to right (least preferred). That is, instead of, say, $a \succ_1 b \succ_1 c$ we simply write $a\ b\ c$. Also, for (A, R) an election and $A' \subseteq A$, we write (A', R) to denote the election with alternatives A' and the votes in R restricted to A'. For example, if (A, R) is the election from Table 7.1(a) consisting of five voters who rank six alternatives and $A' = \{b, c, d\}$, then $(A', R) = (\{b, c, d\}, (c\ b\ d,\ b\ c\ d,\ c\ d\ b,\ d\ b\ c,\ d\ c\ b))$.

We briefly recall some voting rules, see Chapter 2 for more details. *Positional scoring rules* are defined by an m-alternative *scoring vector* $\vec{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)$, where the σ_i are nonnegative integers with $\sigma_1 \geqslant \sigma_2 \geqslant \cdots \geqslant \sigma_m$. Each alternative scores σ_i points for each vote where it is ranked in the ith position, and whoever scores the most points wins. Examples are *plurality voting* with scoring vector $(1, 0, \ldots, 0)$, *veto* (aka *antiplurality*) with $(1, \ldots, 1, 0)$, *k-approval* with $(1, \ldots, 1, 0, \ldots, 0)$ having a 1 in each of the first $k \leqslant m$ positions (note that 1-approval is plurality), *k-veto*, which is the same as $(m - k)$-approval (note that 1-veto is veto), and *Borda count* with $(m - 1, m - 2, \ldots, 0)$. For example, in the election given in Table 7.1(a), e wins under plurality; d under 2-approval; b under 3-approval; b, c, and f under veto; and b under Borda (with a Borda score of 16, whereas a, c, d, e, and f score, respectively, 11, 15, 12, 12, and 9 points).

Under *approval voting* (or *AV*), proposed by Brams and Fishburn (1978, 2007), instead of using preference orders the voters specify sets of alternatives they approve of. Typically, such votes are represented as m-dimensional 0/1-vectors, where each position corresponds to an alternative and 1-entries mean approval of respective alternatives. All alternatives with the most approvals win. For example, for the approval vectors given in Table 7.1(b), a is the approval winner with a score of 3. A version of approval voting (dubbed *sincere-strategy preference-based approval voting* (or *SP-AV*) by Erdélyi et al. (2009)) combines approval information with preference-order

information (the voters rank the candidates that they approve of). The rule was introduced by Brams and Sanver (2006) and, in essence, is the same as approval, but the additional preference-order information is used to deduce voter behavior when the candidate set changes (we omit detailed discussion and point the reader to the original papers and to the survey of Baumeister et al. (2010)).

Range voting (or *RV*) works just as approval voting, except that entries of the vectors under k-range voting come from the set $\{0, 1, \ldots, k\}$ rather than from the set $\{0, 1\}$. *Normalized range voting* (or *NRV*) is a variant of RV that alters the votes so that the potential impact of each vote is maximized (see, e.g., the work of Menton, 2013).

Let us now move back to rules based on preference orders and, in particular, to those rules that are based on pairwise comparisons of alternatives. A *Condorcet winner* is an alternative that is preferred to every other alternative by a strict majority of votes. For example, in the election from Table 7.1(a), c is preferred to every other alternative by three of the five voters and thus is the Condorcet winner. It is easy to see that there is at most one Condorcet winner in an election, but it is possible that there is none.[1] A voting rule is *Condorcet-consistent* if it elects the Condorcet winner whenever there is one. If there is no Condorcet winner in a given preference profile, many of the known Condorcet-consistent rules elect those candidates that are closest to being Condorcet winners, one way or another. For example, under *Copeland$^\alpha$ voting*, $\alpha \in [0, 1]$, we organize a tournament among the candidates in the following way: Each pair of candidates "plays" against each other and the one that is preferred by more voters wins and receives a point (in case of a tie, both get α points). In the end, the candidates with the highest number of points win. If we omit voter 1 from the election in Table 7.1(a) then d is the unique Copeland$^\alpha$ winner for $\alpha = 0$ and $\alpha = 1/2$ (with a Copeland$^\alpha$ score of 3 if $\alpha = 0$, and of 3.5 if $\alpha = 1/2$), but both c and e are Copeland$^\alpha$ winners with a score of 5 for $\alpha = 1$. Other Condorcet-consistent rules are, for example, the *maximin rule* (aka *Simpson's rule*), *ranked pairs* due to Tideman (1987), or *Schulze's rule* (a rule proposed by Schulze (2011), which satisfies many normative properties).

Other voting rules follow yet other principles, e.g., *single transferable vote* (*STV*) proceeds in stages and eliminates the "weakest" candidates until only the winner remains. We omit the details and point the reader to Chapter 2 instead. Under *Bucklin voting* we first seek the smallest value ℓ such that there is candidate ranked among top ℓ positions by a strict majority of the voters, and then declare as winners those candidates that have highest ℓ-approval scores (or, under *simplified Bucklin voting*, those candidates that are ranked among top ℓ positions by some majority of the voters). *Fallback voting*, introduced by Brams and Sanver (2009), is a rule that combines Bucklin voting with approval voting (the voters rank only the candidates they approve of and Bucklin is used; if there are no Bucklin winners—due to the fact that voters do not have to rank all the candidates—fallback outputs the approval winners). For example, in the partial rankings given in Table 7.1(c), a alone reaches a strict majority

[1] If one requires voting rules to always have at least one winner, Condorcet voting (which elects the Condorcet winner whenever there is one, and otherwise no one wins) would not be a voting rule. However, we take the point of view that voting rules may have empty winner sets. Note that it has become a tradition to study (at least) plurality, Condorcet, and approval on each new approach and each new idea regarding election control (see, e.g., the papers of Bartholdi et al., 1992; Hemaspaandra et al., 2007; Faliszewski et al., 2011c).

of $\lfloor 5/2 \rfloor + 1 = 3$ votes (namely, on the fourth level) and thus is the fallback winner. However, if the first voter approved only of f instead of only of a, then no candidate would reach a majority and fallback would output a, b, d, and e, the approval winners of the election.

In Sections 7.3 and 7.4, we will define a large variety of decision problems, each related to some specific control or bribery scenario. All these problems are members of NP, the class of problems that can be solved in nondeterministic polynomial time, and they will be classified to be either in P or NP-complete.[2] Unlike, for instance, in the case of Kemeny, Dodgson, and Young elections (which we do not consider here, as their winner problems are not in P—see Chapters 4 and 5), the winner(s) can be determined efficiently for all voting systems described earlier.

7.3 Control

Every election needs to be organized, and whoever is responsible for doing so can have some influence on the outcome of the election by changing its structure. We will refer to this person, or authority, as the *chair* of the election, and to the way the election structure is changed by the chair as *control type* or *control action*. Many types of control that the chair might exert are conceivable. We present those that have been studied in the literature, starting with the four most important ones.

7.3.1 Constructive Control by Adding/Deleting Candidates/Voters

Bartholdi et al. (1992) were the first to introduce electoral control and to study it in various scenarios from a computational perspective. In particular, they defined *constructive* control types, where the chair's goal in exerting some control action is to make a given candidate p the unique winner of the resulting election.[3] It is common to assume that the chair has complete knowledge of all votes.

One control action the chair might exert is to change the candidate set, either by adding some new candidates from a given set of spoiler candidates (hoping to make p's most competitive rivals weaker relative to p), or to delete up to k candidates from the given election (to get rid of p's worst rivals). For the former, Bartholdi et al. (1992) originally defined a variant that allows adding an unlimited number of spoiler candidates. To be in sync with the other control problems (e.g., control by deleting candidates), Hemaspaandra et al. (2009) defined a variant of this problem where a bound k on the number of spoiler candidates that may be added is given. We will see later that the complexity of the resulting problems can sharply differ.

[2] A problem B is NP-*hard* if every NP problem A reduces to B, where "reduction" always refers to a *polynomial-time many-one reduction*, that is, a polynomial-time function r mapping instances of A to instances of B such that for each x, $x \in A \iff r(x) \in B$. B is NP-*complete* if it is NP-hard and in NP.

[3] As we do here, control problems have commonly, most especially in the earlier papers on control, been studied in their *unique-winner* variant. Alternatively, many papers on control consider the *nonunique-winner* (or *co-winner*, or simply *winner*) variant where the chair's goal is merely to make the designated candidate a winner. The complexity of control problems is usually the same in both models, requiring only minor adjustments to the proofs.

Definition 7.1. Let f be a voting rule. In the CONSTRUCTIVE-CONTROL-BY-ADDING-AN-UNLIMITED-NUMBER-OF-CANDIDATES problem for f (f-CCAUC), we are given (a) a set A of qualified candidates, a set B of spoiler candidates, where $A \cap B = \emptyset$, and an election $(A \cup B, R)$ and (b) a preferred candidate $p \in A$. We ask if we can choose a subset $B' \subseteq B$ of the spoiler candidates such that p is the unique f-winner of the election $(A \cup B', R)$. The CONSTRUCTIVE-CONTROL-BY-ADDING-CANDIDATES problem for f (f-CCAC) is defined similarly: In addition to (a) and (b) we are also given (c) a bound $k \in \mathbb{N}$, and we ask if there is a subset $B' \subseteq B$ of spoiler candidates such that $|B'| \leqslant k$ and p is the unique f-winner of $(A \cup B', R)$. In the CONSTRUCTIVE-CONTROL-BY-DELETING-CANDIDATES problem for f (f-CCDC), we are given (a) an election (A, R), (b) a preferred candidate $p \in A$, and (c) a bound $k \in \mathbb{N}$. We ask if p can be made a unique f-winner of the election resulting from (A, R) by deleting at most k candidates.

The issue of control by changing the candidate set is very natural and, indeed, happens in real-life political elections. For example, it is widely speculated that "adding" Nader to the 2000 U.S. presidential election had the effect of ensuring Bush's victory (otherwise, Gore would have won). Similarly, there are known cases where "spoiler" candidates were added to political elections to confuse the voters (see, e.g., the *New York Times* article of Lacey (2010) for a reported example). It is also easy to imagine control by deleting candidates: Some of the candidates who perform poorly in pre-election polls may be forced (or persuaded) to withdraw.

Example 7.1. For a Borda-CCAUC instance, let $(A \cup B, R)$ be the election from Table 7.1(a), where $A = \{a, b, c, d\}$ is the set of qualified candidates and $B = \{e, f\}$ is the set of spoiler candidates. Table 7.2(a) shows the restriction (A, R) of this election to the qualified candidates, which has the Borda winner c scoring 9 points, while the Borda scores of a, b, and d, respectively, are 4, 6, and 8. Supposing that b is the chair's favorite candidate, we have a yes-instance of Borda-CCAUC, since adding both spoiler candidates makes b the unique Borda winner (see Table 7.1(a)).

To turn this into a Borda-CCAC instance, we in addition need to specify an addition limit, k. If $k = 1$, we have a yes-instance of the problem: Though the chair will not succeed by adding e (which gives the election in Table 7.2(c), still won by c), adding f (giving the election in Table 7.2(b)) will make b the unique Borda winner with a score of 13, while a, c, d, and e score 8, 12, 11, and 6 points.

Finally, consider again the Borda election in Table 7.1(a) with winner b, and suppose the chair, who now wants to make c win, is allowed to delete one candidate. Deleting the current champion, b, will reach this goal. Alternatively, the chair can delete f (see Table 7.2(c)) to turn c into the unique winner with a Borda score of 12, while the Borda scores of a, b, d, and e, respectively, are reduced to 8, 11, 9, and 10. Thus this is a yes-instance of the problem Borda-CCDC.

The chair might also change the voter set, either by encouraging further voters to participate (knowing that their votes will be beneficial for p), or by excluding certain voters from the election (knowing that deleting their votes will help p). In real life, political parties often try to influence the outcome of elections by such actions (e.g.,

Table 7.2. CCAUC, CCAC, *and* CCDC *for the Borda election in Table 7.1(a)*

(a) Without spoilers e and f				
points:	3	2	1	0
voter 1:	a	c	b	d
voter 2:	b	a	c	d
voter 3:	c	d	b	a
voter 4:	d	b	c	a
voter 5:	d	c	b	a
winner:	c (score 9)			

(b) Deleting e					
points:	4	3	2	1	0
voter 1:	a	c	b	f	d
voter 2:	b	a	f	c	d
voter 3:	c	d	b	a	f
voter 4:	d	b	f	c	a
voter 5:	d	c	b	f	a
winner:	b (score 13)				

(c) Deleting f					
points:	4	3	2	1	0
voter 1:	a	c	b	e	d
voter 2:	b	a	c	e	d
voter 3:	c	d	b	a	e
voter 4:	e	d	b	c	a
voter 5:	e	d	c	b	a
winner:	c (score 12)				

think of targeted "get-out-the-vote" drives on one hand, and of voter suppression efforts or even disenfranchisement of voters, on the other).

Definition 7.2. Let f be a voting rule. In the CONSTRUCTIVE-CONTROL-BY-ADDING-VOTERS problem for f (f-CCAV), we are given (a) a list R of already registered votes, a list S of as yet unregistered votes, and an election $(A, R + S)$, where "profile addition" means concatenation of profiles, (b) a preferred candidate $p \in A$, and (c) a bound $k \in \mathbb{N}$. We ask if we can choose a sublist $S' \subseteq S$ of size at most k such that p is the unique f-winner of $(A, R + S')$. In the CONSTRUCTIVE-CONTROL-BY-DELETING-VOTERS problem for f (f-CCDV), we are given (a) an election (A, R), (b) a preferred candidate $p \in A$, and (c) a bound $k \in \mathbb{N}$, and we ask if we can make p a unique f-winner of the election resulting from (A, R) by deleting no more than k votes.

Example 7.2. Look again at the Borda election in Table 7.1(a) and assume that the chair wants to make c win. If one voter may be deleted, the chair's goal can be reached by deleting voter 2: c then is the unique Borda winner with a score of 13, while a, b, d, e, and f score only 7, 11, 12, 11, and 6 points, so this is a yes-instance of the problem Borda-CCDV. On the other hand, if a were the chair's favorite choice, the chair would not succeed even if two votes may be deleted, giving rise to a no-instance of Borda-CCDV. As an example of a Borda-CCAV instance, suppose voters 1 and 2 from the election in Table 7.1(a) are registered already, but 3, 4, and 5 are not. The current winner is a. Suppose the chair wants to make c win and is allowed to add two voters. Adding any single one of the as yet unregistered voters is not enough (the best c can reach, by adding voter 3, is to tie with a and b for first place, each having 11 points). Adding either $\{3, 4\}$ or $\{4, 5\}$ is not successful either. However, adding $\{3, 5\}$ makes c the unique Borda winner with a score of 14, while a, b, d, e, and f score only 11, 13, 8, 7, and 7 points.

Depending on the voting rule, it may never (for no preference profile at all) be possible for the chair to successfully exert some control action (e.g., constructive control by deleting voters) in the sense that p can be turned (by deleting voters) from not being a unique winner into being one. If that is the case, we say this voting rule is *immune* to this type of control. Otherwise (i.e., if there is at least one preference profile where the chair can successfully exert this control action), we say this voting rule is *susceptible* to this type of control. For a voting rule f that is susceptible to

some type of control (e.g., to constructive control by adding voters), f is said to be *vulnerable* (respectively, *resistant*) to this control type if the corresponding problem (e.g., f-CCAV) is in P (respectively, NP-hard).

Immunity results appear to be very desirable. Indeed, if a voting rule is immune to a given type of control, then it is impossible to compromise its result by a corresponding type of malicious action. Nonetheless, as we will soon see, immunity can also bring some undesired side effects. First, however, let us argue that immunity for candidate control is rare. This is so due to the study of *strategic candidacy* of Dutta et al. (2001) (see also recent work on strategic candidacy of Lang et al. (2013) and Brill and Conitzer (2015)). They have considered a setting where the candidates have preferences regarding election outcomes, and can strategically choose to join the race or not. Dutta et al. (2001) have shown that for most typical election rules there are settings where some candidates would prefer not to participate in the election. In effect, such rules cannot be immune to candidate control. Nonetheless, in some rare cases (e.g., for Condorcet and approval voting) immunity results for candidate control hold (see Table 7.3).

For the case of voter control, immunity is not only rare, but also is utterly undesirable. Indeed, it is natural to expect that if we add sufficiently many voters with the same preference order, then their most preferred candidate becomes a winner. Formally, this is known as *voting rule continuity* (or, as the *Archimedean property*). Continuity says that if some candidate c is a winner in some election (A, R), then for every election (A, R'), there is a natural number t such that c is a winner in an election of the form $(A, R' + tR)$, where tR refers to a profile of t copies of profile R. See, for example, the work of Smith (1973).

The first voting rules studied with respect to control were plurality, Condorcet, and approval voting. The following theorem summarizes some of the results obtained for them by Bartholdi et al. (1992) and Hemaspaandra et al. (2007).

Theorem 7.3 (Bartholdi et al., 1992; Hemaspaandra et al., 2007).

1. *Condorcet and approval voting are immune and plurality is resistant to constructive control by adding (respectively, adding an unlimited number of) candidates.*
2. *Condorcet and approval voting are vulnerable and plurality is resistant to constructive control by deleting candidates.*
3. *Condorcet and approval voting are resistant and plurality is vulnerable to constructive control by both adding and deleting voters.*

These immunity claims generally follow from the fact that Condorcet and approval voting satisfy the ("unique" version of the) Weak Axiom of Revealed Preference, which states that a unique winner p in a set A of alternatives always is also a unique winner among each subset $A' \subseteq A$ including p. Hemaspaandra et al. (2007) identify many links (i.e., implications and equivalences) among the susceptibility/immunity statements for the control types defined previously and to be defined in Section 7.3.2. We refrain from repeating them here but point the reader to Figure 4.16 in the book by Rothe et al. (2011) for an overview.

The vulnerability claims in Theorem 7.3 follow by simple P algorithms. For example, that approval-CCDC is in P follows from this algorithm: On input $((A, R), p, k)$, if p

already is the unique winner in (A, R) (which is easy to test), output "yes"; otherwise, if no more than k candidates have at least as many approvals as p, output "yes" (as p can be made a unique winner by deleting them all), and else output "no." By contrast, vulnerability proofs for the partitioning cases to be defined in Section 7.3.2 are often much more involved (and are omitted here).

The resistance claims in Theorem 7.3 typically follow by reductions from NP-complete problems such as EXACT-COVER-BY-3-SETS (X3C), which given a base set $B = \{b_1, \ldots, b_{3k}\}$, $k > 0$, and a sequence $\mathcal{S} = (S_1, \ldots, S_n)$ of 3-element subsets of B, asks whether B can be exactly covered by k sets chosen from $\mathcal{S}$. For example, to show that approval-CCDV is NP-hard, let $(B, \mathcal{S})$ be an instance of X3C. Let $\ell_j = |\{S_i \in \mathcal{S} \mid b_j \in S_i\}|$ for each j, $1 \leqslant j \leqslant 3k$. Construct from $(B, \mathcal{S})$ the election (A, R) with $A = B \cup \{p\}$ and R consisting of the following $2n$ voters: (1) For each i, $1 \leqslant i \leqslant n$, one voter in R approves of all candidates in S_i and disapproves of all other candidates; (2) there are n voters $v_1, \ldots, v_n$ in R such that, for each i, $1 \leqslant i \leqslant n$, v_i (a) approves of p, and (b) approves of b_j if and only if $i \leqslant n - \ell_j$. Thus, every candidate in (A, R) has exactly n approvals. If there is an exact cover for B, then deleting the k votes from R corresponding to the exact cover turns p into the unique winner. Conversely, suppose that p can be turned into a unique approval winner by deleting at most k votes from R (where we may assume that none of them approves of p, so only votes from group (1) have been deleted). For p to become the unique approval winner, every $b_j \in B$ must have lost at least one approval. Thus, the deleted votes correspond to an exact cover for B.

The next system to be comprehensively studied regarding control was Copeland$^\alpha$.

Theorem 7.4 (Faliszewski et al., 2009c). *For each rational number α, $0 \leqslant \alpha \leqslant 1$, Copeland$^\alpha$ is resistant to all types of control from Definitions 7.1 and 7.2, except for $\alpha \in \{0, 1\}$ where Copeland$^\alpha$ is vulnerable to constructive control by adding an unlimited number of candidates.*

The most interesting point to note in Theorem 7.4 is that Copeland$^\alpha$-CCAUC is in P for $\alpha = 0$ and $\alpha = 1$, but is NP-complete for all other values of α. The vulnerability results are proven by the following simple P algorithm: On input $((A \cup B, R), p)$, set D_1 to be the set of all $b \in B$ such that the Copeland$^\alpha$ score of p in $(\{b, p\}, R)$ is 1; initialize D to be D_1; and then successively delete every b from D such that the Copeland$^\alpha$ score of p in $(A \cup D, R)$ is no greater than that of b. Correctness of the algorithm follows from (1) the observation that for each $D \subseteq B$, whenever p is the unique Copeland$^\alpha$ winner in $(A \cup D, R)$, then so is p in $(A \cup (D_1 \cap D), R)$, and (2) a more involved argument showing that if p is the unique winner in $(A \cup D, R)$ for some $D \subseteq D_1$, yet the above algorithm computes a set D' such that p is *not* a unique winner in $(A \cup D', R)$, then this leads to a contradiction. On the other hand, NP-hardness of Copeland$^\alpha$-CCAUC for $0 < \alpha < 1$ follows by a reduction from the NP-complete problem VERTEX-COVER (and is omitted here).

Unlike Copeland$^\alpha$-CCAUC, Copeland$^\alpha$-CCAC is NP-complete for *all* (rational) values of $\alpha \in [0, 1]$. In fact, Copeland$^\alpha$ with $0 < \alpha < 1$ (including the original system by Copeland (1951)) is the first family of voting rules known to be fully resistant to all types of constructive control, including those to be defined in Section 7.3.2. Other voting rules having this property have followed: SP-AV (Erdélyi et al., 2009),

Table 7.3. *The complexity of control problems for various voting rules*

Voting Rule	CAUC C	CAUC D	CAC C	CAC D	CDC C	CDC D	CPC-TE C	CPC-TE D	CPC-TP C	CPC-TP D	CRPC-TE C	CRPC-TE D	CRPC-TP C	CRPC-TP D	CAV C	CAV D	CDV C	CDV D	CPV-TE C	CPV-TE D	CPV-TP C	CPV-TP D
plurality	R	R	R	R	R	R	R	R	R	R	R	R	R	R	V	V	V	V	V	V	R	R
(Bartholdi et al., 1992; Hemaspaandra et al., 2007)																						
Condorcet	I	V	I	V	V	I	V	I	V	I	V	I	V	I	R	V	R	V	R	V	R	V
(Bartholdi et al., 1992; Hemaspaandra et al., 2007)																						
approval	I	V	I	V	V	I	V	I	I	I	V	I	I	I	R	V	R	V	R	V	R	V
(Hemaspaandra et al., 2007)																						
Copeland$^{\alpha}$																						
for $\alpha = 0$	V	V	R	V	R	V	R	V	R	V	R	V	R	V	R	R	R	R	R	R	R	R
$0 < \alpha < 1$	R	V	R	V	R	V	R	V	R	V	R	V	R	V	R	R	R	R	R	R	R	R
$\alpha = 1$	V	V	R	V	R	V	R	V	R	V	R	V	R	V	R	R	R	R	R	R	R	R
(Faliszewski et al., 2009c)																						
maximin	V	V	R	V	V	V	–	–	–	–	–	–	–	–	R	R	R	R	–	–	–	–
(Faliszewski et al., 2011b)																						
Borda	–	–	R	V	R	V	–	–	–	–	–	–	–	–	R	V	–	V	–	V	–	–
(Russel, 2007; Elkind et al., 2011a; Loreggia et al., 2014; Chen et al., 2015)																						
SP-AV	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	V	R	V	R	V	R	R
(Erdélyi et al., 2009)																						
fallback	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	V	R	V	R	R	R	R
(Erdélyi and Rothe, 2010; Erdélyi et al., 2011; see also Erdélyi et al., 2015a)																						
Bucklin	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	V	R	V	R	R	R	S
(Erdélyi et al., 2011; see also Erdélyi et al., 2015a)																						
RV	I	V	I	V	V	I	V	I	I	I	V	I	I	I	R	V	R	V	R	V	R	V
(Menton, 2013)																						
NRV	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	V	R	V	R	R	R	R
(Menton, 2013)																						
Schulze	R	S	R	S	R	S	R	V	R	V	R	V	R	V	R	V	R	V	R	R	R	R
(Parkes and Xia, 2012; Menton and Singh, 2013)																						

Key: "I" means immunity, "S" susceptibility, "V" vulnerability, and "R" resistance. We write "–" if a given result is not directly available in the literature.

fallback and Bucklin voting (Erdélyi and Rothe, 2010; Erdélyi et al., 2011, 2015a), NRV (Menton, 2013), and Schulze voting (Parkes and Xia, 2012; Menton and Singh, 2013), as shown in Table 7.3.

7.3.2 The Partitioning Cases and Destructive Control

In addition to control by adding/deleting candidates/voters, Bartholdi et al. (1992) also introduced various types of control by partitioning either candidates or voters, modeled

by elections proceeding in two stages. While their original definitions were a bit unclear about what happens when more than one candidate wins some first-stage pre-election, Hemaspaandra et al. (2007) defined two rules for how to handle such pre-election ties: TE ("ties eliminate") says that whenever at least two candidates are tied for winner in a pre-election, no candidate proceeds to the final stage from it (i.e., only unique pre-election winners move forward); TP ("ties promote") says that all pre-election winners, no matter how many, proceed forward.

Definition 7.3. Let f be a voting rule. In the CONSTRUCTIVE-CONTROL-BY-RUNOFF-PARTITION-OF-CANDIDATES problem for f under TE or TP (f-CCRPC-TE or f-CCRPC-TP), we are given (a) an election (A, R), and (b) a preferred candidate $p \in A$. We ask if we can partition A into A_1 and A_2 such that p is the unique f-winner of the election $(W_1 \cup W_2, R)$, where $W_i, i \in \{1, 2\}$, is the set of those pre-election (A_i, R) winners that are promoted to the final stage according to the tie-handling rule (TE or TP). The CONSTRUCTIVE-CONTROL-BY-PARTITION-OF-CANDIDATES problem for f under TE or TP (f-CCPC-TE or f-CCPC-TP) is defined similarly, except that we ask if p can be made a unique f-winner of the election $(W_1 \cup A_2, R)$ by partitioning A into A_1 and A_2, i.e., there is only one pre-election (A_1, R) whose winners proceed (according to the tie-handling rule, TE or TP) to the final stage to face all of A_2.

Example 7.5. Let (A, R) be the Borda election in Table 7.1(a) again, and let c be the distinguished candidate the chair wants to win. This is a yes-instance in all four cases, for both CCRPC and CCPC, each in TE and TP, as witnessed by the partition of A into $A_1 = \{a, f\}$ and $A_2 = \{b, c, d, e\}$. It does not matter whether we are in the TE or TP model,[4] since both subelections have a unique winner: a alone wins (A_1, R), and c alone wins (A_2, R) (with a score of 9, while b, d, and e score only 7, 6, and 8 points). For CCRPC, both subelection winners, c and a, proceed to the final stage, which c wins. For CCPC, the winner of the first subelection, a, faces all candidates of A_2 in the final stage, and as we have seen in Table 7.2(c), the unique Borda winner of $(\{a, b, c, d, e\}, R)$ is c, again as desired by the chair.

The analogues of f-CCRPC-TE/TP where not the candidates but the voters are partitioned model a very basic kind of *gerrymandering*. (Note that it would not make sense to define voter-partition analogues of f-CCPC-TE/TP, at least not for natural voting systems f.[5])

Definition 7.4. Let f be a voting rule. In the CONSTRUCTIVE-CONTROL-BY-PARTITION-OF-VOTERS problem for f under TE or TP (f-CCPV-TE or f-CCPV-TP), we are given (a) an election (A, R), and (b) a preferred candidate $p \in A$. We ask if R can be partitioned into R_1 and R_2 such that p is the unique f-winner of $(W_1 \cup W_2, R)$,

[4] By contrast, partitioning A into $A'_1 = \{a, b, c\}$ and $A'_2 = \{d, e, f\}$ would reveal a difference between the two tie-handling models: Since b and c tie for winning (A'_1, R), they both proceed to the final stage in model TP (where they face e, the winner of (A'_2, R), and c wins the final stage), but b and c eliminate each other in model TE (so e alone proceeds to the final stage and wins).

[5] In such an analogue, given an election (A, R) and $p \in A$, we would partition R into R_1 and R_2 just as in Definition 7.4, but there is only one pre-election, say (A, R_1), whose TE/TP-winners W_1 would then face *all* candidates in the final stage, yet $(W_1 \cup A, R) = (A, R)$ is just the original election.

where W_i, $i \in \{1, 2\}$, is the set of those pre-election (A, R_i) winners that are promoted to the final stage according to the tie-handling rule (TE or TP).

Example 7.6. Again looking at the Borda election (A, R) in Table 7.1(a) with preferred candidate c, the chair will succeed (in both TE and TP) by partitioning R into $R_1 = \{1, 2, 4, 5\}$ and $R_2 = \{3\}$: b alone wins (A, R_1) (with a score of 13, while a, c, d, e, and f score only 9, 10, 8, 12, and 8 points), and c alone wins (A, R_2). Thus they both proceed to the final stage where c beats b by 3 to 2.

The proofs that show the complexity of control by partitioning candidates or voters are often based on similar constructions as analogous proofs for the case of deleting candidates or voters, but usually are more involved technically.

For each constructive control problem, there is also a destructive variant, introduced by Hemaspaandra et al. (2007), where the chair's goal is to preclude a given candidate from being the unique winner of the election resulting from the chair's control action. We denote the destructive control problems analogously, replacing the initial "C" by a "D," as, for example, in DCDC for "destructive control by deleting candidates." (In this problem, it is forbidden to delete the designated candidate p; otherwise, the problem would be trivial.)

7.3.3 Overview and Some Other Approaches to Control

Table 7.3 summarizes the control complexity results for some prominent voting rules. In most cases we have full knowledge of the complexity of all the basic types of control, but for Borda and maximin some types of control were never studied, and for Bucklin and Schulze for some types of control there are only susceptibility results in the literature.

We already mentioned that besides Copeland$^\alpha$ voting, $0 < \alpha < 1$, also SP-AV (Erdélyi et al., 2009), fallback and Bucklin voting (Erdélyi and Rothe, 2010; Erdélyi et al., 2011, 2015a), NRV (Menton, 2013), and Schulze voting (Parkes and Xia, 2012; Menton and Singh, 2013) are resistant to all constructive control types. Among those, Schulze has many vulnerabilities to destructive control types, but SP-AV is vulnerable to only three of them (DCAV, DCDV, and DCPV-TE), and fallback, Bucklin, and NRV even to only two (DCAV and DCDV), where the case of Bucklin-DCPV-TP is still open. Note that SP-AV is a somewhat unnatural system (as has been discussed by Baumeister et al. (2010) in detail), due to a rule introduced by Erdélyi et al. (2009) that, to cope with certain control actions, can move the voters' approval lines *after they have cast their votes*. It may be argued that NRV has a similar issue (though perhaps to a lesser extent), since after the voters have cast their votes (namely, their range voting vectors), the normalization process can change the points the alternatives will score from the votes. Fallback's drawback, on the other hand, is that it is a hybrid of two "pure" voting rules, Bucklin and approval, and requires the voters to report both approval vectors and rankings. All three voting systems have the disadvantage that it is rather complicated (even though far less complicated than in Schulze voting) to determine the winners; for example, it is hardly conceivable that many of the voters in a real-world political election would be fully aware of the effect of normalization in NRV. But this is the price to pay if we wish to have a (relatively) natural voting rule

with P-time winner determination that is resistant to as many control actions as these voting rules are resistant to. On the other hand, if one is willing to accept an artificial voting rule, Hemaspaandra et al. (2009) have shown how to combine well-known rules to obtain ones that are resistant to all types of control. While their method produces rules that are not attractive in practice (even though they satisfy some natural normative properties), it suggests that indeed there might exist natural voting rules with P-time winner determination that are resistant to all types of control considered here.

Now, let us quickly point to some related work and to other approaches to control. Meir et al. (2008) study control for multiwinner voting rules. In the multiwinner setting, we are given an election (A, R) and an integer k, and the goal is to pick a "committee" of k winners. Multiwinner voting rules can be used to choose parliaments (or other collective bodies), to choose finalists in competitions, or even within recommendation systems (see, e.g., the work of Lu and Boutilier (2011a) for the application in recommendation systems and the work of Elkind et al. (2014a) for a recent general discussion of multiwinner voting). To study control in the multiwinner setting (and analogous approaches apply to other manipulative scenarios), Meir et al. (2008) assume that the election chair associates some utility value with each candidate and his or her goal is to ensure that the sum of the utilities of the candidates in the elected committee is as high as possible. As a side effect, this approach creates a natural unification of the constructive and destructive cases: In the constructive case the chair would have positive utility only for the most preferred candidate, whereas in the destructive setting the chair would have positive, equal utilities for all the candidates except the despised one.

Faliszewski et al. (2011b) provide a unified framework to capture "multimode control attacks" that simultaneously combine various of the control actions considered here. Specifically, in their setting the chair can, for example, simultaneously add some candidates and remove some voters. One of the conclusions of this work is that, typically, the complexity of such a multimode control attack is the same as that of the hardest basic control type involved. In particular, if a voting rule is vulnerable to several basic types of control, it is also vulnerable if the chair can perform these control types simultaneously (i.e., coordinating the attacks is easy). However, this conclusion is based on studying a number of natural voting rules and is not a general theorem (indeed, such a general theorem does not hold).

Fitzsimmons et al. (2013) study the complexity of control in the presence of manipulators, both in the case where the chair and the manipulators coordinate their actions and in the case where they compete with each other. While all the related cooperative problems are in NP, they show that the competitive problems can be complete for the second and the third level of the polynomial hierarchy for suitably designed artificial voting systems (though their complexity is much lower for many natural voting systems). Another approach to unifying different types of strategic behavior is due to Xia (2012b), who proposes a general framework that is based on so-called vote operations and can be used to express, for example, bribery and control by adding or deleting voters. Xia (2012b) shows that if the votes are generated i.i.d. with respect to some distribution, then, on the average, the number of vote operations (e.g., the number of voters that need to be added) necessary to achieve a particular effect (e.g., ensuring that some candidate is a winner) is either zero (the effect is already achieved), or is

proportional to the square root of the number of original voters, or is linear with respect to the number of original voters, or the effect is impossible to achieve.

Going in a somewhat different direction, Chen et al. (2014) consider control by adding candidates in a combinatorial setting, where one can add whole groups of voters at unit cost. They show that even for the plurality rule, for which standard voter control is very easy, the combinatorial setting is challenging (indeed, combinatorial control by adding voters is NP-hard for plurality even in the settings where the groups of voters to add contain at most two voters each).

Hemaspaandra et al. (2014b) established the first control-related dichotomy result, showing for which pure scoring rules CCAV is easy to solve and for which this problem is NP-complete (however, Faliszewski et al. (2013) study voter control in weighted elections and in the technical report version of their paper show a dichotomy result as well). This complements similar dichotomy results of Hemaspaandra and Hemaspaandra (2007) on manipulation and of Betzler and Dorn (2010) and Baumeister and Rothe (2012) on the possible winner problem.

Some researchers investigate not only the classical complexity, but also the parameterized complexity of control problems, with respect to such parameters as the solution size (e.g., "number of added voters") or the election size (e.g., "number of candidates"); for example, see the work of Liu et al. (2009) for a discussion of plurality, Condorcet, and approval; Liu and Zhu (2010, 2013) for maximin, Copeland, Borda, Bucklin, and approval; Betzler and Uhlmann (2009) and Faliszewski et al. (2009c) for Copeland$^{\alpha}$; Erdélyi et al. (2015a) for Bucklin and fallback; and Hemaspaandra et al. (2013b) for Schulze and ranked-pairs voting. On the other hand, Brelsford et al. (2008) study the approximability of control, manipulation, and bribery. Faliszewski et al. (2013) discuss approximation algorithms for voter control under k-approval.

Faliszewski et al. (2011b, 2011a) and Brandt et al. (2010a) study to what extent complexity shields for manipulation and control disappear in elections with domain restrictions, such as in single-peaked or nearly single-peaked electorates (see also the book chapter by Hemaspaandra et al., 2015). Magiera and Faliszewski (2014) show similar results for single-crossing electorates.

Hemaspaandra et al. (2013a) compare the decision problems for manipulation, bribery, and control with their search versions and study conditions under which search reduces to decision. They also notice that two destructive control types that previously have been viewed as distinct are in fact identical (in both the unique-winner and the nonunique-winner model): DCRPC-TE = DCPC-TE (and, in only the nonunique-winner model, they additionally show equality of another pair of control types: DCRPC-TP = DCPC-TP).

So far, almost all the research on the complexity of control has been theoretical, establishing NP-hardness of various problems. Recently, Rothe and Schend (2012) have initiated the experimental study of control (see also the survey by Rothe and Schend (2013) and the work of Erdélyi et al. (2015b)), showing that NP-hard control problems can, sometimes, be solved efficiently in practice (cf. the work of Walsh (2011a) for such studies on manipulation).

Finally, there are a number of problems that are very closely related to control, but that, nonetheless, are usually not classified as "standard control types." These problems include, for example, candidate cloning (see the brief discussion in Section 7.5), fixing

knockout tournaments (see Chapter 19 for more details), the problem of controlling sequential elections by choosing the order of voting on related issues (see the work of Conitzer et al., 2009a), and online control in sequential elections (see the work of Hemaspaandra et al., 2012a, 2012b), which is inspired by online manipulation in sequential elections due to Hemaspaandra et al. (2014a). Ideas originating from election control have also found applications in other settings. For example, Baumeister et al. (2012b, 2013b) have studied control for the case of judgment aggregation (for more details on judgment aggregation, see Chapter 17 and the book chapter by Baumeister et al. (2015)).

7.4 Bribery

Let us now move on to the study of bribery in elections. As opposed to the case of control, this time it is not possible to affect the structure of the election at hand (that is, the sets of candidates or voters cannot be changed), but it is possible to change some of the votes instead. Election bribery problems, introduced by Faliszewski et al. (2009b), model situations where an outside agent wants a particular alternative to win and pays some of the voters to vote as the agent likes. The problem name, bribery, suggests settings where an outside agent is dishonestly affecting election results, but there are other interpretations of these problems as well. For example, the formal framework of bribery can capture scenarios such as political campaign management and election fraud detection. We discuss such aspects of bribery (and control) in Section 7.5; for now we focus on the algorithmic properties of bribery problems without making judgments as to their morality.

The briber's task has two main components. First, the briber needs to decide *who* to bribe. Second, the briber has to decide *how* to change the chosen votes. In that sense, election bribery combines a control-like action (picking which voters to affect) with a manipulation-like action (deciding how to change the selected votes; see Chapter 6 and Section 4.3.3 in the book chapter by Baumeister and Rothe (2015) for more details on manipulation). Furthermore, it might be the case that while a voter agrees to change her vote in some ways, she may refuse to change it in some other ways (e.g., the voter might agree to swap the two least preferred alternatives, but not to swap the two most preferred ones). The following definition, based on the ones given by Faliszewski et al. (2009b) and—later—by Elkind et al. (2009c),[6] tries to capture these intuitions. (A careful reader should see that this definition is not sufficient for algorithmic applications; however, it will be a convenient base for further refinements.)

Definition 7.5. Let f be a voting rule. In the priced bribery problem for f, we are given (a) an election (A, R), where the set of voters is $N = \{1, \ldots, n\}$ and R contains a preference order $\succ_i$ for each $i \in N$, (b) a preferred alternative $p \in A$, (c) a budget $B \in \mathbb{N}$, and (d) a collection of price functions $\Pi = (\pi_1, \ldots, \pi_n)$. For each i, $1 \leqslant i \leqslant n$, and each preference order $\succ$ over A, $\pi_i(\succ)$ is the cost of convincing the ith voter to

[6] To provide historical perspective, let us mention that the paper of Faliszewski et al. (2009b) was presented in 2006 at the 21st National Conference on Artificial Intelligence (AAAI).

cast vote $\succ$ (we require that for each i, $1 \leqslant i \leqslant n$, $\pi_i(\succ_i) = 0$). We ask if there exists a preference profile $R' = (\succ'_1, \ldots, \succ'_n)$ such that (*i*) p is an f-winner of election (A, R'), and (*ii*) $\sum_{i=1}^{n} \pi_i(\succ'_i) \leqslant B$.[7]

Informally speaking, in condition (*i*) we require that the bribery is successful (p becomes a winner) and in condition (*ii*) we require that it is cheap enough (i.e., within our budget B). However, it is impossible to use this definition directly in our algorithmic analysis. The problem is that given an election (A, R), each price function should be defined for $|A|!$ different arguments. If we represented each price function by listing all the $|A|!$ argument-value pairs, the encoding of the problem would grow exponentially and for most natural voting rules the problem could be solved by brute force (yet without giving any real insight into the nature of election bribery). In other words, to make the problem interesting (and practical), we have to limit our attention to families of price functions that can be described succinctly. To this end, researchers have mostly focused on the following families of functions (in the following description we use the notation from Definition 7.5; we use the terms *discrete* and *$discrete* to unify the discussion of bribery problems even though these terms did not appear in the original papers):

1. We say that the price functions are *discrete* if for each π_i, $1 \leqslant i \leqslant n$, and for each preference order $\succ$, it holds that $\pi_i(\succ) = 0$ if $\succ = \succ_i$, and $\pi_i(\succ) = 1$ otherwise.
2. We say that the price functions are *$discrete* if for each π_i, $1 \leqslant i \leqslant n$, there is an integer c_i such that for each preference order $\succ$, it holds that $\pi_i(\succ) = 0$ if $\succ = \succ_i$, and $\pi_i(\succ) = c_i$ otherwise. (Each voter can have a different value c_i.)
3. We say that the price functions are *swap-bribery price functions* if for each π_i, $1 \leqslant i \leqslant n$, and for each two alternatives $x, y \in A$, there is a value $c_i^{\{x,y\}}$ such that for each preference order $\succ$, $\pi_i(\succ)$ is the sum of the values $c_i^{\{x,y\}}$ such that $\succ$ ranks x and y in the opposite order than $\succ_i$ does.

That is, discrete functions give cost one for changing a vote (irrespective of which vote it is or how it is changed), $discrete functions give a (possibly different) cost for changing each vote (irrespective of the nature of the change), and swap-bribery price functions define a cost for swapping each two alternatives and, then, sum up these costs. Clearly, functions in each of these families can be described succinctly.

From the historical perspective, the first paper on the complexity of bribery in elections (due to Faliszewski et al., 2009b) focused largely on discrete and $discrete functions. Swap-bribery functions were introduced first by Faliszewski et al. (2009c) in the context of so-called irrational votes, and were later carefully studied by Elkind et al. (2009c) in the standard setting of linear preference orders. Naturally, one can also define other families of cost functions (and some researchers—including the ones just cited—have done so) but in this chapter we will focus on these three.

Definition 7.5 can be applied to weighted elections as well. In such a case, it is tempting to introduce some explicit relation between the voters' weights and the costs

[7] As opposed to the case of control, research on bribery typically focuses on the nonunique-winner model; the unique-winner model has been considered in addition in some papers on bribery (see, e.g., Faliszewski et al., 2015).

of changing their votes. However, doing so is not necessary and we assume that such dependencies, if needed, are embedded in the price functions.

7.4.1 Bribery, Weighted-Bribery, $Bribery, Weighted-$Bribery, and Swap-Bribery

We focus on the bribery problems that can be derived using discrete, $discrete, and swap-bribery price functions. For the former two, consider the following definition.

Definition 7.6 (Faliszewski et al., 2009b). Let f be a voting rule. By f-BRIBERY we denote the priced bribery problem with discrete price functions and by f-$BRIBERY we denote the priced bribery problem with $discrete price functions. The problems f-WEIGHTED-BRIBERY and f-WEIGHTED-$BRIBERY are defined in the same way, but for weighted elections.

Example 7.7. Consider the Borda election in Table 7.1(a) and suppose that each voter has the same unit price, and that the goal is to ensure the victory of f through bribery. Prior to the bribery, b has 16 points and f has 9. It suffices to bribe voter 3 to cast vote $f\ d\ a\ c\ e\ b$. (Afterward, b, e, and f have score 13 each, and a, c, and d have score 12 each.) This means that there is a successful bribery with cost one. On the other hand, if voters 1 and 5 had cost one and the remaining voters had cost three each, then it would be better to bribe voters 1 and 5 to shift f to the top positions in their votes.

For swap-bribery price functions, Elkind et al. (2009c) have defined the following problem (they have not studied swap bribery for weighted elections).

Definition 7.7 (Elkind et al., 2009c). Let f be a voting rule. By f-SWAP-BRIBERY we denote the priced bribery problem with swap-bribery price functions.

Example 7.8. Consider the Borda election in Table 7.1(a) once again. This time, by applying swap bribery, we want to ensure victory of candidate d. We assume that swapping each two adjacent candidates has unit cost. Prior to the bribery, b has 16 points, c has 15 points, d and e have 12 points, a has 11 points, and f has 9 points. We perform the bribery as follows: We swap b in the preference order of voter 1 first with f, then with e, and finally with d. This way b loses three points and d, e, and f gain one point each. Thus b, d, and e have score 13 each, a and f score less than 13 points, but c still has 15 points. So, next we swap c and d in the preference order of voter 3. This way both c and d have score 14 and they both tie as winners. This is a successful swap bribery of cost four (and, indeed, it is the cheapest successful swap bribery for d in this scenario).

To familiarize ourselves with bribery problems further, let us consider their complexity for the plurality rule.

Theorem 7.9. *For plurality voting it holds that:*

1. BRIBERY, WEIGHTED-BRIBERY, *and* $BRIBERY *are each in* P, *but* WEIGHTED-$BRIBERY *is* NP*-complete (Faliszewski et al., 2009b), and*
2. SWAP-BRIBERY *is in* P *(Elkind et al., 2009c).*

It is easy to see that plurality-BRIBERY can be solved by (repeating in a loop) the following greedy algorithm: If the preferred alternative is not a winner already,

then pick one of the current winners and bribe one of her voters to vote for the preferred alternative. Unfortunately, such greedy approaches do not work for plurality-WEIGHTED-BRIBERY. For example, consider an algorithm that works in iterations and in each iteration bribes the heaviest voter among those that vote for one of the current winners. Let (A, R) be an election where $A = \{p, a, b, c\}$ and where we have 9 weight-1 voters voting for a, a single weight-5 voter voting for b, and a single weight-5 voter voting for c. Clearly, it suffices to bribe the two weight-5 voters, but the heuristic would bribe five voters with weight 1 each. On the other hand, bribing the heaviest voter first does not always work either (Faliszewski et al. (2009d) give a counterexample with $A = \{p, a, b\}$, p receiving no votes at first, a receiving three weight-2 votes and one weight-1 vote, and b receiving two weight-3 votes; to make p a winner it suffices to bribe one weight-2 vote and one weight-3 vote, but the heuristic bribes three votes). Nonetheless, a combination of these two heuristics does yield a polynomial-time algorithm for plurality-WEIGHTED-BRIBERY.

Let us consider some weighted plurality election and let us say that somehow we know that after an optimal bribery, our preferred alternative p has at least T points. Naturally, all the other alternatives have to end up with at most T points (and we can assume that at least one of them will get exactly T points). Thus for each alternative a that has more than T points, we should keep bribing its heaviest voters until its score decreases to at most T (this corresponds to running the *bribe the current winner's heaviest voter* heuristic). If, after bringing each alternative to at most T points, the preferred alternative still does not have T points, we bribe the globally heaviest voters to vote for the preferred alternative. We do so until the preferred alternative reaches at least T points (this corresponds to running the *bribe the heaviest voter* heuristic). If we chose the value of T correctly, by this point we would have found an optimal bribery strategy. But how do we choose T? If the weights were encoded in unary, we could try all possible values, but doing so for binary-encoded weights would give an exponential-time algorithm. Fortunately, we can make the following observation: For each alternative a, we bribe a's voters in the order of their nonincreasing weights. Thus, after executing the above-described strategy for some optimal value T, a's score is in the set {a's original score, a's score without its heaviest voter, a's score without its two heaviest voters, ...}. Thus it suffices to consider values T of this form only (for each candidate) and to pick one that leads to a cheapest bribery.

It is an easy exercise for the reader to adapt the plurality-WEIGHTED-BRIBERY algorithm to the case of plurality-$BRIBERY. On the other hand, solving plurality-SWAP-BRIBERY requires a somewhat different approach. The reason is that under SWAP-BRIBERY it might not always be optimal to push our preferred candidate to the top of the votes, but sometimes it may be cheaper and more effective to replace some high-scoring alternatives with other, low-scoring ones. To account for such strategies, Elkind et al. (2009c) compute, for each vote v, the lowest cost of replacing v's current top-alternative with each other one, and then run a flow-based algorithm of Faliszewski (2008) to find the bribing strategy. We omit the details here.

For plurality-WEIGHTED-$BRIBERY, it is easy to see that the problem is in NP and so we only show NP-hardness. We give a reduction from the PARTITION problem to plurality-WEIGHTED-$BRIBERY. Recall that in the PARTITION problem the input consists of a sequence of positive integers that sum up to some value S, and we ask

Table 7.4. *The complexity of f-*BRIBERY *for various voting rules*

f	f-BRIBERY	reference
plurality	P	Faliszewski et al. (2009b)
veto	P	Faliszewski et al. (2009b)
2-approval	P	Lin (2012)
k-veto, $k \in \{2, 3\}$	P	Lin (2012)
k-approval, $k \geqslant 3$	NP-complete	Lin (2012)
k-veto, $k \geqslant 4$	NP-complete	Lin (2012)
Borda	NP-complete	Brelsford et al. (2008)
STV	NP-complete	Xia (2012a)
Bucklin	NP-complete	Faliszewski et al. (2015)
fallback	NP-complete	Faliszewski et al. (2015)
maximin	NP-complete	Faliszewski et al. (2011b)
Copeland	NP-complete	Faliszewski et al. (2009c)
Schulze	NP-complete	Parkes and Xia (2012)
ranked pairs	NP-complete	Xia (2012a)
approval	NP-complete	Faliszewski et al. (2009b)
range voting	NP-complete	follows from the approval result

if it is possible to partition this sequence into two subsequences that both sum up to $S/2$ (naturally, for that S needs to be even). Let $(s_1, \ldots, s_n)$ be the input sequence and let $S = \sum_{i=1}^{n} s_i$. We form an election (A, R), with $A = \{p, d\}$ and with R containing n voters voting for d; for each i, $1 \leqslant i \leqslant n$, the ith voter has weight s_i and her price function is *"it costs s_i to change the vote."* The budget B is $S/2$. In effect, any bribery of cost at most B can give p a score of at most $S/2$. The only such briberies that would ensure that p is among the winners must give p score exactly $S/2$, by solving the original PARTITION instance. This result is particularly useful because its proof easily adapts to most other typical voting rules, showing that WEIGHTED-$BRIBERY is NP-complete for them as well.

Theorem 7.9 suggests that, perhaps, for various voting rules f, not only is f-BRIBERY easy but so are even its more involved variants, f-$BRIBERY and f-WEIGHTED-BRIBERY. However, in-depth study of f-BRIBERY has shown that the problem is NP-complete for most natural voting rules f. We survey these results in Table 7.4. Naturally, the hardness results for BRIBERY immediately transfer to $BRIBERY and WEIGHTED-BRIBERY.

Theorem 7.10 (Faliszewski et al., 2009b). *For each voting rule f, f-*BRIBERY *reduces to f-*$BRIBERY *and to f-*WEIGHTED-BRIBERY.

Furthermore, for the case of $BRIBERY we can inherit multiple hardness results from the coalitional manipulation problem, through a simple reduction.

Definition 7.8 (Conitzer et al., 2007). Let f be a voting rule. In the (constructive, coalitional) f-MANIPULATION problem we are given (a) an election (A, R), (b) a preferred alternative $p \in A$, and (c) a collection R' of voters with unspecified preference orders. We ask if it is possible to ensure that p is an f-winner of election $(A, R + R')$

by setting the preference orders of the voters in R'. The (constructive, coalitional) f-WEIGHTED-MANIPULATION problem is defined analogously, but for weighted elections, where the manipulators' weights are given.

Theorem 7.11 (Faliszewski et al., 2009b). *For each voting rule f, f-*MANIPULATION *reduces to f-*$BRIBERY*, and f-*WEIGHTED-MANIPULATION *reduces to f-*WEIGHTED-$BRIBERY.

For the case of SWAP-BRIBERY, hardness results are even more abundant. Elkind et al. (2009c) have shown that the problem is NP-complete for k-approval (for $k \geqslant 2$)[8] and for Borda, Copeland, and maximin (for the latter three systems, NP-hardness holds even for SHIFT-BRIBERY, a special case of SWAP-BRIBERY where the swaps have to involve the preferred candidate). Furthermore, the SWAP-BRIBERY problem generalizes the POSSIBLE-WINNER problem, which itself generalizes the MANIPULATION problem.

Definition 7.9 (Konczak and Lang, 2005). Let f be a voting rule. In the f-POSSIBLE-WINNER problem we are given (a) an election (A, R), where the voters in R are represented through (possibly) partial orders, and (b) an alternative $p \in A$. We ask if it is possible to extend the partial orders in R to linear orders in such a way that p is an f-winner of the resulting election.

Theorem 7.12 (Elkind et al., 2009c). *For each voting rule f, f-*POSSIBLE-WINNER *reduces to f-*SWAP-BRIBERY.

Xia and Conitzer (2011a) have shown hardness of POSSIBLE-WINNER for a number of voting rules (including STV, ranked pairs, Borda, Copeland, maximin, and many other rules); Betzler and Dorn (2010) together with Baumeister and Rothe (2012) show a dichotomy result regarding the complexity of POSSIBLE-WINNER for pure scoring rules, obtaining hardness for almost all of them (see Chapter 10 and Section 4.3.2 in the book chapter by Baumeister and Rothe (2015) for more details on the POSSIBLE-WINNER problem and on related issues). By Theorem 7.12, these hardness results immediately translate to hardness results for SWAP-BRIBERY and the same voting rules.

Such an overwhelming number of hardness results (either shown directly or implied by Theorems 7.11 and 7.12) suggests that, perhaps, SWAP-BRIBERY is too general a problem. That is why Elkind et al. (2009c) defined SHIFT-BRIBERY, a variant of SWAP-BRIBERY where, as mentioned earlier, the only legal briberies shift the preferred candidate up in the voters' preference orders. While this problem turned out to typically be NP-complete as well, Elkind et al. (2009c), Elkind and Faliszewski (2010), and Schlotter et al. (2011) have found some interesting polynomial-time algorithms, exact and approximate, and Bredereck et al. (2014b) have studied the parameterized complexity of SHIFT-BRIBERY (see Section 7.5 for more motivating discussions regarding SHIFT-BRIBERY).

We now show that (the optimization variant of) Borda-SHIFT-BRIBERY can be efficiently approximated within a factor of 2.

Theorem 7.13 (Elkind et al., 2009c). *There is a polynomial-time 2-approximation algorithm for the cost of a cheapest shift bribery under Borda voting.*

[8] The result for $k = 2$ follows from the work of Betzler and Dorn (2010); for $k = 1$ the problem is in P; for $k = m/2$, where m is the number of alternatives, Elkind et al. (2009c) have shown hardness even for the case of a single voter.

Proof sketch. Consider an instance of our problem where the goal is to ensure candidate p's victory. By definition, the only possible actions are shifting p forward in (some of) the votes (costs are specified through swap-bribery price functions where swaps that do not involve p have infinite cost and we can think of shifting p forward in terms of its swaps with other candidates).

We start with two observations. First, there is a polynomial-time algorithm that given an instance of the optimization variant of Borda-SHIFT-BRIBERY computes the cost of a cheapest shift bribery that gives p a given number of points (the algorithm uses standard dynamic programming). Second, if there is a successful shift bribery that increases the score of p by K points, then every shift bribery that increases p's score by $2K$ points is successful (the best imaginable shift bribery gets K points for p in such a way that in each swap it increases the score of p and decreases the score of its strongest competitor; we achieve the same—or better—effect by getting $2K$ points for p).

Now the algorithm proceeds as follows: First, we guess the number K of points that p gets in the optimal solution. Then, we guess a number K', $K' \leqslant K$. (Because we are dealing with Borda elections, both guesses boil down to trying polynomially many computation paths.) We compute a cheapest shift bribery S_1 that gives K points to p. Then, we compute a cheapest shift bribery S_2 that gives K' additional points to p (we apply S_2 after we have applied S_1). We claim that $S_1 + S_2$ (that is, the two shift briberies taken together) form a 2-approximate solution.

Why is this so? Consider some optimal shift bribery O that ensures that p wins. By assumption, this shift bribery obtains K points for p. Now imagine the following situation: We start with the original election and perform only those swaps that are included in both O and S_1. In effect, p gains some K'' points. If we continued with the optimal solution, p would obtain additional $K - K''$ points and would become a winner of the election. By our second observation, this means that if after performing the swaps that occur both in O and in S_1 we obtain additional $2(K - K'')$ points for p, p certainly wins. We obtain the first of these $K - K''$ points by simply performing the remaining swaps from S_1. For the second $K - K''$ points, we can assume that we guessed $K' = K - K''$. In effect, performing the swaps from S_2 ensures p's victory. Furthermore, by definition of S_1 we know that its cost is no higher than that of O. On the other hand, the cost of S_2 also has to be at most as high as that of O because, by definition, the cost of S_2 cannot be higher than the cost of the shift bribery that contains exactly the swaps that are in O but not in S_1. □

So far, there has been relatively little research on how to cope with the hardness of bribery problems (except for results regarding special cases such as SHIFT-BRIBERY, as seen in the preceding theorem). For example, many parameterized-complexity results boil down to polynomial-time algorithms for the case where the number of candidates is constant. In this case, bribery problems can either be solved by an appropriate brute-force search, or by solving a linear integer program using the algorithm of Lenstra, Jr. (1983); see the papers of Faliszewski et al. (2009b, 2011b), Elkind et al. (2009c), Dorn and Schlotter (2012), and Hemaspaandra et al. (2013b) for examples. These approaches, however, do not work for weighted elections and, indeed, for weighted elections bribery problems are typically NP-hard (see, e.g., the dichotomy results of Faliszewski et al. (2009b)). On the other hand, there are several detailed

studies of parameterized complexity of SWAP-BRIBERY (due to Dorn and Schlotter, 2012), SUPPORT-BRIBERY (Schlotter et al. (2011); we omit the discussion of SUPPORT-BRIBERY), and SHIFT-BRIBERY (Bredereck et al., 2014b).

Another natural way of coping with the hardness of bribery problems would be to design approximation algorithms. Brelsford et al. (2008) have made some attempts in this direction (though, using a rather involved goal function instead of approximating the cost of a successful bribery), Faliszewski (2008) gave a fully polynomial-time approximation scheme for plurality-WEIGHTED-$BRIBERY, and Xia (2012a) gave several approximation algorithms for destructive bribery problems (where the goal is to ensure, through buying votes, that some candidate does *not* win the election). There are also approximation results regarding SHIFT-BRIBERY (due to Elkind and Faliszewski, 2010; Bredereck et al., 2014b). While surprising at first, this limited enthusiasm for studying approximation algorithms for bribery problems can, to some extent, be understood. Theorems 7.11 and 7.12 show how to reduce the MANIPULATION and POSSIBLE-WINNER problems to appropriate $BRIBERY and SWAP-BRIBERY problems, and they do so via showing that a given MANIPULATION (POSSIBLE-WINNER) instance is a "yes" instance if and only if there is a zero-cost bribery. This means that, unless P = NP, those $BRIBERY and SWAP-BRIBERY problems whose hardness can be shown via Theorems 7.11 and 7.12 do not have constant-factor polynomial-time approximation algorithms (for finding the cheapest successful bribery). Nonetheless, it is interesting to study the approximability of f-BRIBERY for various voting rules f.

It would also be interesting to study the complexity of bribery in elections with restricted domains, for example, in single-peaked elections. While this direction has been pursued successfully for the case of control, we are aware of only a single paper that attempted it for bribery (Brandt et al., 2010a), showing that, indeed, for single-peaked elections bribery problems often become easy (see also Section 5.4 in the book chapter by Hemaspaandra et al. (2015)).

7.4.2 Other Bribery Problems

So far, we have focused on the most standard election model, where voter preferences are represented by total orders over the set of alternatives. Naturally, there are numerous other settings in which bribery was studied, and in what follows we give several (though certainly not all) examples of such settings.

Mattei et al. (2012a) have considered bribery in combinatorial domains, where the voters express their preferences over bundles of alternatives in a certain compact way. This compact representation can lead to quite interesting results. The particular language used to express preferences in the work of Mattei et al. (2012a) (CP-nets) does not allow one to express certain preference orders and, as a result, BRIBERY for k-approval becomes easy in this model (see Chapter 9 for more details on voting in combinatorial domains). If there are no direct interrelations between the bundles of items, it may be more reasonable to model bribery as the *lobbying problem* (studied by Christian et al. (2007) and later on by Bredereck et al. (2014a) and Binkele-Raible et al. (2014)): We are given a collection of yes/no votes over all items independently, where an item is accepted with a simple majority of yes votes, and is rejected otherwise. The

lobby's goal is to change the outcome to its liking by bribing certain voters without exceeding its budget.

Examples of bribery problems in other settings include, for example, the work of Baumeister et al. (2011) on bribery in judgment aggregation (see Chapter 17 and the book chapter by Baumeister et al. (2015) for more details on judgment aggregation), the work of Rey and Rothe (2011) and Marple et al. (2014) on bribery in path-disruption games, and the work of Mattei et al. (2012b) on bribery in tournaments.

7.5 A Positive Look

There are a number of settings where control and bribery (and similar problems) have positive interpretations (from particular points of view). In the following we very briefly list a few examples of such settings.

Election control problems deal with affecting their structure in order to change the winner. Instead of viewing this as someone manipulating the result, we can think of it as predicting the winners given how the election's structure may change. For example, this research direction was pursued by Chevaleyre et al. (2012) and Baumeister et al. (2012c). Specifically, Chevaleyre et al. (2012) have studied a situation where we have already elicited voters' preferences regarding some set of candidates, but afterward some new candidates appeared, of whom we have no knowledge whatsoever. Naturally, possibly each new candidate can be better than each old one, so each of them, possibly, might win the election. However, can we decide which of the original candidates still have chances of winning? This problem of predicting possible winners is very close in spirit to control by adding candidates (and to cloning; see later), though—formally—it is a special case of the Possible-Winner problem (and, as such, it is a special case of the Swap-Bribery problem).

Another way of predicting election winners was suggested by Wojtas and Faliszewski (2012), who have used counting variants of election control problems. In particular, they considered the following setting: We know the preference orders of the voters, but we do not know which of them will eventually cast votes. Having some prior distribution on the number of voters that do cast votes (and assuming that if k voters participate in the election, then each size-k subset of voters is equally likely to vote), what is the probability that a given candidate wins? Formally, this problem reduces to counting the number of ways of adding (deleting) voters to (from) an election to ensure a given candidate's victory.

Quite interestingly, many of the problems that model attacks on elections have direct applications in protecting them. For example, in the margin-of-victory problem (see, e.g., the work of Cary (2011), Magrino et al. (2011), Xia (2012a), and Reisch et al. (2014)) we ask how many voters need to cast different votes to change the result of an election. If this number is high then it is unlikely that the election was tampered with. However, if this number is low, it means that it would have been easy to manipulate the result in some way and thus we should carefully check the election. The margin-of-victory problem is, in some sense, simply a destructive bribery problem. Similarly, Birrell and Pass (2011) have used bribery-related problems in the context of approximate strategyproofness of voting rules. Yet another application of a control-like

problem to protect elections was given by Elkind et al. (2012a). They have considered the problem of candidate cloning, where some candidate c is replaced by a number of clones, $c_1, \ldots, c_t$, that—from the point of view of the voters—are indistinguishable (consider, for example, a party submitting several candidates for a given position and the voters forming their preference orders based on party membership only). If an election is single-peaked and we clone a candidate, it is likely that this election ceases to be single-peaked. Motivated by this observation, Elkind et al. (2012a) have given an algorithm that finds an optimal "decloning" of the candidates, so that the resulting election is single-peaked (similar results, though in a different context, were later given by Cornaz et al. (2012, 2013); we also mention that cloning, originally defined by Tideman (1987) and by Zavist and Tideman (1989), resembles control by adding candidates; its computational analysis is due to Elkind et al. (2011a)).

Finally, let us mention some positive interpretations of bribery problems. In political elections, prior to casting the votes, the candidates run their campaigns and wish to convince the voters to rank them as highly as possible. Naturally, running a campaign has cost (both in terms of money and in terms of invested time) and it is important for the candidates to decide which voters they should try to convince. However, deciding how much effort to spend on each voter (or, group of voters) is just the bribery problem (see the work of Hazon et al. (2013) for a different twist on this idea). With the campaign management interpretation in mind, it is natural to study various special cases of the bribery problems. Indeed, Shift-Bribery of Elkind et al. (2009c), where we can only convince the voters to rank the preferred candidate higher and we cannot affect the relative order of the other candidates, models campaign management in a natural way. While the Shift-Bribery problem is NP-hard for many voting rules, Elkind et al. (2009c) have given a 2-approximation algorithm for this problem with Borda's rule (see Theorem 7.13 here), Elkind and Faliszewski (2010) have extended this result to all scoring rules (and provided weaker approximations for Copeland and maximin), and Schlotter et al. (2011) have shown that Shift-Bribery is in P for Bucklin and fallback voting. These results for Bucklin and fallback voting were recently complemented by Faliszewski et al. (2015) who studied various bribery problems for these rules, including so-called Extension-Bribery, introduced by Baumeister et al. (2012a) in the context of campaign management in the presence of truncated ballots.

7.6 Summary

We surveyed the known results on control and bribery. While often studied in the context of attacking elections, these problems also have many other applications and interpretations, often very positive ones. Many NP-hardness results have been obtained, yet recent work focuses on solving these problems effectively, either by approximation or fixed-parameter tractable algorithms, or efficient heuristics. We strongly encourage the readers to study control and bribery and to add their own contributions to the field.