CHAPTER 6

# Barriers to Manipulation in Voting

Vincent Conitzer and Toby Walsh

## 6.1 Introduction

In many situations, voters may vote strategically. That is, they may declare preferences that are not their true ones, with the aim of obtaining a better outcome for themselves. The following example illustrates this.

**Example 6.1.** Consider an election with three alternatives, $a$, $b$, and $c$, and three voters, 1, 2, and 3. Suppose the rule used is plurality—an alternative gets a point each time it is ranked first by a voter, and the alternative with the most points wins—with ties broken toward alternatives earlier in the alphabet. Suppose voter 3 knows (or strongly suspects) that voter 1 will rank $a$ first in her vote, and that voter 2 will rank $b$ first. Voter 3's true preferences are $c \succ b \succ a$. If she votes truthfully, this will result in a three-way tie, broken in favor of $a$ which is 3's least preferred alternative. If, instead, voter 3 ranks $b$ first, then $b$ will win instead. Hence, voter 3 has an incentive to cast a vote that does not reflect her true preferences.

This is often referred to as *manipulation* or *strategic voting*; we will use "manipulation" throughout.[1] Voting rules that are never manipulable are also referred to as *strategyproof*. We start by reviewing the Gibbard-Satterthwaite impossibility result (discussed also in Chapter 2), which states that with unrestricted preferences over three or more alternatives, only very unnatural rules are strategyproof. The main focus of the chapter is on exploring whether computational complexity can be an effective barrier to manipulation. That is, we may not be concerned about manipulation of a voting rule if it is computationally hard to discover how to manipulate it.

[1] Of course, one may disagree, at least in some circumstances, that strategic voting is really "manipulative" in the common sense of the word. We simply use "manipulation" as a technical term equivalent to strategically reporting one's preferences incorrectly. Nevertheless, we will give some reasons why it can be undesirable in what follows.

## 6.2 Gibbard-Satterthwaite and Its Implications

An important axiomatic result about the properties of voting rules is the Gibbard-Satterthwaite Theorem:

**Theorem 6.2 (Gibbard, 1973; Satterthwaite, 1975).** *Consider a (resolute)*[2] *voting rule that is defined for some number $m$ of alternatives with $m \geqslant 3$, with no restrictions on the preference domain. Then, this rule must be at least one of the following:*

*1.* dictatorial*: there exists a single fixed voter whose most-preferred alternative is chosen for every profile;*
*2.* imposing*: there is at least one alternative that does not win under any profile;*
*3.* manipulable *(i.e., not strategyproof).*

Properties 1 and 2 are not acceptable in most voting settings. Hence, under the conditions of the theorem, we are stuck with property 3: there will exist profiles such that at least one of the voters has an incentive to misreport her preferences.

Before discussing how we might address this, we should first discuss why manipulability is a significant problem. It may not seem so. For example, consider a plurality election with three alternatives. If one of the candidates[3] is considered to have a poor chance of winning the election (consider, for example, a third party in the United States), then everyone might vote for one of the other two candidates, in order to avoid wasting their votes. Is this a significant problem? Will it not simply result in the same winner that plurality-with-runoff (or STV)[4] would have chosen (if everyone had voted truthfully), and is that so bad? Additionally, there are those who argue that democrats should not be worried about manipulation (Dowding and Hees, 2008). There are, however, several potential downsides to such manipulation, including the following. (Formalizing all these downsides would go beyond the scope of this chapter, so we present them informally; we hope the reader would be able to formalize these concepts if needed.)

- *Bad equilibria.* In the above example, it is not at all clear that the resulting winner will be the same as the true plurality-with-runoff winner. All that is required is that voters *expect* the third alternative to have poor chances. It is possible that this alternative is actually very much liked across the electorate, but nobody is aware of this. Even more strikingly, it is possible that everyone *is* aware of this, and yet the alternative is expected to perform poorly—for example, because nobody is aware that others are aware of the alternative's popularity. Hence, an alternative that is very much liked, and perhaps would have won under just about any reasonable rule had everyone voted truthfully, may not win.

[2] Recall that a voting rule is *resolute* if it returns only a single alternative for every profile.
[3] We use "alternatives" and "candidates" interchangeably.
[4] Recall that under the plurality-with-runoff rule, the alternatives with the top two plurality scores proceed to a runoff round, and the one that is preferred to the other by more voters wins. Under STV (also known as Instant Runoff Voting), only the alternative with the lowest plurality score is eliminated in each round; it is then removed from all the votes, so that votes that ranked it first now rank another alternative first. This procedure is repeated until only one alternative—the winner—remains. (For an axiomatization of this rule, see Freeman et al. (2014).)

- *Lack of information*. Even if the bad equilibria described above are in fact avoided, we cannot be sure that this is the case, because we will never know exactly how popular that third alternative really was. This also interferes with the process of identifying more desirable alternatives in the next election.
- *Disenfranchisement of unsophisticated voters*. Voters who are less well informed may end up casting less effective votes than those who are well informed (for example, votes for the third alternative). Knowledge is power—but in many elections, this is not considered desirable.
- *Wasted effort*. Even if all agents manipulate to the same extent, still much effort, whether of the computational, information gathering, or communicational variety, is expended in figuring out how to manipulate well, and presumably this effort could have been more productively spent elsewhere. This can be seen as a type of tragedy of the commons; everyone would be better off if nobody spent effort on manipulation, but individually voters are still better off manipulating.

In the theory of *mechanism design*—which applies not only to the design of voting rules but also to that of auctions, matching mechanisms, and any other setting where a decision must be made based on the preferences of multiple strategic agents—there is generally a focus on designing mechanisms in which agents have no incentive to misreport their preferences. This is justified by a result known as the *revelation principle*. Stating it formally here would take us too far afield, but roughly speaking, it says that for any mechanism that results in a good equilibrium (in a game-theoretic sense), there exists another mechanism that results in the same outcomes, but in which agents report their preferences directly and they have no incentive to misreport them.[5] That is, at some level, we should be able to get incentives to report truthfully (i.e., use a truthful mechanism) for free. The revelation principle has been criticized on the basis that it implicitly assumes agents to be computationally unbounded, and indeed it has been shown that in some cases there exist mechanisms (that are not truthful) that will perform at least as well as any truthful mechanism, and strictly better if agents are unable to compute their strategically optimal actions (Conitzer and Sandholm, 2004).

Taken together, there seem to be several arguments for attempting to erect barriers to manipulation. However, the Gibbard-Satterthwaite Theorem poses a fundamental limit to such barriers. How can we get around it? We will first discuss some avenues that are not computational in nature. Then, we devote most of the chapter to computational avenues.

## 6.3 Noncomputational Avenues around Gibbard-Satterthwaite

One way of sidestepping the Gibbard-Satterthwaite Theorem is to restrict the domain of preferences. Probably the best-known such restriction is that of *single-peaked*

[5] It should be noted that the notion of not having any incentive to misreport here is weaker than strategyproofness. Rather, it is *Bayes-Nash equilibrium*, which means that an agent is best off telling the truth *in expectation* over a prior distribution over the other agents' preferences—but the agent might be better off misreporting for a particular realization of the reports. There is a version of the revelation principle that results in a strategyproof mechanism, but this requires the original mechanism to have dominant strategies for all agents.

*preferences*. Here, the assumption is that there exists an ordering $<$ of the alternatives—for example, political candidates may be ordered on the left-to-right political spectrum, or the alternatives may be tax rates, locations along a single road, and so on. Moreover, the following assumption is made: if voter $i$'s most-preferred alternative is $a$, and $a < b < c$ or $c < b < a$, then $b \succ_i c$ ($i$ prefers $b$ to $c$). In this case (assuming, for simplicity, an odd number of voters) consider the *median voter rule*: order the voters by their most-preferred alternatives, and choose the median voter's most-preferred alternative. (Note that this rule does not require voters to specify preferences beyond their top choice.) This rule is strategyproof and always elects a Condorcet winner[6]. Of course, the usefulness of this result is limited by the fact that we cannot simply *make* the voters' preferences single-peaked when they are not. We could declare any vote that is not single-peaked invalid, but this just comes down to forcing voters to manipulate. For more discussion of single-peaked preferences, see Chapter 2.

Another possible avenue is to use *randomized* rules, which map every profile of votes to a probability distribution over the alternatives. For example, if we break the ties of a voting rule randomly, then we have a randomized voting rule. However, there are many other ways to obtain a randomized voting rule. The Gibbard-Satterthwaite Theorem above applies to deterministic rules only, so one might hope that randomized rules are not subject to such an impossibility. Unfortunately, as it turns out, there is a subsequent result by Gibbard that generalizes the Gibbard-Satterthwaite Theorem to randomized rules. To present this result, we first need to define strategyproofness in the context of randomized rules, and for that, we need to define preferences over lotteries over alternatives. For example, if a voter's preferences are $a \succ b \succ c$, should the voter prefer $b$, or a 50-50 lottery over $a$ and $c$? Both could be reasonable. For example, if the voter has utilities 3, 2, and 0 for the alternatives respectively, $b$ would give higher expected utility ($2 > 1.5$), but if the voter has utilities 3, 1, and 0, then the 50-50 lottery over $a$ and $c$ gives higher utility ($1.5 > 1$). Therefore, in this context, a quite conservative definition of strategyproofness is often used: a randomized rule is strategyproof if and only if for *every* utility function over the alternatives that is consistent with the voter's preferences over the (pure) alternatives, the voter maximizes her utility by reporting these true preferences (regardless of how the others vote).[7] We can now present Gibbard's result:

**Theorem 6.3 (Gibbard, 1977).** *If there are no restrictions on the preference domain, any strategyproof randomized rule is a randomization over a collection of the following types of rules:*

- unilateral rules*, under which at most one voter's vote affects the outcome;*
- duple rules*, under which there are at most two alternatives that have a possibility of winning (i.e., that win under some profile).*

The result makes it clear that randomization is not the answer to all our problems. A coin flip results in the discarding of all but one of the votes, or in the discarding of all

[6] Recall that an alternative $a$ is a *Condorcet winner* if it wins all its pairwise contests. That is, for every other alternative $b$, more voters prefer $a$ to $b$ than vice versa.

[7] For studies of other ways of extending strategyproofness to randomized voting rules, see Aziz et al. (2013d) and Aziz et al. (2014c).

but two of the alternatives. In many situations, these rules will not be acceptable. Still, the result allows for some randomized rules that are perhaps not *entirely* unreasonable. For example, we can randomly choose a dictator (the theorem implies that, with three or more alternatives, this is in fact the only way to guarantee a Pareto-optimal outcome), or randomly choose two alternatives and have a majority election between them. Barberà (1979) gives some characterizations of randomized strategyproof rules as well; these are consistent with Gibbard's result above, but seem to cast the rules in a more positive light. More recently, Procaccia (2010) studied the extent to which strategyproof randomized rules can achieve formal approximations to the scores from common voting rules.

A final possible avenue is to use *irresolute* rules, which return a *set* of alternatives (possibly larger than one) and leave it at that. Can such a rule be strategyproof (and simultaneously reasonable)? To make sense of this question, we first need to say something about what an agent's preferences over sets of alternatives can be. Building on earlier results, Brandt (2011b) and Brandt and Brill (2011) have recently provided results that show that various irresolute rules are in fact strategyproof with respect to various extensions of preferences to sets of alternatives.[8] While these positive results are encouraging, they do face a major limitation. In many voting settings, in the end, we require a single winning alternative. If we add any procedure for going from the winning set of alternatives to a single one—for example, choosing the lexicographically first alternative in the set—then the combination of the irresolute rule and the subsequent procedure is a resolute rule, and we run right back into the Gibbard-Satterthwaite impossibility result. Similarly, if we randomly choose from the winning set, we run into the impossibility results for randomized rules. Thus, for these positive results to apply, the procedure for going from the selected set of alternatives to a single alternative fundamentally needs to remain unspecified, and moreover the voters need to respond to this lack of information in a particular way. For more detail, see Chapter 3.

## 6.4 Computational Hardness as a Barrier to Manipulation

Another potential barrier to manipulation is computational hardness. Even if we cannot prevent a voting rule from being manipulable in principle, this may not be a significant concern as long as determining how to manipulate it is computationally prohibitive.

The argument that the complexity of computing a manipulation might be a barrier to strategic voting was first put forward in an influential paper by Bartholdi et al. (1989a). A whole subfield of social choice has since grown from this proposal, studying the computational complexity of manipulating different voting rules under several different assumptions (e.g., Conitzer et al., 2007). For two recent surveys, see Faliszewski et al. (2010) and Faliszewski and Procaccia (2010); Brandt et al. (2013a) also discuss the topic at some length. In the remainder of this section, we discuss this line of work in more detail.

[8] Other extensions lead to negative results (Duggan and Schwartz, 2000). For more on strategyproofness and other notions of monotonicity in this context, see Sanver and Zwicker (2012) and the references cited in that work.

### 6.4.1 The Basic Variant

The original paper (Bartholdi et al., 1989a) defined a basic model which has since been investigated extensively. We suppose all but one voter, the *manipulator*, have voted and that these votes and the rule to be used are known to the manipulator. We ask whether it is possible for the manipulator to ensure that a given alternative wins. More formally, we can define the following decision problem.

#### *Manipulation Problem*

**Given.** A profile of votes $\Pi$ cast by everyone but the manipulator, and a preferred alternative $a$.
**Question.** Is there a vote that the manipulator can cast so that $a$ wins?

This problem is typically in NP as a simple witness is a vote that ensures $a$ wins. Supposing that the voting rule is polynomial to execute,[9] this witness can be checked in polynomial time. There is also a *destructive* variant of this question, where we ask if it is possible for the manipulator to cast a vote so that a given alternative does *not* win. Note that these problems correspond exactly to the predicament of voter 3 in Example 6.1, with the exception that the question is now whether she can make a particular alternative win. One may wonder if a more natural problem would be to determine the *best* (according to her own true preferences) alternative that she can make win. This problem is effectively equivalent; to answer it, it is sufficient to evaluate for each of the alternatives in turn whether she can make it win (and, conversely, it is necessary to at least evaluate whether she can make her most-preferred alternative win).

Of course, when the rule is plurality, this problem is computationally trivial: to see if you can make alternative $a$ win, it suffices to see what would happen if you submitted a vote that ranks $a$ first. Indeed, for many rules, the problem is in P. Bartholdi et al. (1989a) provided an algorithm that solves the problem in polynomial time for many voting rules.

**Definition 6.1.** Say that a voting rule satisfies the *BTT conditions* if

1. it can be run in polynomial time,
2. for every profile $\Pi$ and every alternative $a$, the rule assigns a score $S(\Pi, a)$ to $a$,
3. for every profile $\Pi$, the alternative with the maximum score wins,[10] and
4. the following monotonicity condition holds: for any $\Pi$, $\Pi'$, for any alternative $a$, if for each voter $i$ we have that $\{b : a \succ_i b\} \subseteq \{b : a \succ'_i b\}$, then $S(\Pi, a) \leqslant S(\Pi', a)$. (That is, if we modify a vote in a way that does not rank anyone ahead of $a$ that was previously ranked behind $a$, then $a$'s score cannot have decreased.)

**Theorem 6.4 (Bartholdi et al., 1989a).** *The manipulation problem can be solved in polynomial time for any rule satisfying the BTT conditions.*

The algorithm for constructing a manipulator vote that successfully makes alternative $a$ win (if any such vote exists) is quite straightforward. Rank $a$ first. For the next

[9] See earlier chapters in the book for discussion of rules for which this is not the case.
[10] Assume, say, a fixed tie-breaking order.

position in the vote, find some remaining alternative $b$ that can be ranked there so that $a$ still wins. (To check this, complete the rest of the vote arbitrarily, and calculate $b$'s score; by the monotonicity condition above, $a$ and $b$'s scores will not depend on how the rest of the vote is completed. This is because if we change the relative ordering of the remaining alternatives, this is a modification that satisfies the condition, and so cannot decrease $a$ or $b$'s score; it can also not increase these scores, because then the reverse modification would decrease it.) If no such alternative can be found, declare failure; if the vote is completed, declare success; otherwise, repeat for the next position. This algorithm applies not only to positional scoring rules such as plurality and Borda, but also to rules such as Copeland and maximin.[11]

Bartholdi et al. (1989a) were also the first to show that the problem is NP-hard for some rules. Specifically, they showed NP-hardness for manipulating the second-order Copeland rule, under which an alternative's score is the sum of the Copeland scores of the alternatives that it defeats. (Note that this way of scoring violates the third condition above: if in some vote, we change the relative ordering of the alternatives ranked (say) behind $a$ only, this can affect those alternatives' Copeland scores, and thereby $a$'s second-order Copeland score.) They also showed NP-hardness of manipulation for the (first-order) Copeland rule when ties are broken by the second-order Copeland rule; we will say more about the importance of the tie-breaking procedure later in this chapter. Shortly after, Bartholdi and Orlin (1991) proved that the better-known STV rule is NP-hard to manipulate in this sense. The problem has been shown to be NP-hard for several other rules more recently, including ranked pairs (Xia et al., 2009), and Nanson and Baldwin's rules (Narodytska et al., 2011). The ranked pairs rule orders the pairwise outcomes by the size of the victory. It then constructs a total ordering over alternatives by taking these pairs in order and fixing the order unless this contradicts previous decisions. The top of the order constructed in this way is the overall winner. Nanson and Baldwin's rules are elimination versions of Borda voting. Nanson's rule repeatedly eliminates all alternatives with less than the average Borda score. Baldwin's rule, on the other hand, successively eliminates the alternative with the lowest Borda score. Table 6.1 gives a representative sample of complexity results for this manipulation problem, as well as for some related manipulation problems discussed in the next subsections.

### 6.4.2 Coalitions of Manipulators

So far, we have considered the computational complexity of just *one* voter trying to manipulate the election. In practice, multiple voters may collude to manipulate the result. Indeed, it is often the case that we need a coalition of manipulators to be able to change the result.

[11] Recall that the Borda rule gives an alternative $m-1$ points each time it is ranked first, $m-2$ points each time it is ranked second, ..., and 0 points each time it is ranked last. More generally, a positional scoring rule associates a score with each rank, and the alternative with the highest score wins. Under the Copeland rule, an alternative $a$ gets a point for each other alternative $b$ such that more votes rank $a$ ahead of $b$ than vice versa (and some fraction of a point if the number of votes ranking $a$ ahead of $b$ is the same as vice versa). Finally, under the maximin rule, we find, for each alternative $a$, the alternative $b$ that minimizes the number of votes that rank $a$ ahead of $b$ (the worst pairwise outcome for $a$); this number is $a$'s score, and the alternative with the maximum score wins.

**Table 6.1.** *Computational complexity of deciding the manipulation problem with a small number of voters (unweighted votes) or a coalition of voters (weighted votes), for various voting rules*

| | unweighted votes | | weighted votes | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | constructive manipulation | | constructive | | | | destructive | | |
| # alternatives | | | 2 | 3 | 4 | ⩾5 | 2 | 3 | ⩾4 |
| # manipulators | 1 | ⩾ 2 | | | | | | | |
| plurality | P | P | P | P | P | P | P | P | P |
| plurality with runoff | P | P | P | NP-c | NP-c | NP-c | P | NP-c | NP-c |
| veto | P | P | P | NP-c | NP-c | NP-c | P | P | P |
| cup | P | P | P | P | P | P | P | P | P |
| Copeland | P | P | P | P | NP-c | NP-c | P | P | P |
| Borda | P | NP-c | P | NP-c | NP-c | NP-c | P | P | P |
| Nanson | NP-c | NP-c | P | P | NP-c | NP-c | P | P | NP-c |
| Baldwin | NP-c | NP-c | P | NP-c | NP-c | NP-c | P | NP-c | NP-c |
| Black | P | NP-c | P | NP-c | NP-c | NP-c | P | P | P |
| STV | NP-c | NP-c | P | NP-c | NP-c | NP-c | P | NP-c | NP-c |
| maximin | P | NP-c | P | P | NP-c | NP-c | P | P | P |
| Bucklin | P | P | P | NP-c | NP-c | NP-c | P | P | P |
| fallback | P | P | P | P | P | P | P | P | P |
| ranked pairs | NP-c | NP-c | P | P | P | NP-c | P | P | ? |
| Schulze | P | P | P | P | P | P | P | P | P |

*Note:* P means that the problem is polynomial, NP-c that the problem is NP-complete. For example, constructive manipulation of the veto rule is polynomial for unweighted votes or for weighted votes with a coalition of 2 manipulators, but NP-hard for 3 or more manipulators. On the other hand, destructive manipulation of the veto rule is polynomial for weighted votes with a coalition of 2 or more manipulators. We consider the variant of Copeland where an alternative gets 1 point if it defeats an opponent, 0.5 points for a draw, and 0 if it loses. "?" indicates that the computational complexity is open at the time of writing this chapter. For references, see: Faliszewski et al. (2008) and Conitzer et al. (2007) for Copeland; Davies et al. (2011), Conitzer et al. (2007), and Betzler et al. (2011) for Borda; Narodytska et al. (2011) and Davies et al. (2014) for Nanson and Baldwin; Narodytska and Walsh (2013) for Black; Xia et al. (2009) for maximin; Xia et al. (2009) and Faliszewski et al. (2014) for Bucklin; Faliszewski et al. (2014) for fallback; Xia et al. (2009) and Hemaspaandra et al. (2014c) for ranked pairs; Parkes and Xia (2012) and Gaspers et al. (2013) for Schulze; and Conitzer et al. (2007) for other results or references to them.

### *Coalitional Manipulation Problem*

**Given.** A profile of votes $\Pi$ cast by everyone but the manipulators, a number of manipulators, and a preferred alternative $a$.
**Question.** Is there a way for the manipulators to cast their votes so that $a$ wins?

Again, it can be debated if this should be called "manipulation" because the manipulators might not have to vote strategically to ensure their preferred alternative wins. However, as has become common in the literature, we will refer to this problem as coalitional manipulation. Coordinating even a small coalition of voters introduces fresh computational challenges. For example, with the Borda rule, a simple greedy procedure will compute an optimal strategic vote for one voter, but it is NP-hard to compute how two voters together can manipulate the result (Davies et al., 2011; Betzler et al., 2011). Similar results hold for Copeland voting (the first rule for which it was shown that the problem is easy with one manipulator but hard with two) (Faliszewski et al., 2008), other scoring rules (Xia et al., 2010b), maximin (Xia et al., 2009), and Black's rule

(Narodytska and Walsh, 2013). Intriguingly, in all these cases, it requires only two manipulators to make manipulation hard. Black's rule is the voting rule that elects the Condorcet winner if it exists, and otherwise the Borda winner.

One criticism that can be made about the complexity results considered so far is that they require the number of alternatives to grow in an unbounded fashion. If the number of alternatives is held constant, then a single manipulator would have only a constant number ($m!$) of votes to consider. Even for a coalition of $n'$ manipulators, if the rule is anonymous, then the total number of joint votes for the coalition is the number of ways $n'$ indistinguishable balls (voters) can be placed into $m!$ urns (possible votes), which is $\binom{n'+m!-1}{m!-1}$, which is polynomial in $n'$. Hence, as long as there is a polynomial-time algorithm for executing the rule, a manipulation (if one exists) can be computed in polynomial time when the number of alternatives is constant. However, this argument fundamentally relies on the voters being indistinguishable, which is not the case when voters have weights.

### 6.4.3 Weighted Votes

Weighted votes occur in a number of real-world settings (e.g., shareholder elections and various parliaments). Weights are typically integers and a vote of weight $k$ can be seen as $k$ identical and unweighted votes. It turns out that with weighted votes, we encounter complexity in manipulation problems even with a small number of alternatives. We consider the following decision problem for weighted votes.

#### *Coalitional Weighted Manipulation Problem*

**Given.** A profile of weighted votes $\Pi$ cast by everyone but a coalition of manipulators, a weight for each of the manipulators, and a preferred alternative $a$.
**Question.** Is there a way for the manipulators to cast their votes so that $a$ wins?

There is again a destructive variant of this problem where the coalition wants a given alternative not to win.

With two alternatives, most common voting rules degenerate to majority voting. In addition, by May's Theorem, this is the only voting rule over two alternatives that is anonymous, neutral, and positively responsive. With majority voting, the manipulators' best action even when their votes are weighted is always to vote for the alternative that they wish to win. With three or more alternatives, however, computing a manipulation can be computationally hard, provided we have a coalition of manipulators (whose size is allowed to increase) and votes that are weighted. For example, computing how to manipulate the veto (aka. antiplurality) rule[12] is polynomial with unweighted votes but NP-complete with weighted votes and just 3 alternatives (Conitzer et al., 2007). Some intuition for this result is as follows. The manipulators could find themselves in the situation where, after counting the nonmanipulators' votes, two alternatives ($b$ and $c$)

[12] Recall that under the veto rule, the winner is the alternative that is ranked last in the fewest votes. Equivalently, it is the positional scoring rule in which the bottom rank receives 0 points and all other ranks receive 1 point.

are tied for the lead (i.e., they have been vetoed the least), but the third alternative ($a$) is the one that the manipulators want to win. Clearly the manipulators do not want to veto $a$. To make $a$ win, however, they may need to divide their total veto weight very evenly between $b$ and $c$, so that $a$ comes out just barely ahead of each of them. Thus, the manipulators face the problem of partitioning a set of integers (their weights) into two subsets (vetoing $b$ or vetoing $c$) so that each subset has the same weight—and this is an NP-complete problem. This intuition can be turned into a formal NP-hardness reduction as follows.

**Theorem 6.5.** *The coalitional weighted manipulation problem is NP-complete under the veto rule, even with only three alternatives.*

*Proof.* The problem is in NP because a profile of votes for the manipulators will serve as a certificate (because the veto rule is computationally easy to execute). To prove NP-hardness, we reduce from the PARTITION problem, in which we are given a set of integers $w_1, \dots, w_{n'}$ with $\sum_{i=1}^{n'} w_i = W$ (where $W$ is even) and are asked whether there exists a subset $S \subseteq \{1, \dots, n'\}$ such that $\sum_{i \in S} w_i = W/2$. We reduce this problem to the coalitional weighted manipulation problem under the veto rule with three alternatives, as follows. Let $a$, $b$, and $c$ be the alternatives, where $a$ is the alternative that the manipulators would like to win. Create one nonmanipulator vote with weight $W - 1$ that ranks $a$ last. Furthermore, for each $i \in \{1, \dots, n'\}$, create a manipulator (the $i$th manipulator) with weight $2w_i$.

We now show that the manipulators can succeed in this instance if and only if the original partition instance has a solution. If the partition instance has a solution $S$, then let the manipulators in $S$ rank $b$ last, and let the ones outside $S$ rank $c$ last. Then, $a$ wins, appearing in last place only for $W - 1$ of the weight, whereas $b$ and $c$ each appear in last place for $\sum_{i \in S} 2w_i = 2W/2 = W$ of the weight.

Now suppose that the partition instance has no solution. This implies that for each subset $S \subseteq \{1, \dots, n'\}$, either $\sum_{i \in S} w_i \leqslant W/2 - 1$ or $\sum_{i \notin S} w_i \leqslant W/2 - 1$ (due to the integrality of the $w_i$ and $W/2$). Then, for any profile of votes for the manipulators, let $S$ be the set of manipulators that rank $b$ last. Then, we have either $\sum_{i \in S} 2w_i \leqslant W - 2 < W - 1$, so that $b$ ranks ahead of $a$, or $\sum_{i \notin S} 2w_i \leqslant W - 2 < W - 1$, so that $c$ ranks ahead of $a$. So the manipulators cannot make $a$ win. □

Note that the reduction is set up in such a way that $a$ cannot end up tied for the win, so it does not matter how ties are handled. On the other hand, note that this is only a weak NP-hardness result because the reduction is from PARTITION. Indeed, we can compute a manipulation for a coalition of voters using dynamic programming in pseudopolynomial time—that is, in polynomial time when the weights are represented in unary (or equivalently, when the weights are small). Similar (though often more involved) reductions can be given for many other rules. In fact, a dichotomy result holds for positional scoring rules in general: every scoring rule that is not isomorphic to plurality is NP-hard to manipulate with three or more alternatives and weighted votes (Hemaspaandra and Hemaspaandra, 2007; Procaccia and Rosenschein, 2007b; Conitzer et al., 2007).

### 6.4.4 Tie-Breaking

For complexity-of-manipulation results like these, it is important to specify precisely how ties are broken. This perhaps should not be surprising, because a single manipulator can only change the result if the election is close to being tied. A common assumption is that we break ties in favor of the manipulator. That is, we suppose that the preferred alternative wins if it is among the set of co-winners. This is usually justified on the grounds that if ties are broken, say, at random, then this corresponds to increasing the probability that the given alternative wins. However, the choice of the tie-breaking procedure is not a minor detail. It can actually change the computational complexity of computing a manipulation. We can get different results if we break ties against the manipulator (that is, we suppose that the manipulator's preferred alternative wins only if it is the unique winner).

The importance of tie-breaking can be seen in the earliest literature on computational social choice. Recall that Bartholdi et al. (1989a) proved that a single agent can manipulate the result of a Copeland election (with "straightforward" tie-breaking schemes) in polynomial time using their greedy algorithm, but when the second-order tie-breaking rule is added manipulation becomes NP-hard.

Faliszewski et al. (2008) proved that for Copeland voting, changing the way that pairwise ties (two alternatives that are each ranked above the other equally often) are handled can change the computational complexity of manipulation. For example, with weighted votes and three alternatives, if ties result in a score of 0, then it is NP-hard for a coalition to compute a manipulation that makes a given alternative the unique winner of the election, but this problem becomes solvable in polynomial time if ties are given any other score. (Note that this is an "internal" form of tie-breaking, rather than tie-breaking between multiple winners at the end of applying an irresolute rule.) Also, if instead the manipulators seek to make that alternative just *one* of the winners, then the problem is solvable in polynomial time when a tie results in a score of 1, but NP-hard if ties are given any other score.

To study tie-breaking at random in more detail, Obraztsova et al. (2011) set up a model where the manipulators have utilities over the alternatives and the goal is to increase the expected utility of the result. *All* scoring rules, as well as Bucklin and plurality with runoff, can be manipulated in polynomial time in such a situation. On the other hand, Copeland, maximin, STV and ranked pairs are NP-hard to manipulate in this case (Obraztsova and Elkind, 2011).

Another method to deal with ties is to select a vote at random and select the highest-ranked of the tied alternatives from this vote (Tideman, 1987).[13] Aziz et al. (2013f) show that, in general, there is no connection between the complexity of computing a manipulating vote when tie-breaking with a random alternative or with a random vote. However, for common rules like $k$-approval, Borda, and Bucklin, the computational complexity increases from polynomial to NP-hard when tie-breaking with a random vote rather than at random among the co-winners. For other rules like plurality, veto, and plurality with runoff, it remains possible to compute a manipulating vote in polynomial

[13] For more on tiebreaking schemes in computational social choice, see Freeman et al. (2015).

time. Finally, for rules like STV, computing a manipulation is NP-hard irrespective of the tie-breaking method as it is possible to prove NP-hardness with a class of elections in which there are never any ties.

### 6.4.5 Incomplete Information

So far, we have assumed that the manipulator has complete knowledge of the other votes. This is a strong assumption that, extreme circumstances aside, is at best a rough approximation of the truth. It is often defended on the grounds that if it is NP-hard to compute a manipulation with complete information then it must remain so when we have probabilistic information about the nonmanipulators' votes (Conitzer et al., 2007). There has, however, been some work relaxing this assumption. For example, Conitzer et al. (2011a) consider the complexity of computing manipulations given only partial information about the nonmanipulators' votes. Given such partial information, they consider whether the manipulator has a *dominating* nontruthful vote that makes the winner always at least as preferable as, and sometimes more preferable than, the alternative that would win if the manipulator voted sincerely. This was further studied by Reijngoud and Endriss (2012).

### 6.4.6 Building in Hardness

Once we accept hardness of manipulation as a desirable property of voting rules, it becomes an interesting question whether we can engineer voting rules to be more computationally complex to manipulate. One general construction is to "hybridize" together two or more existing voting rules. For example, we might add one elimination pre-round to the election, in which alternatives are paired off and only the one preferred by more voters goes through (Conitzer and Sandholm, 2003). This generates a new voting rule that is often computationally hard to manipulate. In fact, the problem of computing a manipulation can now move to complexity classes higher than NP depending on when the schedule of the pre-round is announced. Such hybrid voting rules also inherit some (but not all) of the properties of the voting rules from which they are constructed. For example, if the initial rule is Condorcet consistent, then adding a pre-round preserves Condorcet consistency.

Other types of voting rules can be hybridized together. For example, we can construct a hybrid of the Borda and Copeland rules in which we run two rounds of Borda, eliminating the lowest-scoring alternative each time, and then apply the Copeland rule to the remaining alternatives. Such hybrids are often resistant to manipulation. For example, many hybrids of STV and of Borda are NP-hard to manipulate (Elkind and Lipmaa, 2005). More generally, voting rules that have multiple stages successively eliminating alternatives tend to be more computationally difficult to manipulate than one-stage rules (Coleman and Teague, 2007; Narodytska et al., 2011; Davies et al., 2012; Walsh and Xia, 2012).

Another way to combine together two or more voting rules is to use some aspect of the particular election (the votes, or the names of the alternatives) to pick which voting rule is used to compute the winner. For example, suppose we have a list of $k$ different voting rules. If all the alternatives' names (viewed as natural numbers) are congruent, modulo $k$, to $i$ then we use the $i$th voting rule, otherwise we use the default last rule. Such a form of hybridization gives elections which are often computationally difficult to manipulate (Hemaspaandra et al., 2009). Another possibility is to just leave it ambiguous which of the voting rules will be used; Elkind and Erdélyi (2012) have studied how hard it is for the manipulators to select their votes so that they succeed for any of a given set of rules. Finally, another possibility is that we have a runoff between the winners of two voting rules. This also often makes manipulations more difficult to compute (Narodytska et al., 2012).

## 6.5 Can Manipulation Be Hard Most of the Time?

NP-hardness is a worst-case notion. For NP-hard manipulation problems, supposing $P \neq NP$, any manipulation algorithm will face *some* families of instances on which it does not scale polynomially. But it is not at all clear that these are the instances that manipulators would need to solve in practice. They may be pathological. Hence, it is possible that these NP-hardness results lull us into a false sense of security regarding the manipulability of our voting rules. A much better type of result would be that the manipulation problem is *usually* hard. Is such a result feasible, and what exactly does "usually" mean here? To investigate this, it is helpful to first consider some actual manipulation algorithms for voting rules that are NP-hard to manipulate.

### 6.5.1 Some Algorithms for NP-Hard Manipulation Problems

Assuming $P \neq NP$, a manipulation algorithm for a voting rule that is NP-hard to manipulate can only hope to either (1) succeed on all instances and require more than polynomial time in the worst case, but still scale "reasonably," particularly on "typical" instances; or (2) run in polynomial time and succeed on many, but not all, instances.

For instance, under the STV rule, Coleman and Teague (2007) give a simple enumerative method for a coalition of $k$ unweighted voters to compute a manipulation, which runs in $O(m!(n+mk))$ time (where $n$ is the number of voters voting and $m$ is the number of alternatives). For a single manipulator, Conitzer et al. (2007) give an $O(n1.62^m)$ time recursive algorithm to compute the set of alternatives that can win an STV election.

Such algorithms have been shown to perform well in practice. For example, Coleman and Teague (2007) showed experimentally that only a small coalition is needed to change the elimination order of the STV rule in many cases. As a second example, Walsh (2010) showed that the Conitzer et al. (2007) algorithm could often quickly compute manipulations of the STV rule even with hundreds of alternatives. Walsh (2009, 2011b) also empirically studied the computational cost of manipulating the veto rule by a coalition of weighted voters. Except in rather artificial and "hung" elections, it was easy to find manipulations or prove that none exist.

An algorithm designed for the manipulation of one specific rule, however effective it may be, may just be exploiting an idiosyncratic property of that particular rule. It may well be the case that other desirable rules do not have this property and are, in fact, "usually" hard to manipulate. One approach to addressing this criticism is to design manipulation algorithms that are not specific to one voting rule. Such algorithms, to the extent that they avoid exhaustive search, are heuristic in nature and do not always succeed. This category of algorithms includes some of the earliest work providing technical results that cast doubt on whether worst-case hardness of manipulation has any significant implications for the "typical" case. Procaccia and Rosenschein (2007b) provide a greedy algorithm for rules based on a score, in which the manipulators create their votes in sequence, at each point ranking their preferred alternative first and the remaining alternatives in increasing order of their current score. Conitzer and Sandholm (2006) provide an algorithm that attempts to find two possible winners, by first choosing an arbitrary vote profile for the manipulators to find one possible winner $a_1$, and then, for every remaining alternative $a$, choosing a vote profile for the manipulators where everyone ranks $a$ first and $a_1$ last. It is argued (and supported by simulations) that usually, if the manipulators are pivotal (have a possibility of changing the outcome of the election) at all, then they can only make two alternatives win. For instances where this is so, and where the voting rule satisfies a weak monotonicity property, the algorithm can be proved to find all alternatives that the manipulators can make win.

All these empirical results suggest that we need to treat results about the NP-hardness of manipulation with some care. Voters may still be able to compute a manipulation successfully using rather simple and direct methods. The theoretically inclined reader, however, may feel dissatisfied with these types of results. Beyond getting intuition from simulations, can we actually *prove* that voting rules remain vulnerable to manipulation in the typical case? In what follows we discuss some of the approaches that researchers have taken to answer this question in the affirmative.

### 6.5.2 Approximation Methods

For almost all voting rules, we can easily make any alternative win provided we have enough manipulators; the hardness results are merely due to a limited supply of manipulators. With this in mind, we can consider manipulation as an optimization problem, where we try to minimize the number of manipulators required to achieve a given outcome. One option is to use approximation methods to tackle such optimization problems.[14]

For example, Zuckerman et al. (2009) consider a variant of the algorithm by Procaccia and Rosenschein (2007b) (presented above) to compute manipulations of the Borda rule. Again, the algorithm constructs the vote of each manipulator in turn. The alternative that the manipulators wish to win is put in first place, and the remaining alternatives are placed in the manipulator's vote in increasing order of their current Borda scores. The method continues constructing manipulating votes until the desired alternative wins. A rather long and intricate argument shows that this method requires at most one additional manipulator relative to the optimal solution. Based on a connection to a scheduling problem, Xia et al. (2010b) provide an algorithm that works

[14] Another notion of approximation in manipulation problems is to approximately maximize an alternative's increase in score, given a fixed set of manipulators (Brelsford et al., 2008). Theorem 4 in that paper relates that notion of approximability to the one discussed here.

for all positional scoring rules, though it may require as many as $m - 2$ additional manipulators.

### 6.5.3 Frequency of Manipulability

Again, whether the manipulators can achieve the result they want depends in large part on their number. We may then wonder whether, given an instance of the coalitional manipulation problem, we can quickly eyeball whether the manipulators are likely to be successful, purely based on the size of their coalition relative to the size of the electorate. It turns out that this is indeed the case. Building on earlier work by Procaccia and Rosenschein (2007a),[15] Xia and Conitzer (2008a) showed that for an extremely large class[16] of voting rules called *generalized scoring rules*, under some assumptions on the distribution of votes, if the number of manipulators is $O(n^p)$ for $p < 1/2$, the probability that a random profile is manipulable goes to zero; whereas if it is $\Omega(n^p)$ for $p > 1/2$, it goes to one. This leaves the knife-edge case of $p = 1/2$, which has been studied both experimentally (Walsh, 2009) and analytically (Mossel et al., 2013).

Another line of research along these lines proves *quantitative* versions of the Gibbard-Satterthwaite impossibility result. Here, the idea is not to be satisfied with a statement that says that somewhere in the space of all possible profiles, there exists a manipulable one; rather, these results state that, under Gibbard-Satterthwaite-like conditions, a randomly chosen profile has a significant probability of being manipulable. After a sequence of earlier partial results along this line (Friedgut et al., 2008; Dobzinski and Procaccia, 2008; Xia and Conitzer, 2008b; Isaksson et al., 2012), Mossel and Rácz (2012) seem to have achieved the gold standard. They prove that under a voting rule with 3 or more alternatives that is $\epsilon$-far away from the set of nonmanipulable rules,[17] a randomly chosen profile has a probability of being manipulable that is at least inverse polynomial in $n$, $m$, and $1/\epsilon$.

### 6.5.4 Restricted Preferences

Finally, it is important to realize that it is unrealistic to assume that profiles of votes are drawn uniformly at random; generally, the voters' preferences over the alternatives are quite structured. For example, the profile may be single-peaked. How does this affect the complexity of the manipulation problem? Several papers have addressed this question, showing that this restriction often, but not always, makes the manipulation problem easier (Walsh, 2007; Faliszewski et al., 2009e; Brandt et al., 2010a). While it may seem odd in this context to focus on single-peaked preferences—for which, after all, a desirable strategyproof voting rule is available in the form of the median voter rule[18] —these results nevertheless provide important insight into how restricting the space of profiles can cause complexity barriers to manipulation to fall apart.

[15] See also Slinko (2004) and Pritchard and Wilson (2009).

[16] Xia and Conitzer (2009) characterize this class as those rules that are anonymous and *finitely locally consistent*.

[17] Here, the distance between two rules is the fraction of inputs on which they differ.

[18] Moreover, under some assumptions on strategic behavior by the voters and/or candidates, even rules such as plurality and STV end up returning the same winner as the median voter rule when preferences are single-peaked (Brill and Conitzer, 2015).

## 6.6 Fully Game-Theoretic Models

The computational problems studied in this chapter so far all make some major simplifying assumptions. In most cases it is assumed that the votes of the other voters are known exactly; even when this is not assumed, the other voters are not modeled as strategic agents. If we do model them this way, this leads us into fully game-theoretic models, and indeed these have received some attention in the computational social choice community.

To make sense of this, a first issue that needs to be addressed is the staggering multiplicity of equilibria in most voting scenarios.[19] Often, most profiles will not allow any single individual to change the outcome, and all of these profiles are Nash equilibria as an immediate consequence. Many of these profiles will have voters vote in ways that make no sense with respect to their true preferences. Based on this observation, we may be able to rule out many of these equilibria—for example, we might require voters not to play weakly dominated strategies.[20] However, other issues are more difficult to address. For example, in a plurality election, any two of the alternatives might be cast in a "front-runner" role, resulting in an equilibrium where everyone votes for one of these two, because to do otherwise would be to waste one's vote. This also illustrates that there will be many alternatives that win in *some* equilibrium.[21]

As it turns out, these issues are avoided when the voters, instead of voting simultaneously, vote in sequence, so that each voter has full knowledge of all the previous votes. If we additionally assume that all the preferences are common knowledge (as well as the order in which the voters vote, and the voting rule used), and all preferences are strict, then there is a unique alternative that wins in subgame-perfect Nash equilibrium.[22] This can be proved by induction on the number of voters, roughly as follows. Suppose it is true for $n - 1$ voters. Then, in the case of $n$ voters, consider the first voter. For every vote that she might cast, she can, by the induction assumption, determine the alternative that will win in equilibrium from that point on. From all these options, she will then choose the one that ranks highest in her own preferences. (There may be multiple votes for the first voter that achieve this, so the equilibrium *votes* are not unique.) This raises several interesting questions. First of all, will this result in good outcomes? Of course, it is tricky to give a general definition of what "good" means in this context. As it turns out, though, for many rules, there exist profiles of preferences that, in equilibrium, result in outcomes that are quite unambiguously bad. Specifically, Xia and Conitzer (2010c) show that this is the case for rules with a low *domination index*, which indicates how many more than half of the voters are needed

[19] Recall that, given the voters' true preferences, a *Nash equilibrium* consists of a profile of votes such that no individual voter can obtain an outcome she prefers to the current one by unilaterally changing her vote.

[20] Recall that one strategy *weakly dominates* another if the former always delivers at least as good a result for the agent, and in some cases a strictly better one.

[21] A recent article investigates what game structures can emerge when multiple voters are considering strategically changing their votes (Elkind et al., 2014b).

[22] Recall that in a *subgame-perfect* Nash equilibrium, the strategies constitute a Nash equilibrium in every subgame. In our case, when a subset of the voters has cast specific votes, the remainder of the voting game constitutes a subgame.

to force the outcome.[23] Some counterintuitive examples for the plurality rule are also given by Desmedt and Elkind (2010).

Another question is whether these equilibria can be efficiently computed. A natural approach is to use a dynamic programming algorithm corresponding to the backward induction process in game theory, as follows. First compute what the last voter would do for every situation in which she might be placed; then compute what the second-to-last voter would do for every situation in which she might be placed (which is now possible because it is known at this point how the last voter would respond to any vote that the second-to-last voter might cast); and so on. This algorithm is correct, but its runtime depends on the number of possible "situations." What is a "situation," anyway? One might interpret this as the entire partial profile of votes cast so far (i.e., the node in the extensive form of the game), but this will scale very poorly. It is also overkill: for example, for a positional scoring rule, all that is needed is the total scores of the alternatives so far, not the precise votes that led to this score. More generally, the amount of information necessary to summarize the votes of a subelectorate is known as the *compilation complexity* of a voting rule (Chevaleyre et al., 2009; Xia and Conitzer, 2010b). Xia and Conitzer (2010c) exploit the connection to this concept to obtain algorithms for solving the game that, while still exponential, scale much better than the naïve approach. (Desmedt and Elkind (2010) give a similar algorithm for plurality.) Intriguingly, from simulations performed by Xia and Conitzer (2010c), the game-theoretic outcomes on random profiles do not look as bad as the worst-case results above might suggest. The exact complexity of the computational problem is not known; it may be PSPACE-complete.

Still, is there nothing substantial that we can say about the equilibria of voting games in which voters vote simultaneously? In fact, we can, if we are willing to make some further assumptions about voters' preferences in voting. One natural assumption is that voters are *truth-biased* (Meir et al., 2010). This can be interpreted as follows: voters derive most of their utility from the outcome of the election, but they also derive a small amount of utility from voting truthfully. Hence, if it makes no difference to the outcome, voters slightly prefer to tell the truth. Thompson et al. (2013) show experimentally that for the plurality rule this dramatically reduces the set of equilibria. (They also study Bayes-Nash equilibria of games in which voters are not sure about each other's preferences.) Obraztsova et al. (2013) study this model from a theoretical perspective, again under the plurality rule.[24] Another direction is to substitute the slight preference for voting truthfully with a slight preference for *abstaining* (Desmedt and Elkind, 2010). Yet another direction is to add *dynamics* where voters start at some initial profile and iteratively update their vote to make themselves better off, until this process converges (Meir et al., 2010; Lev and Rosenschein, 2012; Reyhani and Wilson, 2012; Rabinovich et al., 2014).

[23] A similar negative result is given by Xia et al. (2011a) in a different context, where multiple related binary decisions must be made and these issues are voted on in sequence (but with all the voters voting at the same time on each issue). For more on voting in such combinatorial domains, please see Chapter 9.

[24] They also consider *strong* Nash equilibria, in which no *subset* of the agents can deviate in a way that makes them all better off, and draw a connection to Condorcet winners. More about the relationship between strong equilibrium and Condorcet winners can be found in papers by Sertel and Sanver (2004), Messner and Polborn (2007), and Brill and Conitzer (2015).

The above approaches all rely on *noncooperative* game theory. However, as we have already seen, it is natural to think about *coalitions* of voters coordinating their actions. Doing so in a game-theoretic framework is tricky, because the voters in a coalition may not all have the same preferences. This leads us to *cooperative* (or *coalitional*) game theory. A common solution concept there is that of the *core*, which is the set of all outcomes such that no coalition of agents could break off in a way that would make all of its members happier. In the context of elections, when a group of agents deviates, how happy this makes them depends on how the agents outside of the coalition end up voting. For example, will the agents outside the coalition be able to react to the votes of the coalition, or vice versa? These modeling choices correspond to the notions of the $\alpha$-core and the $\beta$-core. The computational complexity of these concepts in the context of elections is studied by Zuckerman et al. (2011). Bachrach et al. (2011) study the complexity of problems in cooperative game theory models of manipulation where payments are possible.

### 6.6.1 Other Topics

So far, we have supposed that the manipulating coalition can communicate and coordinate perfectly. In practice, this may be optimistic. For example, if the coalition is large, then it may be difficult for the coalition to communicate, as well as to ensure everyone votes appropriately. To address this, Slinko and White (2008) propose a more restricted model of strategic voting in which a single coalition member broadcasts a strategic vote and every member of the coalition either casts this vote or votes sincerely. In such a situation, a *safe strategic vote* is a broadcast vote that never results in an undesirable outcome, however many or few of the coalition follow it. The Gibbard-Satterthwaite Theorem extends to this notion of manipulation. Polynomial-time algorithms for computing a safe strategic vote have been given for $k$-approval, Bucklin, and Borda (Hazon and Elkind, 2010; Ianovski et al., 2011).

Another type of manipulation is for a single agent to vote more than once. This is often a concern in elections run in highly anonymous environments, such as Internet voting. A rule is said to be *false-name-proof* (Yokoo et al., 2004) if there is never an incentive for a voter to cast more than one vote. Conitzer (2008) gives a characterization of false-name-proof rules similar in spirit to the characterization of strategyproof rules by Gibbard (1977) that, perhaps unsurprisingly, is even more negative. Unlike in the case of strategyproofness, under the constraint of false-name-proofness, even the restriction of single-peaked preferences does not allow very appealing rules (Todo et al., 2011).

## 6.7 Conclusions

Besides being of interest in their own right, the computational manipulation problems discussed in this chapter are also important because of their implications for other, closely related problems in computational social choice. For example, the constructive manipulation problem is a special case of the *possible winner problem*, which asks, given a profile of partial votes and a given alternative, whether it is possible to complete the profile in such a way that that alternative wins. Similarly, the destructive manipulation problem is a special case of the *necessary winner problem*. For detailed analysis of the complexity of these problems, see, for example, Konczak and Lang (2005), Walsh (2007), Betzler and Dorn (2010), Xia and Conitzer (2011a), and Baumeister and Rothe (2012). The necessary winner problem, in turn, is important in settings in which we incrementally *elicit* voters' rankings rather than collecting them all at once. In this problem, we would like to be able to compute when we have elicited enough information to announce the winner (Conitzer and Sandholm, 2002). For further discussion of all of this, see also Chapter 10. There are also relations to *control* and *bribery* problems, which will be discussed in Chapter 7.

## Acknowledgments

We thank Felix Brandt, Piotr Faliszewski, Jérôme Lang, Ariel Procaccia, Arkadii
Text
Slinko, and Bill Zwicker for very detailed and helpful comments on earlier versions of this chapter. Conitzer thanks NSF and ARO for support under grants CCF-1101659, IIS-0953756, CCF-1337215, W911NF-12-1-0550, and W911NF-11-1-0332. Walsh is supported by the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program, AOARD grant FA2386-12-1-4056, and the Federal Ministry for Education and Research through the Alexander von Humboldt Foundation. Walsh is now funded by the European Research Council (ERC) under the EU's Horizon 2020 research and innovation programme via grant AMPLIFY 670077.