Chapter 8

Mechanism Design for Bounded Agents

> *Any fool can tell the truth, but it requires a man of some sense to know how to lie well.*
>
> **Samuel Butler**

Mechanism design has traditionally taken the conservative view that agents will always choose the actions that are in their own best interest—the assumption of *perfect rationality*. Specifically, the revelation principle discussed in the previous chapter relies heavily on this assumption. However, this may be an overly conservative assumption in that agents may not always have the computational resources to find the action that is in their own best interest—their rationality is *bounded*. For instance, bidding optimally in a reverse auction for trucking tasks may require the bidder to solve multiple NP-complete vehicle routing problems [Sandholm and Lesser, 1997]. With this in mind, we can ask questions such as:

- Can *impossibility results* in mechanism design that rely on the assumption of perfect rationality be circumvented if agents have limited computational resources?
- Given that the revelation principle ceases to apply when agents' rationality is bounded, are there benefits to using *non-truthful* mechanisms (even when reasonable truthful mechanisms exist)?[1]

It is impossible to answer these questions without some characterization of how the agents' rationality is bounded. Previous work by Larson and Sandholm relies on explicitly modeling the agents' computational choices to derive direct tradeoffs between the cost of additional computation and the benefits of additional computation to the solution [Larson and Sandholm, 2001a, 2005]. In contrast,

[1] Various research has proposed the use of mechanisms that are only *approximately truthful*. These can be easier to execute [Kothari *et al.*, 2003; Archer *et al.*, 2003], or their use can be motivated by impossiblity results that apply to truthful mechanisms [Parkes *et al.*, 2001; Goldberg and Hartline, 2003]. For approximately truthful mechanisms, the idea is not that it will necessarily be computationally difficult for agents to act optimally, but rather that the incentives for the agents to act optimally (rather than simply tell the truth) are somehow too small for the agents to respond to them. However, if the agents do respond to these slight incentives, the desirable properties of the mechanism may unravel completely.

this chapter will not require specific models of how the agents do their computation. Rather, it relies on classic complexity-theoretic notions to determine whether it is hard for the agents to find strategically optimal actions.

The rest of this chapter is layed out as follows. In Section 8.1, we show that there are settings where using the optimal truthful mechanism requires the center to solve a hard computational problem; but there is another, non-truthful mechanism, under which the center does not have to solve any hard optimization problem, but the problem of finding a beneficial manipulation is hard for one of the agents. Moreover, if the agent manages to find the manipulation, the produced outcome is the same as that of the best truthful mechanism; and if the agent does not manage to find it, the produced outcome is strictly *better* [Conitzer and Sandholm, 2004c]. In Section 8.2, we show that adding a *preround* to voting rules can make manipulation of these voting rules computationally much harder [Conitzer and Sandholm, 2003g]. However, those hardness results (as well as others) rely on the number of candidates being unbounded. In Section 8.3, we show that if we consider *coalitional* manipulation by *weighted* voters, then we can get hardness even with constant numbers of candidates [Conitzer and Sandholm, 2002a; Conitzer *et al.*, 2003]. Unfortunately, all of the above hardness results only prove hardness in the worst case (as is common in complexity theory). In Section 8.4, we give an impossibility result that makes it appear unlikely that voting rules can be constructed that are *usually* hard to manipulate [Conitzer and Sandholm, 2006f].

8.1 A failure of the revelation principle with bounded agents

As we have seen in earlier chapters, in many real-world mechanism design settings, the center faces an intractable optimization problem in trying to execute the mechanism. In this section, we question the focus on truthful mechanisms when the setting requires the solution of computationally hard problems. In particular, we show that there are settings where by abandoning truthful mechanisms, we can shift a computationally hard problem from the center to one of the agents. Additionally, whereas not being able to cope with the issue of computational hardness would have hurt the center in achieving its objective, if the agent is unable to cope with it, this actually helps the designer in achieving its objective.

We first observe that dominant strategy implementation and Bayes-Nash implementation differ only on what agents can be expected to know about each other's types and actions. An interesting special case is that of games where only one agent needs to choose an action. In this case, the acting agent always knows everything there is to know about the other agents' actions (namely, nothing). So, both solution concepts coincide here. We prove the remaining two theorems for this types of game, so the results hold both for dominant strategy implementation and Bayes-Nash implementation.

Theorem 51 *Suppose that the center is trying to maximize social welfare, and neither payments nor randomization are allowed.*[2] *Then, even with only two agents (one of whom does not even report a type, so dominant strategy implementation and Bayes-Nash implementation coincide), there exists a family of preference aggregation settings such that:*

[2] It is not immediately clear if this result can be extended to cases with payments or randomization; we leave this as a question for future research.

- *the execution of any optimal truthful mechanism is NP-complete for the center, and*
- *there exists a non-truthful mechanism which 1) requires the center to carry out only polynomial computation, and 2) makes finding any beneficial insincere revelation NP-complete for the type-reporting agent. Additionally, if the type-reporting agent manages to find a beneficial insincere revelation, or no beneficial insincere revelation exists, the social welfare of the outcome is identical to the social welfare that would be produced by any optimal truthful mechanism. Finally, if the type-reporting agent does not manage to find a beneficial insincere revelation where one exists, the social welfare of the outcome is* strictly greater *than the social welfare that would be produced by any optimal truthful mechanism.*

Put in perspective, the mechanism designer would reap two benefits from using the second, non-truthful mechanism rather than a truthful mechanism:

- Doing so shifts the computational hardness from the center to the agent. This can also be seen as a statement about how the social welfare that can be obtained by truthful mechanisms compares to the social welfare that can be obtained by non-truthful mechanisms, as follows. If it is computationally infeasible to execute the optimal truthful mechanism, the designer might resort to another truthful mechanism which merely approximates the social welfare obtained by the optimal truthful mechanism (this approach is often advocated in algorithmic mechanism design).
- If the agent cannot consistently solve instances of an NP-complete problem, then, even if the agent is trying to act strategically, using the second mechanism improves social welfare in some cases (and never decreases it).

Hence, (by the argument under the second bullet) the non-truthful mechanism—which is computationally feasible to execute—outperforms the optimal truthful mechanism, which (by the argument under the first bullet) in turn outperforms any computationally feasible truthful mechanism.

We emphasize that we do *not* require that agents will never be able to solve an NP-complete problem. Our result is more cautious than that: if agents do solve the NP-complete problem, nothing is lost; whereas if they do not solve it, something is gained.

Another point is that individual rationality is still maintained under this approach, by making sure that telling the truth still guarantees an agent nonnegative utility (even if telling the truth is not strategically optimal).

We are now ready to give the proof.

Proof: We are given a graph $G = (V, E)$ (with at least some edges); the outcome space is the set of all subsets of size k of the vertices, $\{X \subseteq V : |X| = k\}$. The type-reporting agent (agent 1) has the following type set Θ. For each $X \subseteq V$ with $|X| = k$, there is a type θ_X which occurs with probability $1/(\binom{n}{k} + 1)$. The utility function for these types is as follows: $u_1(\theta_X, X) = 4$ if X is an independent set (that is, there are no edges within X); $u_1(\theta_X, X) = 3$ if X is not an independent set; $u_1(\theta_X, Y) = 1$ if $X \neq Y$ and Y is an independent set; and $u_1(\theta_X, Y) = 0$ if $X \neq Y$ and Y is not an independent set. Additionally, there is a single additional type θ_0 which occurs with probability $1/(\binom{n}{k} + 1)$, and the utility function for it is given as follows: $u_1(\theta_0, X) = 1$ if X is an independent set; and $u_1(\theta_0, X) = 0$ if X is not an independent set. Agent 2, who does not report a

type, has the following utility function: $u_2(X) = 2$ if X is not an independent set; and $u_2(X) = 0$ if X is an independent set. Now let us consider creating a mechanism for such a setting that uses neither payments nor randomization.

First, we claim that all optimal *truthful* mechanisms are of the following form.

- If agent 1 reports a type θ_X, then choose outcome X;
- If agent 1 reports type θ_0, then 1) if there exists an independent set $X \subseteq V$ with $|X| = k$, choose such an independent set; or 2) if no independent set exists, choose any $X \subseteq V$ with $|X| = k$.

It is straightforward to verify that mechanisms of this form act in agent 1's best interest, that is, they always choose one of the outcomes that are optimal for agent 1 given its type. Hence, agent 1 never has any incentive to misreport its type, so these mechanisms are truthful. All that remains to show is that all other truthful mechanisms have strictly less expected social welfare than these. We first observe that the only case in which we get less than the optimal social welfare with the mechanisms of the given form is when agent 1 has type θ_0, and an independent set of size k exists. In this case, the mechanisms of the given form choose an independent set as the outcome, leading to a social welfare of 1; whereas a social welfare of 2 could have been obtained by choosing a set that is not independent. It follows that the expected social welfare that we get from one of the mechanisms of the given form is at most $\frac{1}{\binom{n}{k}+1}$ below the maximal expected social welfare that we could have obtained if the agents did not play strategically. Now consider an alternative truthful mechanism that, for some $X \subseteq V$ with $|X| = k$, does not choose X when agent 1 reports θ_X. In this case, this mechanism can obtain a social welfare of at most 2, whereas the optimal social welfare in this case is at least 4. It follows that the expected social welfare that we get from this mechanism is at least $\frac{2}{\binom{n}{k}+1}$ below the maximal expected social welfare that we could have obtained if the agents did not play strategically. Hence, all optimal truthful mechanisms always choose X when agent 1 reports θ_X. But then, if an independent set exists in G, an optimal truthful mechanism must choose such an independent set in the case where agent 1 reports θ_0: because if it does not, then when agent 1 has this type, it would benefit from misreporting its type as a type corresponding to the independent set—and the mechanism would no longer be truthful. Thus, we have established that all optimal truthful mechanisms are of the given form. We observe that executing such a mechanism requires solving an NP-complete problem, because we have to construct an independent set if it exists, which is NP-complete.

Now consider the following mechanism:

- If agent 1 reports a type θ_X, then choose outcome X;
- If agent 1 reports type θ_0, then choose some $X \subseteq V$ with $|X| = k$ that is not an independent set.

We observe that this mechanism is computationally easy to execute. Also, this mechanism is not truthful if there is an independent set, because in this case, if agent 1 has type θ_0, it would be better off reporting the type corresponding to the independent set. However, there are no other beneficial insincere revelations. Thus, it is straightforward to verify that if agent 1 always reports the type

that is strategically optimal for it, the outcome of this mechanism is always identical to that of one of the optimal truthful mechanisms. Of course, in order for agent 1 to always report the type that is strategically optimal for it, agent 1 needs to construct an independent set (if possible) when it has type θ_0. Because this problem is NP-complete, it is reasonable to suspect that agent 1 will not always be able to construct such a set even when it exists. If agent 1 indeed fails to construct an independent set in this case, the outcome will be some $X \subseteq V$ with $|X| = k$ that is not an independent set. This outcome actually has a social welfare of 2, as opposed to the social welfare of 1 that would have been obtained if agent 1 had managed to construct an independent set. Hence the social welfare is strictly greater than in the case where agent 1 has unlimited computational power; and hence it is also a greater than it would have been with an optimal truthful mechanism. ■

Given this example setting in which the revelation principle "fails" in the sense that non-truthful mechanisms can outperform truthful ones,[3] one may wonder whether similar phenomena occur in other, more standard settings. In the remainder of this chapter, we will study whether this is so for voting settings.

8.2 Tweaking voting protocols to make manipulation hard

Early, seminal work on the complexity of manipulating elections demonstrated that several voting rules are hard to manipulate, including the second-order Copeland rule [Bartholdi *et al.*, 1989a] and the STV rule [Bartholdi and Orlin, 1991]. In this section we take the next step of designing new protocols that are especially hard to manipulate. Rather than designing these protocols from scratch, we show how to tweak existing voting protocols to make manipulation computationally much more difficult, while leaving much of the original nature of the protocol intact, for the following reasons:

- Results on the computational complexity induced by a tweak typically apply to a large family of protocols.
- Some of the original protocol's nice theoretical properties are preserved by the tweak. For example, if a protocol satisfies the *Condorcet criterion* (a candidate that wins all its pairwise elections always wins the election), the tweak will preserve this property.
- In practice, it will be much easier to replace a currently used protocol with a tweaked version of it, than with an altogether new protocol.

The type of tweak we introduce is the following. All the candidates are paired in a *preround*; of each pair of candidates, only the winner of their pairwise election survives. (Recall that the winner of the pairwise election between two candidates is the candidate that is ranked above the other more often in the votes.) After the preround, the original protocol is executed on the remaining candidates. The *schedule* of the preround (i.e., who faces who) can be determined before the votes are collected; after the votes are collected; or while the votes are collected (the processes are interleaved). We study these three cases in Subsections 8.2.2, 8.2.3, and 8.2.4, respectively.

[3] Of course, the principle does not fail in the sense that the formal statements of it are wrong; it is merely that the preconditions of the theorem fail to hold when agents are computationally bounded.

8.2.1 Definitions

Voting protocols

For the purposes of this section, a *deterministic* protocol is a protocol of the type that we have considered earlier in this dissertation, that is, a function from the set of all combinations of votes to C. (We will only be interested in the winner of the election in this section, not in an entire ranking of candidates.) A *randomized* protocol is a function from the set of all combinations of votes to probability distributions over C. An *interleaved* protocol is a procedure for alternating between collecting (eliciting) parts of the voters' votes (*e.g.* whether they prefer candidate a to candidate b) and drawing and publishing random variables (such as parts of the schedule for an election), together with a function from the set of all combinations of votes and random variables to C.

Preround

The tweaks we study in this section all involve the addition of a preround. We will now define how this works.

Definition 37 *Given a protocol P, the new protocol obtained by adding a preround to it proceeds as follows:*

1. *The candidates are paired. If there is an odd number of candidates, one candidate gets a bye.*
2. *In each pairing of two candidates, the candidate losing the pairwise election between the two is eliminated. A candidate with a bye is never eliminated.*
3. *On the remaining candidates, P is executed to produce a winner. For this, the implicit votes over the remaining candidates are used. (For example, if a voter voted $a \succ b \succ c \succ d \succ e$, and b and c were eliminated, the voter's implicit vote is $a \succ d \succ e$.)*

The pairing of the candidates is also known as the schedule *for the preround. If the schedule is decided and published before the votes are collected, we have a* deterministic preround ($DPRE$)*, and the resulting protocol is called $DPRE + P$. If the schedule is drawn completely randomly after the votes are collected, we have a* randomized preround ($RPRE$)*, and the resulting protocol is called $RPRE + P$. Finally, if the votes are elicited incrementally, and this elicitation process is interleaved with the scheduling-and-publishing process (which is again done randomly), as described in detail in Subsection 8.2.4, we have an* interleaved preround ($IPRE$)*, and the resulting protocol is called $IPRE + P$.*

Manipulation

We now define the computational problem of manipulation that we study in this section. Other definitions of manipulation are possible: in the next section, we will give a more thorough analysis of the different variants of the manipulation problem, and study some of the other variants.

Definition 38 (CONSTRUCTIVE-MANIPULATION) *We are given a protocol P, a candidate set C, a preferred candidate p, and a set of votes S corresponding to all the other voters' votes.*

The manipulator has yet to decide on its vote, and wants to make p win. Then the constructive manipulation question is:

- *(For deterministic protocols) Can the manipulator cast its vote to make p win under P?*
- *(For randomized protocols) Can the manipulator cast its vote to make the probability of p winning under P at least some given $k \in [0, 1]$?*
- *(For interleaved protocols) Given the initial random choices (if any) by the protocol, is there a contingency plan (based on the random decisions the protocol takes between eliciting parts of the votes) for the manipulator to answer the queries to make the probability of p winning under P at least some given $k \in [0, 1]$?*

8.2.2 NP-hardness when scheduling precedes voting

In this subsection, we examine the complexity induced by the preround when the voters know the schedule before they vote.

A sufficient condition for NP-hardness

We present a sufficient condition under which adding a preround with a preannounced schedule makes manipulation NP-hard. The condition can be thought of as an NP-hardness reduction template. If it is possible to reduce an arbitrary SAT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol—with a preround—is NP-hard to manipulate.

Theorem 52 *Given a voting protocol P, suppose that it is possible, for any Boolean formula ϕ in conjunctive normal form (i.e., a SAT instance), to construct in polynomial time a set of votes over a candidate set containing at least $\{p\} \cup C_L$ where $C_L = \{c_l : l \in L\}$ (L is the set of literals $\{+v : v \in V\} \cup \{-v : v \in V\}$, where V is the set of variables used in ϕ), with the following properties:*

- *(Property 1a) If we remove, for each $v \in V$, one of c_{+v} and c_{-v}, p would win an election under protocol P against the remaining candidates if and only if for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k. This should hold even if a single arbitrary vote is added.*
- *(Property 1b) For any $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election after these votes.*

Then CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard (and NP-complete if P is deterministic and can be executed in polynomial time).

Proof: Consider the following election under $DPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides those in C_L. To each of the constructed votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the nonmanipulators' votes. A single manipulator's vote is yet to be added. Let the schedule for the preround

be as follows: for each v, c_{+v} and c_{-v} face each other in the preround; and every other original candidate faces (and, because of the dummy candidates' position in the votes, defeats) a dummy candidate. Thus, the set of candidates that make it through the preround consists of, for each $v \in V$, one of c_{+v} and c_{-v}; and all the other original candidates. The manipulator's vote will decide the winner of every c_{+v} vs. c_{-v} match-up, because by property 1b, all these pairwise elections are currently tied. Moreover, it is easy to see that the manipulator can decide the winner of each of these match-ups independently of how it decides the winners of the other match-ups. Thus, we can think of this as the manipulator giving the variables truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 1a it then follows that p wins if and only if the manipulator's assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Hence there is a successful constructive manipulation if and only if there is a solution to the SAT instance, and it follows that CONSTRUCTIVE-MANIPULATION in $DPRE + P$ is NP-hard. (It is also in NP if P is deterministic and can be executed in polynomial time, because in this case, given a vote for the manipulator, it can be verified in polynomial time whether this vote makes p win). ■

Examples

We now show how to apply Theorem 52 to the well-known protocols we discussed, thus showing that each of these protocols—with a preround—is NP-hard to manipulate.

Theorem 53 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the plurality rule.*

When it does not matter for our proofs whether a given vote is $a \succ b \succ c$ or $b \succ a \succ c$, we write $\{a, b\} \succ c$.

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $4|K| + 2$ votes ranking the candidates $p \succ C_L \succ C_K$; for each $k \in K$, $4|K|$ votes ranking the candidates $c_k \succ \{c_{cl} \in C_K : cl \neq k\} \succ C_L \succ p$; and for each $k \in K$, 4 votes ranking the candidates $\{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\} \succ \{c_{cl} \in C_K : cl \neq k\} \succ p$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L.) We now show property 1a is satisfied. We first observe that regardless of which of the candidates corresponding to literals are removed, p will get $4|K| + 2$ votes. Now, if for some $k \in K$, all the candidates c_l with $l \in L, l \in k$ are removed, then c_k will get at least $4|K| + 4$ votes and p will not win. On the other hand, if for each $k \in K$, at least one candidate c_l with $l \in k$ remains, then each of the c_k will get precisely $4|K|$ votes. Because each remaining c_l can get at most $4|K|$ votes as well, p will win. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 54 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the Borda rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$; plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$, which we order in some arbitrary way to get $\{c_1, \ldots, c_{|K|}\}$. Let M be the total number of candidates this defines. Then, let the set of votes be as follows: for every $c_i \in C_K$, $4M$ votes ranking the candidates $c_{i+1} \succ c_{i+2} \succ \ldots \succ c_{|K|} \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_{i-1} \succ \{c_l \in C_L : l \in c_i\} \succ c_i \succ \{c_l \in L : l \notin c_i\}$; (here, the slight abuse of notation $l \in c_i$ means that l occurs in the clause corresponding to c_i;) $4M$ votes ranking the candidates $c_1 \succ c_2 \succ \ldots \succ c_{|K|} \succ p \succ C_L$; one vote $c_1 \succ c_2 \succ \ldots \succ c_{|K|} \succ C_L \succ p$; one vote $c_{|K|} \succ c_{|K|-1} \succ \ldots \succ c_1 \succ C_L \succ p$; and finally, $4|K|M$ votes ranking the candidates $p \succ c_1 \succ c_2 \succ \ldots \succ c_n \succ C_L$, and $4|K|M$ votes ranking the candidates $c_n \succ c_{n-1} \succ \ldots c_1 \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l.) We now show property 1a is satisfied. It is easy to see that none of the c_l can win, regardless of which of them are removed. Thus, we only need to consider the c_i and p. The last $8|K|M$ votes will have no net effect on the relative scores of these candidates, so we need not consider these here. After the first $4(|K|+1)M$ votes, any c_k for which all the c_l with $l \in k$ have been removed will be tied with p, and any other c_k will be at least $4M$ points behind p. Finally, from the last remaining two votes, any c_k ($k \in K$) will gain $2M - 2|V| - |K| - 1$ points on p. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in L$ with $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least $M - |V|$ points, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 55 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the maximin rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $C_K = \{c_k : k \in K\}$. Then, let the set of votes be as follows: $8|K|$ votes ranking the candidates $p \succ C_L \succ C_K$, $8|K|$ votes ranking the candidates $C_L \succ C_K \succ p$, and $8|K|$ votes ranking the candidates $C_K \succ p \succ C_L$; $4|K|$ votes ranking the candidates $C_L \succ p \succ C_K$, $4|K|$ votes ranking the candidates $C_K \succ C_L \succ p$, and, for each $k \in K$, 4 votes ranking the candidates $p \succ \{c_{cl} \in C_K : cl \neq k\} \succ \{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\}$; and finally, 2 votes ranking the candidates $p \succ C_K \succ C_L$, and 2 votes ranking the candidates $C_K \succ p \succ C_L$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the c_l.) We now show property 1a is satisfied. Regardless of which of the candidates corresponding to literals are removed, p's worst score in a pairwise election is against any of the c_k, namely $16|K| + 2$. Any c_k for which all the c_l with $l \in k$ have been removed will get its worst pairwise election score against any of the C_L, namely $16|K| + 4$. Finally, any other c_k will get its worst pairwise election score against one of the c_l with $l \in k$, namely, $16|K|$. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least 2, so a

single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 56 *There exists a reduction that satisfies properties 1a and 1b of Theorem 52 under the STV rule.*

Proof: Given the formula ϕ, let the candidate set be the minimally required candidates $\{p\} \cup C_L$, plus a set of candidates corresponding to the set of clauses K of ϕ, $\{c_{cl} : cl \in K\}$, which we order in some arbitrary way to get $\{c_1, \ldots, c_{|K|}\}$; plus $4|K|$ additional candidates $c_{a_1}, \ldots, c_{a_{8|K|}}$. Then, let the set of votes be as follows: for each $k \in K$, 4 votes ranking the candidates $\{c_l \in C_L : l \in k\} \succ c_k \succ \{c_l \in C_L : l \notin k\} \succ p \succ \{c_{a_j}\}$; for each c_{a_i}, 2 votes ranking the candidates $c_{a_i} \succ c_1 \succ c_1 \succ \ldots \succ c_{|K|} \succ p \succ C_L \succ \{c_{a_j} : j \neq i\}$; and finally, 4 votes ranking the candidates $p \succ C_K \succ C_L \succ \{c_{a_j}\}$. Additionally, we require that these votes are such that after counting them, for each $v \in V$, c_{+v} and c_{-v} are tied in their pairwise election, so that property 1b is satisfied. (This is possible because the total number of votes is even, and the majority of the votes do not yet have any restrictions on the order of the C_L.) We now show property 1a is satisfied. Regardless of which of the candidates corresponding to literals are removed, p will have 4 votes initially, and every c_{a_j} will have 2 votes initially. Any c_k ($k \in K$) for which all the c_l ($l \in L$) with $l \in k$ have been removed will have 4 votes initially. Any other c_k will have 0 votes initially, and hence drop out in the first round. Then, before p or any more c_k drop out, all the c_{a_j} will drop out, because they have only 2 votes initially and no votes will transfer to them. All the $8|K|$ votes that the c_{a_j} have initially will transfer either to the c_i that has the lowest index i among the remaining c_{cl}, or, if there are no remaining c_{cl}, to p. Because these $8|K|$ votes are the majority of votes in the election, it follows that the candidate to which all of these votes transfer will win the election. It follows that p wins if and only if for every clause $k \in K$, there is some $l \in L$ with $l \in k$ such that c_l has not been removed. In both cases there is a "margin" of at least 2 in every round, so a single additional vote will not change this. Thus, property 1a is satisfied. ■

Theorem 57 *In any of $DPRE{+}plurality$, $DPRE{+}Borda$, $DPRE{+}maximin$, and $DPRE{+}STV$[4] , CONSTRUCTIVE-MANIPULATION is NP-complete.*

Proof: NP-hardness is immediate from the previous theorems. The problem is in NP because these protocols can be executed in polynomial time. ■

In the next subsections, we will raise the bar and bring the problem of manipulating elections to higher complexity classes by abandoning the assumption that the schedule for the preround should be known in advance.

[4]The NP-completeness of manipulating $DPRE + STV$ is, in itself, not that interesting, because STV is already NP-hard to manipulate without the preround as we discussed. Nevertheless, our method highlights a different aspect of the NP-hardness of manipulating $DPRE + STV$. We build on this reduction later to prove PSPACE-hardness of manipulating STV with a preround when the scheduling of the preround is interleaved with the vote elicitation.

8.2.3 #P-hardness when voting precedes scheduling

In this subsection, we will examine the complexity induced by the preround when the schedule is drawn completely (uniformly) randomly after all the votes have been collected.

A sufficient condition for #P-hardness

We present a sufficient condition for a voting protocol to become #P-hard[5] to manipulate in this setting. Again, this condition can be thought of as a reduction template. If it is possible to reduce an arbitrary PERMANENT instance to a set of votes satisfying certain properties under the given voting protocol, that protocol is #P-hard to manipulate when a randomized preround is added to it. (In the PERMANENT problem, we are given a bipartite graph B with the same number of vertices k in both parts, and are asked how many matchings there are. This problem is #P-complete [Valiant, 1979].)

Theorem 58 *Given a voting protocol P, suppose that it is possible, for any bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), to construct in polynomial time a set of votes over the candidate set $\{c_1, \ldots, c_{2k}, p\}$ (where c_i corresponds to vertex i in B) with the following properties:*

- *(Property 2a) If we remove k of the c_i, p would win an election under protocol P against the remaining c_i if and only if the removed c_i are exactly all the c_i with $k+1 \leq i \leq 2k$;*
- *(Property 2b) p loses its pairwise election against all c_i with $k+1 \leq i \leq 2k$;*
- *(Property 2c) For any $1 \leq i \leq k$ and $k+1 \leq j \leq 2k$, c_i defeats c_j in their pairwise election if and only if in B, there is an edge between vertices i and j.*
- *(Property 2d) All the previous properties still hold with any additional single vote.*

Then CONSTRUCTIVE-MANIPULATION in $RPRE + P$ is #P-hard.

Proof: Given the set of votes constructed on the basis of an arbitrary B, let us compute the probability that p wins under the protocol $RPRE + P$ with only these votes. In the preround, there are k matches and one bye. By property 2a, p will win the election if and only if the k candidates eliminated in this preround are precisely all the c_i with $k+1 \leq i \leq 2k$. By property 2b, p could not win a preround match against any of these, so p will win the election if and only if it gets the bye, and each of the c_j with $k+1 \leq j \leq 2k$ faces one of the c_i with $1 \leq i \leq k$ that defeats it in the preround. Then, by property 2c, it follows that p wins if and only if the preround pairing corresponds to a matching in B. Thus the probability of p winning is $\frac{m_B}{e(2k,2k+1)}$, where m_B is the number of matchings in B and $e(2k, 2k+1)$ is the number of different ways to pair $2k$ of the $2k+1$ candidates in the preround (which is straightforward to compute). Thus, evaluating p's chances of winning in this election is at least as hard as counting the number of matchings in an arbitrary B, which is #P-hard. Moreover, because we can compute p's chances of winning solely

[5] #P is the class of problems where the task is to count the number of solutions to a problem in NP.

on the basis of properties 2a, 2b, and 2c, and by property 2d, these properties are maintained for any single additional vote, it follows that a manipulator cannot affect p's chances of winning. Thus, CONSTRUCTIVE-MANIPULATION in this case simply comes down to computing p's chances of winning, which is #P-hard as demonstrated. ■

A broadly applicable reduction

In this subsubsection we present a single broadly applicable reduction which will satisfy the preconditions of Theorem 58 for many voting protocols, thus proving them #P-hard to manipulate when the voting precedes the preround scheduling.

Definition 39 *We label the following reduction R_1. Given a bipartite graph B with the same number of vertices k in both parts (labeled 1 to k in one part, $k+1$ to $2k$ in the other), we construct the following set of $12k^3 + 2k^2$ votes:*

- $6k^3$ *votes that rank the candidates* $c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{2k} \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_k$*;*
- $3k^2$ *votes that rank the candidates* $p \succ c_k \succ c_{k-1} \succ \ldots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{k+1}$*;*
- $6k^3 - 3k^2$ *votes that rank the candidates* $c_k \succ c_{k-1} \succ \ldots \succ c_1 \succ c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{k+1} \succ p$*;*
- *For each edge* (i, j) *in* B *(*$1 \leq i \leq k$*,* $k+1 \leq j \leq 2k$*), one vote that ranks the candidates* $c_i \succ c_j \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_{i-1} \succ c_{i+1} \succ \ldots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{j-1} \succ c_{j+1} \succ \ldots \succ c_{2k}$*, and another one that ranks them* $c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{j+1} \succ c_{j-1} \succ \ldots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \ldots \succ c_{i+1} \succ c_{i-1} \succ \ldots \succ c_1 \succ p \succ c_i \succ c_j$ *(i.e., the inverse of the former vote, apart from* c_i *and* c_j *which have maintained their order);*
- *For each pair* i, j *without an edge between them in* B *(*$1 \leq i \leq k$*,* $k+1 \leq j \leq 2k$*), one vote that ranks the candidates* $c_j \succ c_i \succ p \succ c_1 \succ c_2 \succ \ldots \succ c_{i-1} \succ c_{i+1} \succ \ldots \succ c_k \succ c_{k+1} \succ c_{k+2} \succ \ldots \succ c_{j-1} \succ c_{j+1} \succ \ldots \succ c_{2k}$*, and another one that ranks them* $c_{2k} \succ c_{2k-1} \succ \ldots \succ c_{j+1} \succ c_{j-1} \succ \ldots \succ c_{k+1} \succ c_k \succ c_{k-1} \succ \ldots \succ c_{i+1} \succ c_{i-1} \succ \ldots \succ c_1 \succ p \succ c_j \succ c_i$ *(i.e., the inverse of the former vote, apart from* c_j *and* c_i *which have maintained their order).*

We now have to show that this reduction satisfies the preconditions of Theorem 58. We start with the properties that are protocol-independent.

Theorem 59 R_1 *satisfies properties 2b and 2c of Theorem 58 (under any protocol* P*, because these properties are independent of* P*), even with a single additional arbitrary vote.*

Proof: In the pairwise election between p and any one of the c_i with $k+1 \leq i \leq 2k$, p is ranked higher in only $4k^2$ votes, and thus loses the pairwise election. So property 2b is satisfied. For a pairwise election between some c_i and c_j ($1 \leq i \leq k$ and $k+1 \leq j \leq 2k$), the first $12k^3$ votes' net contribution to the outcome in this pairwise election is 0. Additionally, the two votes associated

with any pair q, r ($1 \leq q \leq k$ and $k+1 \leq r \leq 2k$) also have a net contribution of 0, if either $q \neq i$ or $r \neq j$. The only remaining votes are the two associated with the pair i, j, so c_i wins the pairwise election by 2 votes if there is an edge (i, j) in B, and c_j wins the pairwise election by 2 votes otherwise. So property 2c is satisfied. Because both are satisfied with a "margin" of at least 2, a single additional vote will not change this. ■

Finally, because property 2a is protocol-dependent, we need to prove it for our reduction on a per-protocol basis. This is what the following four theorems achieve.

Theorem 60 *R_1 satisfies property 2a of Theorem 58 under the plurality rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, p can get at most $5k^2$ votes, whereas the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes, so p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes, which is more than half the votes, so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 61 *R_1 satisfies property 2a of Theorem 58 under the Borda rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the highest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it h. The first $12k^3$ votes will put h at least $9k^3 - 3k^2$ points ahead of p. ($12k^3 - 3k^2$ of them rank h above p, and the $3k^2$ others can give p an advantage of at most k each.) The $2k^2$ remaining votes can contribute an advantage to p of at most k each, and it follows that h will still have at least $7k^3 - 3k^2$ more points than p. So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, then there are two groups of $6k^3 - 3k^2$ among the first $12k^3$ votes which (over the remaining candidates) are each other's exact inverses and hence have no net effect on the scores. Also, the last $2k^2$ votes, which are organized in pairs, have no net effect on the score because (over the remaining candidates) the votes in each pair are each other's exact inverse. The remaining votes all rank p highest among the remaining candidates, so p wins. In both cases the "margin" is big enough that a single additional vote will not change this. ■

Theorem 62 *R_1 satisfies property 2a of Theorem 58 under the maximin rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, then in any pairwise election between such a candidate and p, p will get at most $5k^2$ votes. However, the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$ will get at least $6k^3$ votes in every one of its pairwise elections. So p does not win. On the other hand, if all the c_i with $k+1 \leq i \leq 2k$ are removed, p will get at least $6k^3 + 3k^2$ votes in every one of its pairwise elections, which is more than half the votes; so p wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 63 *R_1 satisfies property 2a of Theorem 58 under the STV rule. This holds even when there is a single additional arbitrary vote.*

Proof: If at least one of the c_i with $k+1 \leq i \leq 2k$ is not removed, consider the lowest-indexed remaining candidate among the c_i with $k+1 \leq i \leq 2k$; call it l. l will hold at least $6k^3$ votes as long as it is not eliminated, and p can hold at most $5k^2$ votes as long as l is not eliminated. It follows that p will be eliminated before l, so p does not win. On the other hand, if all the the c_i with $k+1 \leq i \leq 2k$ are removed, p will hold at least $6k^3 + 3k^2$ votes throughout, which is more than half the votes; so p cannot be eliminated and wins. In both cases there is a "margin" of at least 2, so a single additional vote will not change this. ■

Theorem 64 *In any of $RPRE+plurality$, $RPRE+Borda$, $RPRE+maximin$, and $RPRE+STV$, CONSTRUCTIVE-MANIPULATION is #P-hard.*

Proof: Immediate from the previous theorems. ■

8.2.4 PSPACE-hardness when scheduling and voting are interleaved

In this subsection, we increase the complexity of manipulation one more notch, to PSPACE-hardness,[6] by interleaving the scheduling and vote elicitation processes.

We first discuss the precise method of interleaving required for our result. The method is detailed and quite complicated. Nevertheless, this does *not* mean that the interleaving should always take place in this particular way in order to have the desired hardness. If the interleaving method used for a particular election is (say, randomly) chosen from a wider (and possibly more naturally expressed) class of interleaving methods containing this one, our hardness result still goes through, as hardness carries over from the specific to the general. Thus, our goal is to find the most specific method of interleaving for which the hardness still occurs, because this gives us the most information about more general methods. We only define the method for the case where the number of candidates is a multiple of 4 because this is the case that we will reduce to (so it does not matter how we generalize the protocol to cases where the number of candidates is not a multiple of 4).

Definition 40 *$IPRE$ proceeds as follows:*

1. *Label the matchups (a matchup is a space in the preround in which two candidates can face each other; at this point they do not yet have candidates assigned to them) 1 through $\frac{|C|}{2}$;*
2. *For each matchup i, assign one of the candidates to play in it, and denote this candidate by $c(i,1)$. Thus, one of the candidates in each matchup is known.*
3. *For some k which is a multiple of 4, for each i with $1 \leq i \leq k$, assign the second candidate to play in matchup i, and denote this candidate $c(i,2)$. Thus, we have k fully scheduled matchups.*

[6] PSPACE is the class of problems solvable in polynomial *space*.

4. *For each pair of matchups $(2i-1, 2i)$ with $i > \frac{k}{2}$, assign two more candidates to face the candidates already in these two matchups, and denote them $c((2i-1, 2i), 1)$ and $c((2i-1, 2i), 2)$. (Thus, at this point, all that still needs to be scheduled is, for each i, which of these two faces $c(2i-1, 1)$ and which $c(2i, 1)$.)*

5. *For $i = \frac{k}{2} + 1$ to $\frac{|C|}{4}$:*

   • *Randomly decide which of $c((2i-1, 2i), 1)$ and $c((2i-1, 2i), 2)$ faces $c(2i-1, 1)$, and which faces $c(2i, 1)$. Denote the former $c(2i-1, 2)$, the latter $c(2i, 2)$,*

   • *Ask all the voters whether they prefer $c(i - \frac{k}{2}, 1)$ or $c(i - \frac{k}{2}, 2)$. (We observe that, even if the number of already scheduled matchups is $k = 0$, the elicitation process trails behind the scheduling process by a factor 2.)*

6. *Elicit the remainder of all the votes.*

One important property of this elicitation process is that the voters are treated symmetrically: when a query is made, it is made to all of the voters in parallel. Thus, no voter gets an unfair advantage with regard to knowledge about the schedule. Another important property is that the elicitation and scheduling process at no point depends on how the voters have answered earlier queries. Thus, voters cannot make inferences about what other voters replied to previous queries on the basis of the current query or the current knowledge about the schedule. These two properties guarantee that many issues of strategic voting that may occur with vote elicitation [Conitzer and Sandholm, 2002c] in fact do not occur here.

We are now ready to present our result.

Theorem 65 *Given a voting protocol P, suppose that it is possible, for any Boolean formula ϕ in conjunctive normal form (i.e., a SAT instance) over variables $V = X \cup Y$ with $|X| = |Y|$ (and corresponding literals L), to construct in polynomial time a set of votes over a candidate set containing at least $\{p\} \cup C_L \cup \{c_y^1 : y \in Y\}$ with the following properties:*

- *(Property 3a) If we remove, for each $v \in V$, one of c_{+v} and c_{-v}, p would win an election under protocol P against the remaining candidates if and only if for every clause $k \in K$ (where K is the set of clauses in ϕ), there is some $l \in L$ such that c_l has not been removed, and l occurs in k. This should hold even if a single arbitrary vote is added.*

- *(Property 3b) For any $x \in X$, c_x and c_{-x} are tied in their pairwise election after these votes.*

- *(Property 3c) For any $y \in Y$, c_y and c_{-y} are both losing their pairwise elections against c_y^1 by at least 2 votes (so that they will lose them regardless of a single additional vote).*

Then CONSTRUCTIVE-MANIPULATION in $IPRE + P$ is PSPACE-hard (and PSPACE-complete if P can be executed in polynomial space).

Proof: Consider the following election under $IPRE + P$. Let the candidate set be the set of all candidates occurring in the votes constructed from ϕ (the "original candidates"), plus one dummy candidate for each of the original candidates besides the c_{+v} and c_{-v}. To each of the constructed

votes, add all the dummy candidates at the bottom; let the resulting set of votes be the set of the nonmanipulators' votes, according to which they will answer the queries posed to them. The manipulator has yet to decide on its strategy for answering queries. After step 4 (according to Definition 40) of $IPRE + P$ (up to which point the manipulator will not have had to make any decisions), let the situation be as follows:

- The number of already fully scheduled matchups is $k = \frac{|C|}{2} - 2|Y|$. In matchup i ($1 \leq i \leq |X|$), c_{+x_i} faces c_{-x_i}. In the remaining fully scheduled matchups, candidates not corresponding to a literal face a dummy candidate.
- Matchups $k+2i-1$ and $k+2i$ ($1 \leq i \leq |Y|$) already have candidates c_{+y_i} and c_{-y_i} in them, respectively. The other two candidates to be assigned to these rounds are $c^1_{y_i}$ and a dummy candidate.

Thus, what will happen from this point on is the following. For i ranging from 1 to $|X|$, first the protocol will schedule which of c_{+y_i} and c_{-y_i} face which of $c^1_{y_i}$ and the dummy candidate. The c_l facing the dummy will move on, and the other will be defeated by $c^1_{y_i}$, by property 3c. Second, everyone will be asked which of c_{+x_i} and c_{-x_i} is preferred, and because the nonmanipulators will leave this pairwise election tied by property 3b, the manipulator's vote will be decisive. Thus, we can think of this as nature and the manipulator alternatingly giving the variables in Y and X respectively truth-values: v is set to *true* if c_{+v} survives, and to *false* if c_{-v} survives. By property 3a it then follows that p wins if and only if the resulting assignment satisfies all the clauses, i.e. is a solution to the SAT instance. Thus, the manipulator's strategy for setting variables should aim to maximize the chance of the SAT instance being satisfied eventually. But this is exactly the problem STOCHASTIC-SAT, which is PSPACE-complete [Papadimitriou, 1985].

If P can be executed in polynomial space, the manipulator can enumerate all possible outcomes for all possible strategies in polynomial space, so the problem is also in PSPACE. ■

Because the preconditions of Theorem 65 are similar to those of Theorem 52, we can build on our previous reductions to apply this theorem to the well-known protocols.

Theorem 66 *For each of $plurality$, $Borda$, $maximin$, and STV, there exists a reduction that satisfies properties 3a, 3b and 3c of Theorem 65. Thus, In any of $IPRE + plurality$, $IPRE + Borda$, $IPRE + maximin$, and $IPRE + STV$, CONSTRUCTIVE-MANIPULATION is PSPACE-complete.*

Proof: We can modify the reductions from Subsection 8.2.2 to satisfy the preconditions of Theorem 65. This is done by adding in the c^1_y in such a way as to achieve property 3c (ranking them just above their corresponding c_y and c_{-y} in slightly more than half the votes), while preserving property 3a (by ranking them as low as possible elsewhere). ■

This concludes the part of this dissertation studying how to tweak voting protocols to make them harder to manipulate. In the next section, we focus on existing, untweaked voting rules: although these rules are easy to manipulate in the sense described in this section (with the exception of STV), it turns out that there are other manipulation problems that are difficult even for these rules, even with few candidates.

8.3 Hardness of manipulating elections with few candidates

We did some of the work in this subsection jointly with Jérôme Lang (IRIT, France).

The hardness results that we proved in the previous section (as well as similar hardness results proven by others [Bartholdi *et al.*, 1989a; Bartholdi and Orlin, 1991; Elkind and Lipmaa, 2005a]) assume that not only the number of voters but also *the number of candidates* is unbounded. Such hardness results lose relevance when the number of candidates is small, because manipulation algorithms that are exponential only in the number of candidates (and only slightly so) might be available. In this section, we first give such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) rule, which has been shown hard to manipulate in the above sense. The algorithm applies whether or not the voters are weighted.

This motivates the core of this section, which studies the complexity of manipulating elections where the number of candidates is a small constant. Restricting the number of candidates to a constant reduces the number of possible votes for a single voter to a constant. If the voters all have equal weight in the election, the number of *de facto* possible combinations of votes that even a coalition can submit is polynomial in the number of voters in the coalition (since the voters have equal weight, it does not matter which agent in the coalition submitted which vote; only the multiplicities of the votes from the coalition matter). We thus get the following straightforward result.

Proposition 9 *Let there be a constant number of candidates, and suppose that evaluating the result of a particular combination of votes by a coalition is in P. If there is only one voter in the coalition, or if the voters are unweighted, the manipulation problem is in P. (This holds for all the different variants of the manipulation problem, discussed later.)*

Proof: The manipulators (an individual agent or a coalition) can simply enumerate and evaluate all possibilities for their votes (there is a polynomial number of them). Specifically, when there are n voters in the coalition and m candidates, then there are at most $(n+1)^{m!}$ possibilities, because for every one of the $m!$ possible orderings of the candidates there must be between 0 and n voters in the coalition voting according to this ordering (and, because the voters are unweighted, it does not matter which voters they are). This expression is polynomial in n. ■

In particular, in the complete-information manipulation problem in which the votes of the noncolluders are known, evaluating the result of a (coalitional) vote is roughly as easy as determining the winner of an election.[7] This leaves open two avenues for deriving high complexity results with few candidates. First, we may investigate the complete-information coalitional manipulation problem when voters have *different weights*. While many human elections are unweighted, the introduction of weights generalizes the usability of voting schemes, and can be particularly important in multiagent systems settings with very heterogenous agents. As a second avenue, we may ask whether there are reasonable settings where *evaluating* a manipulation is NP-hard. For instance, if

[7] Recall from Chapter 3 that there exist voting rules where determining the winner is computationally hard [Bartholdi *et al.*, 1989b; Hemaspaandra *et al.*, 1997; Cohen *et al.*, 1999; Dwork *et al.*, 2001; Rothe *et al.*, 2003; Davenport and Kalagnanam, 2004; Ailon *et al.*, 2005], including the Slater and Kemeny rules—but this is only so for large numbers of candidates.

we merely have probability distributions on the non-colluders' votes, how does the complexity of determining the probability that a given candidate wins change?

We devote most of this section to studying the first avenue. We study both *constructive* manipulation (making a given candidate win) and *destructive* manipulation (making a given candidate not win). We characterize the exact number of candidates for which manipulation becomes hard for *plurality*, *Borda*, *STV*, *Copeland*, *maximin*, *veto*, *plurality with runoff*, *regular cup*, and *randomized cup* rules. It turns out that the voting rules under study become hard to manipulate at 3 candidates, 4 candidates, 7 candidates, or never. The remainder of this section is devoted to the second avenue, by showing that hardness results from the complete-information coalitional weighted manipulation problem imply similar hardness results in the incomplete-information setting, *even without the assumptions of multiple manipulators and weighted votes*.

8.3.1 Manipulating an election

Due to Proposition 9, we cannot hope to obtain hardness of manipulation in the sense of Section 8.2. Hence, we will introduce more general manipulation problems. To do so, we first discuss the different dimensions of the election manipulation problem:

1. *What information do the manipulators have about the nonmanipulators' votes?* In the *incomplete information* setting, the manipulators are uncertain about the nonmanipulators' votes. This uncertainty could be represented in a number of ways, for example, as a joint probability distribution over the nonmanipulators' votes. In the *complete information* setting, the manipulators know the nonmanipulators' votes exactly. We (initially) focus on the complete information case for the following reasons: 1a. It is a special case of any uncertainty model. Therefore, our hardness results directly imply hardness for the incomplete information setting. 1b. As we will demonstrate later in this section, hardness results for manipulation by coalitions in the complete information setting also imply hardness of manipulation *by individuals* in the incomplete information setting. 2. Results in the complete information setting measure only the *inherent* complexity of manipulation rather than any potential complexity introduced by the model of uncertainty.

2. *Who is manipulating: an individual voter or a coalition of voters?* Both of these are important variants, but we focus on coalitional manipulation for the following reasons: 1. In elections with many voters it is perhaps unlikely that an individual voter can affect the outcome—even with unlimited computational power. 2. For any constant number of candidates (even with an unbounded number of voters), manipulation by individuals in the complete information setting is computationally easy because the manipulator can enumerate and evaluate all its possible votes (rankings of candidates) in polynomial time, as we pointed out in the Introduction.[8] 3. Again, as we will demonstrate later in this section, hardness results for manipulation *by coalitions* in the complete information setting also imply hardness of manipulation *by individuals* in the incomplete information setting.

[8]This assumes that the voting rule is easy to execute—as most rules are (including all the ones under study in this section).

3. *Are the voters weighted or unweighted?* Both of these are important variants, but we focus on weighted voters for the following reasons: 1. In the unweighted case, for any constant number of candidates (even with an unbounded number of voters), manipulation by a coalition in the complete information setting is computationally easy because the coalition can enumerate and evaluate all its effectively different vote vectors, as we pointed out earlier. (We recall that the number of effectively different vote vectors is polynomial due to the interchangeability of the different equiweighted voters, see Proposition 9.) 2. As we will demonstrate later in this section, hardness results for manipulation by *weighted* coalitions in the complete information setting also imply hardness of evaluating the probabilities of different outcomes in the incomplete information setting with *unweighted* (but correlated) voters. 3. In many real-world elections, voters are in fact weighted, for example, by their ownership share in the company, by seniority, or by how many other individuals they represent.

4. *What is the goal of manipulation?* We study two alternative goals: trying to make a given candidate win (we call this *constructive* manipulation), and trying to make a given candidate *not* win (we call this *destructive* manipulation). Besides these goals being elegantly crisp, there are fundamental theoretical reasons to focus on these goals.

   First, hardness results for these goals imply hardness of manipulation under any game-theoretic notion of manipulation, because our manipulation goals are always special cases. (This holds both for deterministic and randomized voting rules.) At one extreme, consider the setting where there is one candidate that would give utility 1 to each of the manipulators, and all other candidates would give utility 0 to each of the manipulators. In this case the only sensible game-theoretic goal for the manipulators is to make the preferred candidate win. This is exactly our notion of constructive manipulation. At the other extreme, consider the setting where there is one candidate that would give utility 0 to each of the manipulators, and all other candidates would give utility 1 to each of the manipulators. In this case the only sensible game-theoretic goal for the manipulators is to make the disliked candidate not win. This is exactly our notion of destructive manipulation.

   Second, at least for deterministic voting rules in the complete information setting, the easiness results transfer from constructive manipulation to any game-theoretic definitions of manipulation that would come down to determining whether the manipulators can make some candidate from a subset of candidates win. For example, one can consider a manipulation successful if it causes some candidate to win that is preferred by each one of the manipulators to the candidate who would win if the manipulators voted truthfully. As another example, one can consider a manipulation successful if it causes some candidate to win that gives a higher sum of utilities to the manipulators than the candidate who would win if the manipulators voted truthfully. (This definition is especially pertinent if the manipulators can use side payments or some other form of restitution to divide the gains among themselves.) Now, we can solve the problem of determining whether some candidate in a given subset can be made to win simply by determining, for each candidate in the subset in turn, whether that candidate can be made to win. So the complexity exceeds that of constructive manipulation by at most a factor equal to the number of candidates (*i.e.*, a constant).

   Third, the complexity of destructive manipulation is directly related to the complexity of de-

termining whether enough votes have been elicited to determine the outcome of the election. Specifically, enough votes have been elicited if there is no way to make the conjectured winner not win by casting the yet unknown votes [Conitzer and Sandholm, 2002c].

In summary, we focus on coalitional weighted manipulation (CW-MANIPULATION), in the complete information setting. We study both constructive and destructive manipulation. Formally:

Definition 41 (CONSTRUCTIVE COALITIONAL WEIGHTED (CW) MANIPULATION) *We are given a set of weighted votes S (the nonmanipulators' votes), the weights for a set of votes T which are still open (the manipulators' votes), and a preferred candidate p. For deterministic rules, we are asked whether there is a way to cast the votes in T so that p wins the election. For randomized rules, we are additionally given a distribution over instantiations of the voting rule, and a number r, where $0 \leq r \leq 1$. We are asked whether there is a way to cast the votes in T so that p wins with probability greater than r.*

Definition 42 (DESTRUCTIVE COALITIONAL WEIGHTED (CW) MANIPULATION) *We are given a set of weighted votes S (the nonmanipulators' votes), the weights for a set of votes T which are still open (the manipulators' votes), and a disliked candidate h. For deterministic rules, we are asked whether there is a way to cast the votes in T so that h does* not *win the election. For randomized rules, we are additionally given a distribution over instantiations of the voting rule, and a number r, where $0 \leq r \leq 1$. We are asked whether there is a way to cast the votes in T so that h wins with probability* less *than r.*

For deterministic rules, we do not consider a manipulation successful if it leaves candidate p or h tied for the win.[9] One way of viewing this is as follows. If ties are broken randomly, then technically, we are dealing with a randomized rule. Then, we can set r so that a tie for the win does not give p enough probability of winning (for example, $r = 2/3$), or so that a tie gives h too much probability of winning (for example, $r = 1/(m+1)$).

Before we start studying these problems, we first complete our justification for requiring hardness even with few candidates. We do so by giving an algorithm for an individual voter to manipulate the STV rule that is exponential only in the number of candidates, and scales to reasonably large numbers of candidates. This is the subject of the next subsection (which can be skipped without affecting the reader's ability to comprehend the rest of this section).

8.3.2 Algorithm for individually manipulating the STV rule

When the number of candidates is unbounded, the STV rule is known to be NP-complete to constructively manipulate, even by a single manipulator when the votes are not weighted [Bartholdi and Orlin, 1991]. In this subsection we present an algorithm for manipulating STV as a single voter, when the votes of the others are known. Votes may be weighted.

To study the complexity fundamental to manipulating STV, rather than complexities introduced by tie-breaking rules, for this algorithm we make the following assumption. We assume that the STV rule uses some deterministic method for breaking ties (when choosing the loser to be eliminated at

[9] Our proofs do not depend on this specification, with the exception of Theorem 74.

the end of a round), where the tie-breaking does not depend on aspects of the votes that the STV rule has not considered so far (such as who has been ranked *lowest* by the largest number of voters). However, the tie-breaking can depend on the number of the round, candidates still in the running, *etc.* In the algorithm, the tie-breaking rule is used in the $\arg\min$ function.

The algorithm simulates the various ways in which the elimination of candidates may proceed given various votes by the manipulator. It follows the principle of least commitment in deciding which manipulative votes to consider. It returns the set of candidates that win for some vote by the manipulator. (It is easy to extend the algorithm so that it also provides a vote to effect such a victory.) Again, let there be n voters, where we index the manipulator n. Let C be the set of remaining candidates. Let v_i be the vote of voter i ($1 \leq i < n$) and let w_i be the weight of voter i ($1 \leq i \leq n$). Let s_j be the weight of the voters that rank candidate j first among the remaining candidates.

Some stages of the simulation can be reached only if the manipulator has a certain candidate f ranked first among the remaining candidates. At such stages, we will know precisely how the elimination will proceed, until f is eliminated and the manipulator's vote is freed up again. We say that $f = 0$ when there is no constraint on how the manipulator ranks the remaining candidates in the current stage of the simulation.

The function TRANSFERVOTES takes as input a candidate c, the remaining set of candidates C, the vector $(s_1, \ldots, s_m)$, and the votes and weights. It returns what would be the new vector $(s_1, \ldots, s_m)$ if c were eliminated in this round.

Now, we are ready to present the manipulation algorithm, which, when called with the original set of candidates C and $f = 0$, returns the set of all candidates that will win for some vote by the manipulator.

MANIPULATE$(C, (s_1, \ldots, s_m), (v_1, \ldots, v_{n-1}), (w_1, \ldots, w_n), f)$
if $C = \{c\}$
// we have eliminated all but a single candidate
    return $\{c\}$
else if $(f \neq 0)$
// the manipulator's vote is already committed to a candidate at this stage
    $c \leftarrow \arg\min_{j \in C}(s_j)$
    $(t_1, \ldots, t_m) \leftarrow$ TRANSFERVOTES$(c, C, (s_1, \ldots, s_m), (v_1, \ldots, v_{n-1}),$
        $(w_1, \ldots, w_n))$
    if $c = f$
    // the manipulator's vote is freed up
        return MANIPULATE$(C - \{c\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
            $(w_1, \ldots, w_n), 0)$
    else
    // the manipulator's vote remains committed
        return MANIPULATE$(C - \{c\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
            $(w_1, \ldots, w_n), f)$
else
// the manipulator's vote is not committed at this stage
    $c_1 \leftarrow \arg\min_{j \in C}(s_j)$
    // which candidate is losing before the manipulator assigns his vote?
    $s_{c_1} \leftarrow s_{c_1} + w_n$
    $c_2 \leftarrow \arg\min_{j \in C}(s_j)$
    // which candidate loses if the manipulator supports c_1?
    $(t_1, \ldots, t_m) \leftarrow$ TRANSFERVOTES$(c_1, C, (s_1, \ldots, s_m), (v_1, \ldots, v_{n-1}),$
        $(w_1, \ldots, w_n))$
    if $c_1 = c_2$
    // the manipulator cannot rescue c_1 at this stage
        return MANIPULATE$(C - \{c_1\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
            $(w_1, \ldots, w_n), 0)$
    else
    // the manipulator can choose to rescue c_1 at this stage
        $S_1 \leftarrow$ MANIPULATE$(C - \{c_1\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
            $(w_1, \ldots, w_n), 0)$

// case I: do not rescue c_1
$(t_1, \ldots, t_m) \leftarrow$ TRANSFERVOTES$(c_2, C, (s_1, \ldots, s_m), (v_1, \ldots, v_{n-1}),$
$(w_1, \ldots, w_n))$
$S_2 \leftarrow$ MANIPULATE$(C - \{c_2\}, (t_1, \ldots, t_m), (v_1, \ldots, v_{n-1}),$
$(w_1, \ldots, w_n), c_1)$
// case II: do rescue c_1
return $S_1 \cup S_2$

We now analyze how many recursive calls we will make to this algorithm. Let $T(k)$ be the maximal number of calls we need for a set of remaining candidates of size k. Let $T_0(k)$ be the maximal number of calls we need when the first call has $f \neq 0$. Then we have the following recurrences:

$$T(k) \leq 1 + T(k-1) + T_0(k-1) \tag{8.1}$$
$$T_0(k) \leq 1 + T(k-1) \tag{8.2}$$

Combining the two we get

$$T(k) \leq 2 + T(k-1) + T(k-2) \tag{8.3}$$

The asymptotic bound that we derive from this recurrence is indeed tight for the running time of MANIPULATE, as the following example shows. In the example, candidates are eliminated sequentially, but the manipulator can postpone the elimination of any candidate for exactly one round. Let the candidates be $(c_1, \ldots, c_m)$. For candidate i, let there be exactly i voters that rank it first, for a total of $\sum_{i=1}^{m} i = \frac{(m+1)m}{2}$ voters other than the manipulator, of weight 1 each. All the voters other than the manipulator that do not rank candidate m first, rank it second. The manipulator has weight $1 + \epsilon$.[10] We claim that the arguments passed to MANIPULATE always satisfy one of the following two properties:

- (1) $C = \{c_i, c_{i+1}, \ldots, c_m\}$ and $f = 0$.
- (2) $C = \{c_i, c_{i+2}, c_{i+3}, \ldots, c_m\}$ and $f = c_i$.

The initial call is of type (1). If the current call is of type (1), this will lead to two recursive calls: one of type (1) (the manipulator does not rescue candidate c_i), and one of type (2) (the manipulator does rescue c_i). (The exception is when $i \geq m - 1$ but this is irrelevant to the asymptotic analysis.) This makes the recurrence in Equation 8.1 tight. If the current call is of type (2), this will lead to one recursive call of type (1) because c_i gets eliminated in spite of the fact that the manipulator is ranking it first. This makes the recurrence in Equation 8.2 tight. Because the recurrences in Equations 8.1 and 8.2 are tight, the recurrence in Equation 8.3 is tight.

[10] Alternatively, we can assume unweighted votes and a tie-breaking mechanism that always breaks ties towards lower-indexed candidates, that is, it breaks a tie between c_i and c_{i+1} in favor of c_i.

The solution to the recurrence is $O((\frac{1+\sqrt{5}}{2})^k)$. Observing that one call to MANIPULATE (not counting the recursive calls to MANIPULATE, but counting the calls to TRANSFERVOTES) can be done in $O(n)$ time (assuming that $(s_1, \ldots, s_m)$ can be stored in an array), we have the following result.

Theorem 67 *The algorithm MANIPULATE runs in time $O(n(\frac{1+\sqrt{5}}{2})^m)$, where m is the number of candidates and n is the number of voters.*

So, MANIPULATE runs in $O(n \cdot 1.62^m)$ time. While this function is exponential in m (which is to be expected given that the problem is NP-complete), it is nevertheless not exceedingly large for realistic numbers of candidates. For instance, with 10 candidates, $(\frac{1+\sqrt{5}}{2})^{10} < 123$. Furthermore, on most instances, the algorithm is likely to terminate much faster than this, since we make two recursive calls only when the manipulator can rescue a candidate from elimination in a given round. In large elections where the manipulator's weight is relatively insignificant, it is unlikely that this would happen even more than once.

8.3.3 Complexity of weighted coalitional manipulation with few candidates

We are now ready to present our results on the hardness of coalitional manipulation when there are few candidates and the votes are weighted. We will study not only *whether* any given rule is hard to manipulate with a constant number of candidates, but also *how many* candidates are needed for the hardness to occur. This number is important for evaluating the relative manipulability of different voting rules (the lower this number, the less manipulable the rule). For each rule that we show is hard to manipulate with some constant number of candidates, we show this for the smallest number of candidates for which the hardness occurs, and we show that manipulation becomes easy if we reduce the number of candidates by one. (Once we have identified this transition point, it is easy to see that the manipulation problem remains hard for any greater number of candidates, and remains easy for any smaller number of candidates, for example by adding "dummy" candidates.)

Constructive Manipulation

We first present our results for constructive manipulation.

We begin by laying out some cases where constructive manipulation can be done in polynomial time. We start with some rules that are easy to manipulate constructively regardless of the number of candidates. For the plurality rule, showing this is straightforward:

Theorem 68 *For the* plurality *rule,* CONSTRUCTIVE CW-MANIPULATION *can be solved in polynomial time (for any number of candidates).*

Proof: The manipulators can simply check if p will win if all the manipulators vote for p. If not, they cannot make p win. ■

For the cup rule, the proof is a little more involved:

Theorem 69 *For the* cup *rule (given the assignment of candidates to leaves),* CONSTRUCTIVE CW-MANIPULATION *can be solved in polynomial time (for any number of candidates).*

Proof: We demonstrate a method for finding all the potential winners of the election. In the binary tree representing the schedule, we can consider each node to be a subelection, and compute the set of potential winners for each subelection. (In such a subelection, we may say that the voters only order the candidates in that subelection since the place of the other candidates in the order is irrelevant.) Say a candidate *can* obtain a particular result in the election if it does so for some coalitional vote. The key claim to the proof, then, is the following: a candidate can win a subelection if and only if it can win one of its children, *and* it can defeat one of the potential winners of the sibling child in a pairwise election. It is easy to see that the condition is necessary. To show that it is sufficient, let p be a candidate satisfying the condition by being able to defeat h, a potential winner of the other child (or *half*). Consider a coalitional vote that makes p win its half, and another one that makes h win its half. We now let each coalitional voter vote as follows: it ranks all the candidates in p's half above all those in h's half; the rest of the order is the same as in the votes that make p and h win their halves. Clearly, this will make p and h the finalists. Also, p will win the pairwise election against h since it is always ranked above h by the colluders; and as we know that there is some coalitional vote that makes p defeat h pairwise, this one must have the same result. The obvious recursive algorithm has running time $O(m^3n)$ according to the Master Theorem [Cormen *et al.*, 1990]. ■

The remaining easiness results that we show in this subsubsection only show easiness up to a certain number of candidates. For each of these results, we later show that adding one more candidate causes the problem to become NP-complete.

As a first observation, when there are only two candidates, all the rules are equivalent to the plurality rule, and hence both types of manipulation (constructive and destructive) are in P for all of the rules. However, some rules are still easy to manipulate constructively with more than 2 candidates. In each of the following cases, we prove easiness of manipulation by demonstrating that if there exists a successful manipulation, there also exists one *where all the manipulators vote the same way*. All such ways of voting can be easily enumerated: because the number of candidates is constant, the number of different orderings of the candidates is constant. Also, each way of voting is easy to evaluate in these rules.

Theorem 70 *If the* Copeland *rule with 3 candidates has a* CONSTRUCTIVE CW-MANIPULATION*, then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: Let the 3 candidates be p, a, and b. We are given the nonmanipulators' votes S, and the weights for the manipulators' votes T. Let the total vote weight in T be K.

For a set of weighted votes V and two candidates x, y, we denote by $N_V(x, y)$ the cumulated weights of the votes in V ranking x prior to y, and we let $D_V(x, y) = N_V(x, y) - N_V(y, x)$. Let us consider the following four cases which cover all possible situations:
Case 1: $K > D_S(a, p)$ and $K > D_S(b, p)$.

In this case, *any* configuration of votes for T such that p is ranked first for all votes makes p win the election.
Case 2: $K > D_S(a, p)$ and $K = D_S(b, p)$.

It can easily be shown that it is harmless to assume that all votes in T rank p first. Therefore, what remains to be done in order to have p win is to find who in T should vote (p, a, b) and who should vote (p, b, a). What we know so far (before knowing how the votes in T will split between these two profiles) is: (1) $D_{S\cup T}(p, a) = K - D_S(p, a) > 0$ and (2) $D_{S\cup T}(p, b) = K - D_S(p, b) = 0$. (1) makes p get +1 and a get -1 while (2) makes both p and b get 0. Therefore, the partial Copeland scores (not taking account of the a vs. b pairwise election), are $+1$ for p (which will not change after taking account of the $a-b$ pairwise election), -1 for a and 0 for b; hence, the only way for p to win (with certainty) is to avoid b getting a point in the pairwise election against a, i.e., to ensure that $D_{S\cup T}(a, b) \geq 0$. It can easily be shown that this is possible if and only if $K \geq D_S(b, a)$. Therefore, we have found that there exists a successful manipulation for p iff $K \geq D_S(b, a)$, and in this case a successful manipulation is the one where all voters in the coalition vote (p, a, b).
Case 3: $K = D_S(a, p)$ and $K > D_S(b, p)$.

This is similar to Case 2, switching the roles of a and b; the condition then is $K \geq D_S(a, b)$ and the successful manipulation is the one where all vote (p, b, a).
Case 4: $K < D_S(a, p)$ or $K < D_S(b, p)$ or ($K \leq D_S(a, p)$ and $K \leq D_S(b, p)$).

Here, whatever the votes in T, the Copeland score of p is smaller than or equal to 0 and therefore p cannot be guaranteed to win, so there is no successful manipulation. Thus, in every case, either there is no successful manipulation, or there is a successful manipulation where all manipulators vote identically. ■

Theorem 71 *If the* maximin *rule with 3 candidates has a* CONSTRUCTIVE CW-MANIPULATION*, then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: Let the 3 candidates be p, a, and b. We are given the nonmanipulators' votes S, and the weights for the manipulators' votes T. Let the total vote weight in T be K. Again, it is easy to show that all the manipulators can rank p first without harm.

Let us denote by $P_{K1,K2}$ a vote configuration for T such that a subset T_1 of T, whose cumulated weight is K_1, votes (p, a, b) and $T_2 = T \setminus T_1$, whose cumulated weight is K_2 (with $K_1 + K_2 = K$), votes (p, b, a). Now all that remains to show is the following: if p wins with the votes in T being $P_{K1,K2}$ then either p wins with the votes in T being $P_{K,0}$ or p wins with the votes in T being $P_{0,K}$. Let us consider these two cases for the outcome of the whole election (including the votes in T):
Case 1: the uniquely worst pairwise election for a is against b, and the uniquely worst pairwise election for b is against a. One of a and b must have got at least half the vote weight in the pairwise election against the other (say, without loss of generality, a) and therefore have a maximin score of at least half the vote weight. Since a did even better against p, p received less than half the vote weight in their pairwise election and therefore p does not win.
Case 2: One of a and b (say, without loss of generality, a) does at least as badly against p as against the other (so, a's worst opponent is p). Then all the voters in the coalition might as well vote (p, a, b), because this will change neither a's score nor p's score, and might decrease (but not increase) b's score. ■

Theorem 72 *If the* randomized cup *rule with 6 candidates has a* CONSTRUCTIVE CW-MANIPULATION, *then it has a* CONSTRUCTIVE CW-MANIPULATION *where all of the manipulators vote identically. (This holds regardless of which balanced tree is chosen.) Therefore,* CONSTRUCTIVE CW-MANIPULATION *is in P.*

Proof: We show how to transform a successful manipulation into a successful manipulation where all the manipulators cast the same vote, in a number of steps. Observe that swapping two candidates that are ranked directly after one another in a vote can only affect the outcome of their pairwise election, and not those of the others. (Throughout the proof, when we talk about swapping two candidates in a vote, this only means swapping two candidates *ranked directly behind each other*.)

If p is ranked directly behind another candidate c, the only effect that swapping them can have is to make p the winner of their pairwise election where it was not before; clearly this can never hurt p's chances of winning. Repeated application of this gives us a successful manipulation *where all the manipulators rank p at the top*.

We now divide the other candidates into two sets: B is the set of candidates that defeat p in their pairwise election, G is that of candidates that are defeated by p. We now claim that when a candidate $g \in G$ is ranked directly behind a candidate $b \in B$, swapping them cannot hurt p's chances of winning. This is because of the following reason. Again, the only effect that the swap can have is to make g the winner of its pairwise election with b, where it was not before. We show that for any schedule (assignment of candidates to leaves), if p wins when b defeats g, then p also wins in the case where this pairwise election is changed to make g defeat b (but nothing else is changed). For any schedule where g and b do not meet this is obvious. If b and g face each other in the final, p of course does not win. If b and g face each other in a semifinal, and b defeats g, then again p cannot win because it cannot defeat b in the final. So the only case left to check is a schedule where b and g meet in a quarterfinal (because the tree is balanced). If p wins with this schedule when b defeats g, that means that b is defeated by some other $g' \in G$ in the semifinal (if b wins the semifinal, p could never defeat it in the final; if b loses to some $b' \in B$, p could never defeat b' in the final; and p itself cannot defeat b in the semifinal either), and that p defeats this g' in the final. Then if we change the winner of the quarterfinal to g, regardless of whether g or g' wins the semifinal, p will win the final. Thus the claim is proven. Repeated application of this gives a successful manipulation *where all the manipulators rank p at the top, and all candidates in G above all candidates in B*.

All that remains is to get the manipulators to agree on the order of the candidates within G and the order of the candidates within B. We will show how to do this for G; the case of B is entirely analogous. If there are 0 or 1 candidates in G, there is only one order for these candidates. If there are 2 candidates in G, then swapping them whenever the winner of their pairwise election is ranked below the loser does not affect the outcome of their pairwise election and hence nothing at all.

If there are 3 candidates in G, that means there are only 2 in B. Say $B = \{b_1, b_2\}$. Divide the candidates in G into $G_B, G_{\{b_1\}}, G_{\{b_2\}}, G_{\{\}}$, where a candidate in G_S defeats all candidates in S but none in $B - S$. We first claim that it never hurts to swap when a candidate $g_B \in G_B$ is ranked directly below another candidate $g \in G$. Again we do so by showing that with any schedule where p wins when g defeats g_B, p also wins when g_B defeats g (but everything else is unchanged). If g and g_B never meet this is obvious; if g and g_B meet in a semifinal or the final, it does not matter to p which one wins. If g defeats g_B in a quarterfinal and p wins the election, then one of the three

following cases applies: 1. p defeats g in the semifinal, 2. g defeats some $b \in B$ in the semifinal and is defeated by p in the final, 3. g faces the third candidate $g_3 \in G$ in the semifinal, and p wins the final agaist the winner of this. Now, if g_B instead had defeated g, then in case 1 p defeats g_B in the semifinal and goes on to win the final as before; in case 2, g_B defeats b as well in the semifinal and then loses to p in the final; and in case 3, p faces either g_B or g_3 in the final and wins. Repeated application of this gets all the elements in G_B ranked above all the other elements in G in all the manipulators' votes, and this rule also allows us to get the elements in G_B ordered among themselves in the same way in all the manipulators' votes. Similarly, we can show that we can always swap the elements in $G_{\{\}}$ downwards, so that all the elements of this set are ranked below all other elements in G, and ordered among themselves in a unique manner across all the votes.

Now, if indeed there is some element in G_B or $G_{\{\}}$, then there are at most two candidates left in G_B whose order might differ across manipulators' votes. As in the case of 2 candidates, we can simply always swap the winner of their pairwise election up without changing anything, and we are done. On the other hand suppose there is no element in G_B or $G_{\{\}}$, so that all the remaining elements are in $G_{\{b_1\}}$ or $G_{\{b_2\}}$. We claim that either it does not hurt to swap all the candidates from $G_{\{b_1\}}$ below all those of $G_{\{b_2\}}$, or vice versa. In the case where either $G_{\{b_1\}}$ or $G_{\{b_2\}}$ is empty, this claim is vacuous: so suppose without loss of generality that $G_{\{b_1\}}$ has two elements g_1 and g_2, and $G_{\{b_2\}}$ one element, g_3. Now suppose the contrary, that is, that always swapping g_3 above g_1 and g_2 decreases p's chances of winning the election, but that always swapping g_3 below g_1 and g_2 also decreases p's chances of winning the election. It follows that always swapping g_3 above g_1 and g_2 causes g_3 to win both pairwise elections, and that always swapping g_3 below g_1 and g_2 causes g_3 to lose both pairwise elections. (For otherwise, the manipulators cannot change the winner of one of these pairwise elections, and because the other pairwise election must have a winner in the current state, either always swapping g_3 up or always swapping g_3 down will not affect any pairwise election at all, and hence the probability of p winning would remain unchanged.) Hence in the current state g_3 must be winning exactly one of these pairwise elections (without loss of generality, the one against g_1). Now, because always swapping g_3 below g_1 and g_2 only has the effect of making it lose against g_1 as well, and because this reduces the probability of p winning, it follows that the number of schedules where p wins now is strictly greater than the number of schedules where p wins if g_3 lost to g_1 in its pairwise election but everything else remained the same. Thus, the number of schedules where p wins now but would not win if g_3 lost to g_1 in their pairwise election (call this set of schedules $L(g_1)$), is strictly greater than the number of schedules where p does not win now but would win if g_3 lost to g_1 in their pairwise election (call this set of schedules $W(g_1)$). Similarly we can show that the number of schedules where p wins now but would not win if g_3 defeated g_2 in their pairwise election (call this set of schedules $W(g_2)$), is strictly greater than the number of schedules where p does not win now but would win if g_3 defeated g_2 in their pairwise election (call this set of schedules $L(g_2)$). So, $|W(g_1)| < |L(g_1)|$, and $|W(g_2)| > |L(g_2)|$. We now derive the desired contradiction by showing $W(g_1)| \geq |W(g_2)|$ and $|L(g_1)| \leq |L(g_2)|$. Consider the mapping f from schedules to schedules that simply swaps the position of g_1 and g_2 in the schedule. This mapping is one-to-one. We now claim that if $\sigma \in W(g_2)$, then $f(\sigma) \in W(g_1)$, thereby demonstrating $|W(g_1)| \geq |W(g_2)|$; the case for $|L(g_1)| \leq |L(g_2)|$ is similar. $\sigma \in W(g_2)$ means that p wins in σ now but would not win if g_3 defeated g_2 in their pairwise election. It is straightforward to check that this only happens if g_2 and g_3 meet in a quarterfinal, and the winner

goes on to meet b_1 in the semifinal (whom g_2 would defeat but g_3 would not), while p goes on to the final in the other half of the cup. Now we must show $f(\sigma) \in W(g_1)$. In $f(\sigma)$, g_1 and g_3 face each other in a quarterfinal. The semifinal after this quarterfinal will certainly have b_1 in it (if b_1 plays a quarterfinal before this, b_1 must have the same opponent in this quarterfinal as in σ, because b_1 cannot face g_1 in the quarterfinal in σ and still move on to the semifinal in σ; hence b_1 will defeat its quarterfinal opponent in $f(\sigma)$ as well). Also, the final will certainly have p in it (even if g_1 was in p's half, since the sets of candidates within p's half that g_1 and g_2 defeat are identical, this half will proceed exactly as in σ). Now, in the current state of the votes, g_3 will defeat g_1 in the quarterfinal, upon which b_1 will defeat g_3 in the semifinal and p in the final. On the other hand, if g_1 defeated g_3, it would defeat b_1 in the semifinal and p would win the final against g_1, and hence win the entire election. It follows that $f(\sigma) \in W(g_1)$, as was to be shown. So either it does not hurt to swap all the candidates from $G_{\{b_1\}}$ below all those of $G_{\{b_2\}}$, or vice versa. It remains to be shown that the manipulators' rankings of the candidates within $G_{\{b_1\}}$ and within $G_{\{b_2\}}$ can be made to coincide. Given (without loss of generality) $g_1, g_2 \in G_{\{b_1\}}$ it is straightforward to check that it does not matter to p which of them would win if they faced each other in the final or semifinal. In case they face each other in a quarterfinal, then if p is in the same half of the cup it does not matter which of g_1 and g_2 wins the quarterfinal, because p (if it reaches the semifinal) would defeat either one. If p is in the other half, the only thing that matters is whether this half's finalist is in B or in G. Thus, if the winner of the quarterfinal between g_1 and g_2 faces the last remaining candidate from G the finalist will certainly be in G, so who won the quarterfinal does not matter; on the other hand, if the winner of the quarterfinal between g_1 and g_2 faces a candidate in B, then again it does not matter who wins the quarterfinal, because g_1 and g_2 defeat the same set of candidates from B. It follows that it is irrelevant to p's chances of winning what the outcome of a pairwise election between candidates in $G_{\{b_1\}}$ or between candidates in $G_{\{b_2\}}$ is, so we can order them among themselves in whichever way we like. Hence we can make the manipulators' votes coincide on the order of the 3 candidates in G.

If there are 4 candidates in G, that means there is only one in B. Hence we can partition G into G_B and $G_{\{\}}$. With techniques similar to the case of 3 candidates in G, we can show that we can always swap candidates in G_B above those in $G_{\{\}}$; and we can show that a vote's order of the candidates within G_B (or $G_{\{\}}$) is irrelevant. Hence we can make the manipulators' votes coincide on the order of the 4 candidates in G.

Finally, if there are 5 candidates in G, then p defeats all other candidates in pairwise elections, so p is guaranteed to win regardless of how the candidates in G are ranked. ■

We are now ready to prove hardness results that match the bounds given by the easiness results above. (That is, for every rule which we showed is easy to manipulate with up to l candidates, we now show that it is hard to manipulate with $l+1$ candidates.) In many of the proofs of NP-hardness, we use a reduction from the PARTITION problem, which is NP-complete [Karp, 1972]:

Definition 43 PARTITION. *We are given a set of integers $\{k_i\}_{1 \leq i \leq t}$ (possibly with multiplicities) summing to $2K$, and are asked whether a subset of these integers sums to K.*

Theorem 73 *For any scoring rule other than the* plurality *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: First, note that when there are only 3 candidates, a positional scoring rule is defined by

a vector of integers $\vec{\alpha} = \langle \alpha_1, \alpha_2, \alpha_3 \rangle$ such that $\alpha_1 \geq \alpha_2 \geq \alpha_3$. Without loss of generality, we can assume that $\alpha_3 = 0$ (since translating the α_i's by a constant has no effect on the outcome of the election). We can also assume that $\alpha_2 \geq 2$. (If $\alpha_2 = 0$ then the rule is either meaningless (if $\alpha_1 = 0$) or equivalent to the plurality rule, so we can assume $\alpha_2 > 0$. Then, we can scale the α_i appropriately, which will not affect the rule.) Showing the problem is in NP is easy. To show NP-hardness, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE MANIPULATION instance. The 3 candidates are a, b, and p. In S there are $(2\alpha_1 - \alpha_2)K - 1$ voters voting (a, b, p) and $(2\alpha_1 - \alpha_2)K - 1$ voters voting (b, a, p). In T, for each k_i there is a vote of weight $(\alpha_1 + \alpha_2)k_i$. Suppose there is a partition of the k_i. Then, let the votes in T corresponding to the one half of the partition be (p, a, b) and the votes corresponding to the other half be (p, b, a). Then the score of p is $2(\alpha_1 + \alpha_2)\alpha_1 K$ while the scores of both a and b are $(\alpha_1 + \alpha_2)(2\alpha_1 K - 1)$, therefore p is the winner and there is a manipulation. Conversely, suppose there exists a manipulation. Then, since scoring procedures satisfy monotonicity, we can assume without loss of generality that the voters in the coalition T rank p first. Let x (resp. y) be the total weight of voters in T of the voters who vote (p, a, b) (resp. (p, b, a)). Note that we have $x + y = 2K$. Then the score of p is $2(\alpha_1 + \alpha_2)\alpha_1 K$, the score of a is $(\alpha_1 + \alpha_2)((2\alpha_1 - \alpha_2)K - 1 + x\alpha_2)$, and the score of b is $(\alpha_1 + \alpha_2)((2\alpha_1 - \alpha_2)K - 1 + y\alpha_2)$. Since p is the winner, its score must be at least that of a, which is equivalent to $x\alpha_2 \leq K\alpha_2 + 1$. Because $\alpha_2 > 0$, the latter condition is equivalent to $x \leq K + \frac{1}{\alpha_2}$, which is equivalent to $x \leq K$ (because $\alpha_2 \geq 2$). Similarly, we get $y \leq K$, which together with $x + y = 2K$ enables us to conclude that $x = y = K$, so there exists a partition. ■

Corollary 4 *For the* veto *and* Borda *rules,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

(On a historical note, we actually proved Corollary 4 before we discovered its generalization to Theorem 73; the generalization was independently discovered by us, Hemaspaandra and Hemaspaandra [2005], and Procaccia and Rosenschein [2006].)

Theorem 74 *For the* Copeland *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 4 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 4 candidates, a, b, c and p. In S there are $2K + 2$ voters voting (p, a, b, c), $2K + 2$ voting (c, p, b, a), $K + 1$ voting (a, b, c, p), and $K + 1$ voting (b, a, c, p). In T, for every k_i there is a vote of weight k_i. We show the instances are equivalent. First, every pairwise election is already determined without T, except for the one between a and b. p defeats a and b; a and b each defeat c; c defeats p. If there is a winner in the pairwise election between a and b, that winner will tie with p. So p wins the Copeland election if and only if a and b tie in their pairwise election. But, after the votes in S alone, a and b are tied. Thus, the votes in T maintain this tie if and only if the combined weight of the votes in T preferring a to b is the same as the combined weight of the votes in T preferring b to a. This can happen if and only if there is a partition. ■

Theorem 75 *For the* maximin *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 4 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 4 candidates, a, b, c and p. In S there are $7K-1$ voters voting (a,b,c,p), $7K-1$ voting (b,c,a,p), $4K-1$ voting (c,a,b,p), and $5K$ voting (p,c,a,b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent. Suppose there is a partition. Then, let the votes in T corresponding to the k_i in one half of the partition vote (p,a,b,c), and let the other ones vote (p,b,c,a). Then, p does equally well in each pairwise election: it always gets $9K$ pairwise points. a's worst pairwise election is against c, getting $9K-1$. b's worst is against a, getting $9K-1$. Finally c's worst is against b, getting $9K-1$. Hence, p wins the election. So there is a manipulation. Conversely, suppose there is a manipulation. Then, since moving p to the top of each vote in T will never hurt p in this rule, there must exist a manipulation in which all the votes in T put p at the top, and p thus gets $9K$ as its worst pairwise score. Also, the votes in T cannot change which each other candidate's worst pairwise election is: a's worst is against c, b's worst is against a, and c's worst is against b. Since c already has $9K-1$ points in its pairwise election against b, no vote in T can put c ahead of b. Additionally, if any vote in T puts a right above c, swapping their positions has no effect other than to decrease a's final score, so we may also assume this does not occur. Similarly we can show it safe to also assume no vote in T puts b right above a. Combining all of this, we may assume that all the votes in T vote either (p,a,b,c) or (p,b,c,a). Since a already has $7K-1$ points in the pairwise election against c, the votes in T of the first kind can have a total weight of at most $2K$; hence the corresponding k_i can sum to at most K. The same holds for the k_i corresponding to the second kind of vote on the basis of b's score. Hence, in both cases, they must sum to exactly K. But then, this is a partition. ■

Theorem 76 *For the* STV *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 3 candidates, a, b and p. In S there are $6K-1$ voters voting (b,p,a), $4K$ voting (a,b,p), and $4K$ voting (p,a,b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent. Suppose there is a partition. Then, let the votes in T corresponding to the k_i in one half of the partition vote (a,p,b), and let the other ones vote (p,a,b). Then in the first round, b has $6K-1$ points, a has $6K$, and p has $6K$. So b drops out; all its votes transfer to p, so that p wins the final round. So there is a manipulation. Conversely, suppose there is a manipulation. Clearly, p cannot drop out in the first round; but also, a cannot drop out in the first round, since all its votes in S would transfer to b, and b would have at least $10K-1$ points in the final round, enough to guarantee it victory. So, b must drop out in the first round. Hence, from the votes in T, both a and c must get at least $2K$ weight that puts them in the top spot. The corresponding k_i in either case must thus sum to at least K. Hence, in both cases, they must sum to exactly K. But then, this is a

partition. ■

This also allows us to show that constructively manipulating the plurality with runoff rule is hard:

Theorem 77 *For the* plurality with runoff *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 3 candidates.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we observe that with 3 candidates, the plurality with runoff rule coincides with the STV rule, and CONSTRUCTIVE MANIPULATION for STV with 3 candidates is NP-hard. ■

Theorem 78 *For the* randomized cup *rule,* CONSTRUCTIVE CW-MANIPULATION *is NP-complete for 7 candidates.*

Proof: The problem is in NP because for any vector of votes, we can check for every one of the possible schedules for the cup whether p wins (because the number of candidates is constant, so is the number of possible schedules). To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following CONSTRUCTIVE CW-MANIPULATION instance. There are 7 candidates, a, b, c, d, e, f, and p. In T, for every k_i there is a vote of weight $2k_i$. Let $W = 4K$. Let the votes in S be as follows: there are $\frac{5}{2}W$ votes (d, e, c, p, f, a, b), $\frac{1}{2}W$ votes (d, e, c, p, f, b, a), $\frac{5}{2}W$ votes (f, d, b, p, e, c, a), $\frac{1}{2}W$ votes (f, d, b, p, e, a, c), $\frac{5}{2}W$ votes (e, f, a, p, d, b, c), $\frac{1}{2}W$ votes (e, f, a, p, d, c, b), $2W$ votes (p, a, c, b, d, e, f), W votes (p, c, b, a, f, d, e), W votes (c, b, a, f, d, e, p), $2W$ votes (b, a, c, e, f, d, p), and $4K - 1$ votes that we will specify shortly. Since it takes only $8\frac{1}{2}W$ votes to win a pairwise election, it follows that the outcomes of the following pairwise elections are already determined: p defeats each of a, b, c (it has $9W$ votes in each of these pairwise elections from the given votes); each of d, e, f defeats p ($9W$ in each case); d defeats e, e defeats f, f defeats d ($10W$ in each case); a defeats d, b defeats e, c defeats f ($9W$ in each case); d defeats b and c, e defeats a and c, f defeats a and b ($9W$ in each case). So, the only pairwise elections left to be determined are the ones between a, b and c. We now specify the remaining $8K - 1$ votes in S: there are $2K - 1$ votes (c, b, a, p, d, e, f) and $2K - 1$ votes (b, a, c, p, d, e, f), and 1 vote (b, c, a, p, d, e, f). As a result, given all the votes in S, b is $4K - 1$ votes ahead of a in their pairwise election, b is 1 vote ahead of c, and c is 1 vote ahead of a. We claim that the votes in T can be cast so as to make a defeat b, b defeat c, and c defeat a, if and only if a partition of the k_i exists. If a partition exists, let the votes corresponding to one half of the partition be (a, b, c, p, d, e, f), and those corresponding to the other half be (c, a, b, p, d, e, f); this is easily verified to yield the desired result. Conversely, suppose there is a way to cast the votes in T so as to yield the desired result. Then, since each vote in T has even weight, all the votes in T rank a above b; at least half the vote weight ranks b above c; and at least half the vote weight ranks c above a. Since, if a is ranked above b, it is impossible to have c be ranked simultaneously both below b and above a, it follows that precisely half the vote weight ranks b above c (and the other half, c above a); and hence, we have a partition. To complete the proof of the theorem, we claim that making a defeat b, b defeat c, and c defeat a strictly maximizes the probability that p wins. From this claim it follows that if we set r to a number slightly smaller than

the probability that p wins *if* a defeats b, b defeats c, and c defeats a, then it is possible for the votes in T to make p win with probability at least r, if and only if there is a partition of the k_i. To prove the claim, we do a careful case-by-case analysis on the structure of the cup.

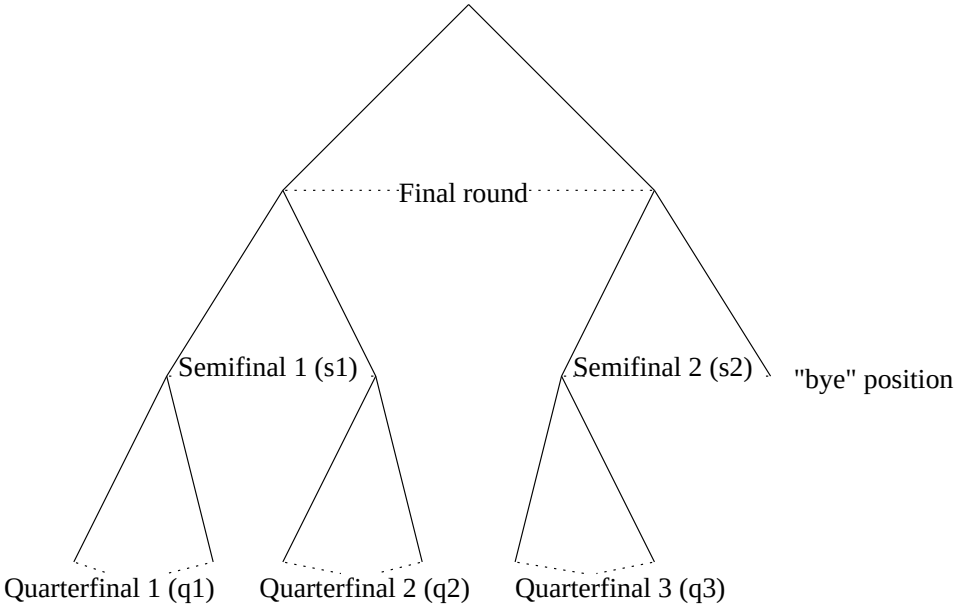

Figure 8.1: *The cup used in the proof.*

For each situation in which two of a, b and c face each other in a round, we analyze whose winning would be most favorable to p. If p does not get the "bye" position, it can only win by facing each of a, b and c in some round, which is impossible if any two of those ever face each other; so in this case the results of the pairwise elections between them are irrelevant as far as p's chances of winning are concerned. So let us assume p gets the bye position. If two of a, b and c face each other in s_1, then the outcome of this round is irrelevant as p would defeat either one in the final. If two of a, b and c face each other in q_3, then the outcome of this round is irrelevant as p would defeat either one in s_2. Now suppose a and b face each other in q_1. Then the only way for p to make it to the final is if c faces f in q_3, so let us assume this happens. Then, d and e face each other in q_2, which confrontation d will win. If a defeats b in q_1, it will win s_1 against d, and p will defeat it in the final. On the other hand, if b defeats a in q_1, it will lose s_1 against d, and d will defeat p in the final.. It follows that in this case, if we want p to win, we would (strictly) prefer a to defeat b in their pairwise election. Symmetrically, we also prefer a to defeat b if they face each other in q_2. Since we have analyzed all possibilities where a may face b, and in these is always favorable to p for a to defeat b, sometimes strictly so; it follows that it is strictly favorable to p's chances of winning if a defeats b in their pairwise election. Analogously (or by symmetry), it can be shown that it is strictly favorable to p's chances of winning if b defeats c, and c defeats b in their pairwise elections. Hence achieving these pairwise results simultaneously strictly maximizes p's chance of winning the election. ■

Recall that the cup rule (without randomization) is easy to manipulate constructively for any number of candidates. Thus, the previous result shows that *randomizing over instantiations of the rules* (such as schedules of a cup) *can be used to make manipulation hard*.

Destructive Manipulation

We now present our results for destructive manipulation.

We begin by laying out some cases where destructive manipulation can be done in polynomial time. It is easy to see that destructive manipulation can never be harder than constructive manipulation (except by a factor m) because in order to solve the former, we may simply solve the latter once for each candidate besides h. Thus, we immediately know that the plurality and cup rules can be destructively manipulated in polynomial time, for any number of candidates. Interestingly, for most of the other rules under study, destructive manipulation turns out to be drastically easier than constructive manipulation! The following theorem shows a sufficient condition for rules to be easy to manipulate destructively.

Theorem 79 *Consider any voting rule where each candidate receives a numerical score based on the votes, and the candidate with the highest score wins. Suppose that the score function is monotone, that is, if voter i changes its vote so that $\{b : a \succ_i^{old} b\} \subseteq \{b : a \succ_i^{new} b\}$ (here, $a \succ_i b$ means that voter i prefers a to b), a's score will not decrease. Finally, assume that the winner can be determined in polynomial time. Then for this rule, destructive manipulation can be done in polynomial time.*

Proof: Consider the following algorithm: for each candidate a besides h, we determine what the outcome of the election would be for the following coalitional vote. All the colluders place a at the top of their votes, h at the bottom, and order the other candidates in whichever way. We claim there is a vote for the colluders with which h does not win if and only if h does not win in one of these $m-1$ elections. The *if* part is trivial. For the *only if* part, suppose there is a coalitional vote that makes $a \neq h$ win the election. Then, in the coalitional vote we examine where a is always placed on top and h always at the bottom, by monotonicity, a's score cannot be lower (because for each manipulator i, $\{b : a >_i b\}$ is maximal) and h's cannot be higher (because for each manipulator i, $\{b : h >_i b\}$ is minimal) than in the successful coalitional vote. It follows that here, too, a's score is higher than h's, and hence h does not win the election. The algorithm is in P since we do $m-1$ winner determinations, and winner determination is in P. ■

Corollary 5 *Destructive manipulation can be done in polynomial time for the veto, Borda, Copeland, and maximin rules.*

Theorem 79 does not apply to the STV and plurality with runoff rules. We now show that destructive manipulation is in fact hard for these rules, even with only 3 candidates.

Theorem 80 *For the* STV *rule with 3 candidates,* DESTRUCTIVE CW-MANIPULATION *is NP-complete.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we reduce an arbitrary PARTITION instance to the following DESTRUCTIVE CW-MANIPULATION instance. The 3 candidates are a, b and h. In S there are $6K$ voters voting (a, h, b), $6K$ voters voting (b, h, a), and $8K - 1$ voters voting (h, a, b). In T, for every k_i there is a vote of weight $2k_i$. We show the instances are equivalent.

We first observe that h will not win if and only if it gets eliminated in the first round: for if it survives the first round, either a or b gets eliminated in the first round. Hence either all the votes in S that ranked a at the top or all those that ranked b at the top will transfer to h, leaving h with at least $14K - 1$ votes in the final round out of a total of $24K - 1$, so that h is guaranteed to win the final round.

Now, if a partition of the k_i exists, let the votes in T corresponding to one half of the partition vote (a, b, h), and let the other ones vote (b, a, h). Then in the first round, a and b each have $8K$ votes, and h only has $8K - 1$ votes, so that h gets eliminated. So there exists a manipulation.

On the other hand, if a manipulation exists, we know by the above that with this manipulation, h is eliminated in the first round. Hence at least $2K - 1$ of the vote weight in T ranks a at the top, and at least $2K - 1$ of the vote weight in T ranks b at the top. Let A be the set of all the k_i corresponding to votes in T ranking a at the top; then $\sum_{k_i \in A} k_i \geq K - \frac{1}{2}$, and since the k_i are integers this implies $\sum_{k_i \in A} k_i \geq K$. If we let B be the set of all the k_i corresponding to votes in T ranking b at the top, then similarly, $\sum_{k_i \in B} k_i \geq K$. Since A and B are disjoint, it follows that $\sum_{k_i \in A} k_i = \sum_{k_i \in B} k_i = K$. So there exists a partition. ∎

This result also allows us to establish the hardness of destructive manipulation in the plurality with runoff rule:

Theorem 81 *For the* plurality with runoff *rule with 3 candidates,* DESTRUCTIVE CW-MANIPULATION *is NP-complete.*

Proof: Showing the problem is in NP is easy. To show it is NP-hard, we observe that with 3 candidates, plurality with runoff coincides with STV, and DESTRUCTIVE MANIPULATION for STV with 3 candidates is NP-hard, as we proved in Theorem 80. ∎

8.3.4 Effect of uncertainty about others' votes

So far we have discussed the complexity of coalitional manipulation when the others' votes are known. We now show how those results can be related to the complexity of manipulation by an individual voter when only a *distribution* over the others' votes is known. If we allow for arbitrary distributions, we need to specify a probability for each possible combination of votes by the others, that is, exponentially many probabilities (even with just two candidates). It is impractical to specify so many probabilities.[11] Therefore, we should acknowledge that it is likely that the language used

[11] Furthermore, if the input is exponential in the number of voters, an algorithm that is exponential in the number of voters is not necessarily complex in the usual sense of input complexity.

for specifying these probabilities would not be fully expressive (or would at least not be very convenient for specifying complex distributions). We derive the complexity results of this subsection for extremely restricted probability distributions, which any reasonable language should allow for. Thus our results apply to any reasonable language. We only present results on constructive manipulations, but all results apply to the destructive cases as well and the proofs are analogous. We restrict our attention to deterministic rules.

Weighted voters

First we show that with weighted voters, in rules where coalitional manipulation is hard in the complete-information case, even evaluating a candidate's winning probability is hard when there is uncertainty about the votes (even when there is no manipulator).

Definition 44 (WEIGHTED EVALUATION) *We are given a weight for each voter, a distribution over all possible vectors of votes, a candidate p, and a number r, where $0 \leq r \leq 1$. We are asked whether the probability of p winning is greater than r.*

Theorem 82 *If* CONSTRUCTIVE CW-MANIPULATION *is NP-hard for a deterministic rule (even with k candidates), then* WEIGHTED EVALUATION *is also NP-hard for it (even with k candidates), even if $r = 0$, the votes are drawn independently, and only the following types of (marginal) distributions are allowed: 1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote.*

Proof: For the reduction from CONSTRUCTIVE CW-MANIPULATION to WEIGHTED EVALUATION, we use exactly the same voters, and p remains the same as well. If a voter was not a colluder in the CONSTRUCTIVE CW-MANIPULATION instance and we were thus given its vote, in the WEIGHTED EVALUATION instance its distribution places all of the probability mass on that vote. If the voter was in the collusion, its distribution is now uniform. We set $r = 0$. Now, clearly, in the WEIGHTED EVALUATION instance there is a chance of p winning if and only if there exists some way for the latter votes to be cast so as to make p win - that is, if and only if there is an effective collusion in the CONSTRUCTIVE CW-MANIPULATION problem. ■

Next we show that if evaluating the winning probability is hard, individual manipulation is also hard.

Definition 45 (CONSTRUCTIVE INDIVIDUAL WEIGHTED (IW)-MANIPULATION UNDER UNCERTAINTY) *We are given a single manipulative voter with a weight, weights for all the other voters, a distribution over all the others' votes, a candidate p, and a number r, where $0 \leq r \leq 1$. We are asked whether the manipulator can cast its vote so that p wins with probability greater than r.*

Theorem 83 *If* WEIGHTED EVALUATION *is NP-hard for a rule (even with k candidates and restrictions on the distribution), then* CONSTRUCTIVE IW-MANIPULATION UNDER UNCERTAINTY *is also NP-hard for it (even with k candidates and the same restrictions).*

Proof: For the reduction from WEIGHTED EVALUATION to CONSTRUCTIVE IW-MANIPULATION UNDER UNCERTAINTY, simply add a manipulator with weight 0. ■

Combining Theorems 82 and 83, we find that with weighted voters, if in some rule coalitional manipulation is hard in the complete-information setting, then even individual manipulation is hard if others' votes are uncertain. Applying this to the hardness results from Subsection 8.3.3, this means that all of the rules of this section other than plurality and cup are hard to manipulate by individuals in the weighted case when the manipulator is uncertain about the others' votes.

Finally, we show that WEIGHTED EVALUATION can be hard even if CONSTRUCTIVE CW-MANIPULATION is not. If we relax the requirement that a vote is represented by a total order over the candidates, we can also allow for the following common voting rule:

- *approval.* Each voter labels each candidate as either approved or disapproved. The candidate that is approved by the largest number of voters wins.

For the approval rule, CONSTRUCTIVE CW-MANIPULATION is trivial: the universally most potent manipulation is for all of the manipulators to approve the preferred candidate p, and to disapprove all other candidates. However, WEIGHTED EVALUATION is hard:

Theorem 84 *In the approval rule,* WEIGHTED EVALUATION *is NP-hard, even if $r = 0$, the votes are drawn independently, and the distribution over each vote has positive probability for at most 2 of the votes.*

Proof: We reduce an arbitrary PARTITION instance to the following WEIGHTED EVALUATION instance. There are 3 candidates, p, a, and b. There are $2K + 1$ votes approving $\{p\}$. Additionally, for each k_i in the PARTITION instance, there is a vote of weight $2k_i$ that approves $\{a\}$ with probability $\frac{1}{2}$, and $\{b\}$ with probability $\frac{1}{2}$. We set $r = 0$. Clearly, p wins if and only if a and b are each approved by precisely $2K$ of the vote weight. But this is possible (and happens with positive probability) if and only if there is a partition. ■

Unweighted voters

Finally, we study what implications can be derived for the hardness of manipulation in settings with unweighted voters.

Definition 46 UNWEIGHTED EVALUATION *is the special case of* WEIGHTED EVALUATION *where all the weights are* 1. CONSTRUCTIVE INDIVIDUAL UNWEIGHTED (IU)-MANIPULATION UNDER UNCERTAINTY *is the special case of* CONSTRUCTIVE INDIVIDUAL WEIGHTED (IW)-MANIPULATION UNDER UNCERTAINTY *where all the weights are* 1.

First, we show that for rules for which WEIGHTED EVALUATION is hard, UNWEIGHTED EVALUATION is also hard. This assumes that the language for specifying the probability distribution is rich enough to allow for perfect correlations between votes (that is, some votes are identical with probability one[12]).

[12]Representation of such distributions can still be concise.

Theorem 85 *If* WEIGHTED EVALUATION *is NP-hard for a rule (even with k candidates and restrictions on the distribution), then* UNWEIGHTED EVALUATION *is also NP-hard for it if we allow for perfect correlations (even with k candidates and the same restrictions—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: For the reduction from WEIGHTED EVALUATION to its unweighted version, we replace each vote of weight κ with κ unweighted votes; we then make these κ votes perfectly correlated. Subsequently we pick a representative vote from each perfectly correlated group, and we impose a joint distribution on this vote identical to the one on the corresponding vote in the WEIGHTED EVALUATION problem. This determines a joint distribution over all votes. It is easy to see that the distribution over outcomes is the same as in the instance from which we reduced; hence, the decision questions are equivalent. ■

We would like to have an analog of Theorem 83 here, to show that UNWEIGHTED EVALUATION being hard also implies that CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY is hard. Unfortunately, the strategy used in the proof of Theorem 83—setting the manipulator's weight to 0—does not work, because the weight of the manipulator must now be 1. Instead, we rely on the following two theorems, which each require an additional precondition. The first one shows that if the WEIGHTED EVALUATION problem is hard even in settings where there is no possibility that the candidate p is tied for winning the election, then the CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY problem is also hard.

Theorem 86 *If* WEIGHTED EVALUATION *is NP-hard for a rule even in settings where ties will not occur (even with k candidates and restrictions on the distribution of the votes), then* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is also NP-hard (with the same r) if we allow for perfect correlations (even with k candidates and the same restrictions on the distribution of the nonmanipulators' votes—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: We reduce the EVALUATION instance to a MANIPULATION instance by first adding a single manipulator. Because ties will not occur, there must exist a (rational) weight $w > 0$ such that if the manipulator's vote has this weight, then the manipulator's vote will never affect the outcome. Without loss of generality, we can assume that this weight can be written as $w = \frac{1}{M}$ for some sufficiently large integer M. Now, multiply all the weights by M so that the manipulator's vote has weight 1 and all the weights are integers again. Then, replace each voter of weight κ by κ perfectly correlated, unweighted voters. Clearly, the manipulator will still not affect the outcome, and thus the distribution over outcomes is the same as in the instance we reduced from; hence, the decision questions are equivalent. ■

Theorem 86 applies to most of the rules under study:

Corollary 6 *For each one of the following rules,* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is NP-hard: Borda (even with 3 candidates), veto (even with 3 candidates), STV (even*

with 3 candidates), plurality with runoff (even with 3 candidates), and maximin (even with 4 candidates). This holds even if $r = 0$, the votes are either drawn independently or perfectly correlated, and only the following types of (marginal) distributions are allowed: 1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote. (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)

Proof: To show that we can apply Theorem 86 to each of these rules, we first make the following observation. For each of these rules, the reduction that we gave to show that CONSTRUCTIVE CW-MANIPULATION is hard has the property that if there exists no successful manipulation, candidate p cannot even be tied for winning the election (and, of course, if there is a successful manipulation, no other candidate will tie with p for winning the election). Because of this, when we apply the reduction from Theorem 82 to these instances, there is no chance that a tie for winning the election between p and another candidate will occur, and we can apply Theorem 86. ■

Unfortunately, for the Copeland rule, in the reduction given in Theorem 74, an unsuccessful manipulation may still leave p tied for winning the election. To show that CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY is hard for this rule as well, we need the following theorem:

Theorem 87 *If* WEIGHTED EVALUATION *is NP-hard for a rule even in settings where $r = 0$ and one of the voters with a uniform distribution over votes has weight 1 (even with k candidates and restrictions on the distribution of the votes), then* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is also NP-hard even in settings where $r = 0$ if we allow for perfect correlations (even with k candidates and the same restrictions on the distribution of the nonmanipulators' votes—except those conflicting with perfect correlations). (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)*

Proof: We reduce the EVALUATION instance to a MANIPULATION instance by replacing the voter with a uniform distribution over votes and weight 1 by the manipulator, and using the same distribution over the other voters' votes as before. If there is nonzero probability of p winning in the EVALUATION instance, then there must exists some vector of votes with nonzero probability for which p wins with nonzero probability. Then, in the MANIPULATION instance, consider the vote in this vote vector cast by the voter that was replaced by the manipulator. If the manipulator places this vote, then with nonzero probability, the same vector will occur and p will win with nonzero probability. Conversely, suppose that in the MANIPULATION instance there exists a vote for the manipulator such that p wins with nonzero probability. Then, in the EVALUATION instance, there is some nonzero probability that the voter replaced by the manipulator casts this vote (because that voter's distribution over votes is uniform). It follows that there is nonzero probability that p will win in the EVALUATION instance. Hence, the decision questions are equivalent. ■

Corollary 7 *For the Copeland rule (even with 4 candidates),* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is NP-hard . This holds even if $r = 0$, the votes are either drawn independently or perfectly correlated, and only the following types of (marginal) distributions are allowed:*

1) the vote's distribution is uniform over all possible votes, or 2) the vote's distribution puts all of the probability mass on a single vote. (This is assuming that a group of κ perfectly correlated votes can be represented using only $O(\log(\kappa))$ space.)

Proof: Because PARTITION is hard even when one of the integers to be partitioned is 1, we can assume that one of the manipulators in the proof of Theorem 74 has weight 1, which allows us to apply Theorem 87. ■

As a final remark, we observe that the manipulation questions discussed in this subsection are not necessarily even in NP. However, when $r = 0$, the manipulation question can also be phrased as saying "does there exist a manipulation that has *some* chance of succeeding?" We note that this question is in fact in NP.

The following figure summarizes the flow of the theorems presented in this subsection.

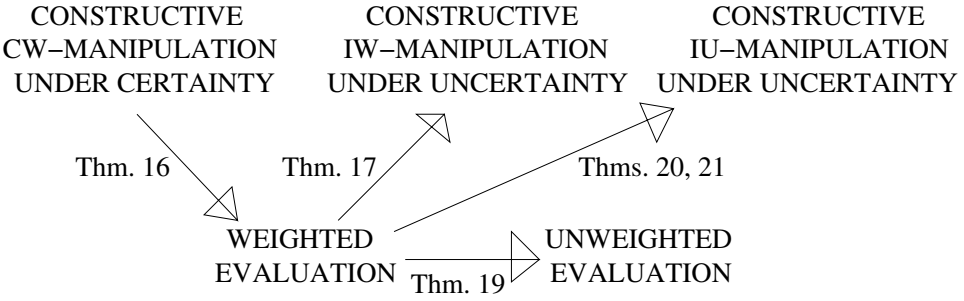

Figure 8.2: *The flow of the theorems in this subsection.*

This concludes the part of the dissertation studying worst-case hardness of manipulation. In the next section, we move on to a more ambitious goal: voting rules that are *usually* hard to manipulate.

8.4 Nonexistence of usually-hard-to-manipulate voting rules

One weakness that all of the above results have in common is that they only show *worst-case* hardness. That is, the results show that it is unlikely that an efficient algorithm can be designed that finds a beneficial manipulation in *all* instances for which a beneficial manipulation exists. However, this does not mean that there do not exist efficient manipulation algorithms that find a beneficial manipulation in *many* instances. If such algorithms do in fact exist, then computational hardness constitutes a leaky barrier to manipulation at best (though it is presumably still better than nothing).

A truly satisfactory solution to the problem would be to have a rule that is hard to manipulate in *all* instances. However, this is too much to ask for: for example, a manipulation algorithm could have a small database of instances with precomputed solutions, and it would merely need to check against this database to successfully manipulate some instances. Still, we may ask whether it is possible to make (say) 99% of instances hard to manipulate. It is generally agreed that this would have a much greater impact on the design of voting rules in practice than merely worst-case hardness [Conitzer *et al.*, 2003; Elkind and Lipmaa, 2005b], but none of the multiple efforts to achieve this objective have succeeded.

In this section, we present an impossibility result that makes it seem unlikely that such an objective can be achieved by any reasonable voting rule. This is not the first such impossibility result: a previous result [Procaccia and Rosenschein, 2006] shows that a specific subclass of voting rules is usually easy to manipulate when the number of candidates is constant and a specific distribution over instances is used (where the distribution is chosen to have certain properties that would appear to make manipulation more difficult). By contrast, our result does not require any restriction on the voting rule, number of candidates, or distribution over instances. Our result states that a voting rule/instance distribution pair cannot simultaneously be usually hard to manipulate, and have certain natural properties (which depend on the rule and distribution).

8.4.1 Definitions

Manipulation

As we saw in the previous section, the computational problem of manipulation has been defined in various ways, but typical definitions are special cases of the following general problem: given the nonmanipulators' votes, can the manipulator(s) cast their votes in such a way that one candidate from a given set of preferred candidates wins? In this section, we study a more difficult manipulation problem: we require that the manipulator(s) find the set of *all* the candidates that they can make win (as well as votes that will bring this about). This stronger requirement makes our impossibility result stronger: we will show that even this more powerful type of manipulation cannot be prevented.

Definition 47 *A* manipulation instance *is given by a voting rule R, a vector of* nonmanipulator votes $v = (r_1^{NM}, \ldots, r_n^{NM})$*, a vector of* weights $v^s = (d_1^{NM}, \ldots, d_n^{NM})$ *for the nonmanipulators, and a vector of weights $w^s = (d_1^M, \ldots, d_k^M)$ for the manipulators. A manipulation algorithm* succeeds *on this instance if it produces a set of pairs $\{(w_{i_1}, c_{i_1}), \ldots, (w_{i_q}, c_{i_q})\}$ such that 1) if the manipulators cast the vector of votes w_{i_j}, then c_{i_j} wins, and 2) if a candidate c does not occur in this set as one of the c_{i_j}, then there is no vector of manipulator votes w that makes c win.*

An instance is *manipulable* if the manipulators can make more than one candidate win. Nonmanipulable instances are easy to solve: any algorithm that is sound (in that it does not produce incorrect (w_{i_j}, c_{i_j}) pairs) and that returns at least one (w_{i_j}, c_{i_j}) pair (which is easy to do, by simply checking what the rule will produce for a given vector of votes) will succeed. Hence, we focus on manipulable instances only.

To investigate whether voting rules are *usually* easy to manipulate, we also need a probability distribution over (manipulable) instances. Our impossibility result does not require a specific distribution, but in the experimental subsection of the section, we study a specific family of distributions.

Weak monotonicity

Informally, a rule is *monotone* if ranking a candidate higher never hurts that candidate. All the rules mentioned before, with the exceptions of STV and plurality with runoff, are monotone. In this subsubsection, we formally define a weak notion of monotonicity that is implied by (but does not imply) standard notions of monotonicity. We note that we want our definition of monotonicity to be as weak as possible so that our impossibility result will be as strong as possible. We define when an

instance is weakly monotone, so that even rules that are not (everywhere) weakly monotone can be (and typically are) weakly monotone on most instances.

We will first define a stronger, more standard notion of monotonicity. For rules that produce a score for every candidate, a natural definition of monotonicity is the following: if a manipulator changes his vote so that a given candidate is ranked ahead of a larger set of candidates, then that candidate's score should not decrease. However, not every rule produces a score. Thus, we use the following definition of monotonicity, which does not rely on scores. (Monotonicity as defined above for rules that produce a score implies monotonicity in the sense of the following definition.)

Definition 48 *We say that a voting rule R is* monotone *for manipulators with weights w^s and nonmanipulator votes v with weights v^s if for every pair of candidates c_1, c_2 and every pair of manipulator vote vectors $w_1 = (r_1^1, \ldots, r_k^1), w_2 = (r_1^2, \ldots, r_k^2)$, the following condition holds: if*

- *c_2 wins when the manipulators vote w_1, and*
- *for any manipulator i, for any candidate c such that $c_2 \succ_{r_i^1} c$, we have $c_2 \succ_{r_i^2} c$, and*
- *for any manipulator i, for any candidate c such that $c \succ_{r_i^1} c_1$, we have $c \succ_{r_i^2} c_1$;*

then c_1 does not win when the manipulators vote w_2.

Thus, given a monotone instance, if each manipulator decreases the set of candidates that he prefers to the current winner, and increases the set of candidates that he prefers to a given other candidate, then the latter candidate cannot become the winner due to this. (It is, however, possible that a third candidate will win after the change, since we did not restrict how that candidate's position in the ranking changed.)

Now we can define our weaker notion of monotonicity:

Definition 49 *We say that a voting rule R is* weakly monotone *for manipulators with weights w^s and nonmanipulator votes v with weights v^s if for every pair of candidates c_1, c_2, one of the following conditions holds: 1) c_2 does not win for any manipulator votes; or 2) if all the manipulators rank c_2 first and c_1 last, then c_1 does not win.*

We now show that our notion is indeed weaker:

Theorem 88 *Monotonicity implies weak monotonicity.*

Proof: Given a monotone rule R, consider any pair of candidates c_1, c_2, and votes $w_2 = (r_1^2, \ldots, r_k^2)$ for the manipulators in which c_2 is always ranked first and c_1 is always ranked last. Suppose that there are votes $w_1 = (r_1^1, \ldots, r_k^1)$ that make c_2 win. Then if the manipulators changed from w_1 to w_2, every manipulator would decrease the set of candidates that he prefers to c_2 and increase the set of candidates that he prefers to c_1. Hence, by monotonicity, c_1 cannot win. ■

On the other hand, weak monotonicity does not imply monotonicity. For instance, consider the scoring rule defined (for four candidates) by $\langle 3, 1, 2, 0 \rangle$. Ranking a candidate second instead of third can end up hurting that candidate, so this rule is clearly not monotone. However, under this rule, if all the manipulators rank candidate c_2 first and c_1 last, and c_1 still wins, then c_1 must be at least $3k$ points ahead of c_2 (not counting the manipulators' votes), so c_2 does not win for any manipulator votes. Hence, the rule does satisfy weak monotonicity.

8.4.2 Impossibility result

We now present an algorithm that seeks to identify two candidates that can be made to win. The algorithm is universal in that it does not depend on the voting rule used, except for the places in which it calls the rule as a (black-box) subroutine.

Find-Two-Winners(R, C, v, v^s, w^s)
choose an arbitrary manipulator vote vector w_1
$c_1 \leftarrow R(C, v, v^s, w_1, w^s)$
for every $c_2 \in C, c_2 \neq c_1$ {
  choose w_2 in which every vote ranks c_2 first and c_1 last
  $c \leftarrow R(C, v, v^s, w_2, w^s)$
  if $c_1 \neq c$ **return** $\{(w_1, c_1), (w_2, c)\}$ }
return $\{(w_1, c_1)\}$

If voting rule R can be executed in polynomial time, then so can Find-Two-Winners.

Theorem 89 Find-Two-Winners *will succeed on every instance that both a) is weakly monotone and b) allows the manipulators to make either of exactly two candidates win.*

Proof: Find-Two-Winners is sound in the sense that it will never output a manipulation that is incorrect. It will certainly find one candidate that the manipulators can make win (c_1). Thus, we merely need to show that it will find the second candidate that can be made to win; let us refer to this candidate as c'. If the algorithm reaches the iteration of the **for** loop in which $c_2 = c'$, then in this round, either $c \neq c_1$, in which case we must have $c = c'$ because there are no other candidates that can be made to win, or $c = c_1$. But in the latter case, we must conclude that $c_2 = c'$ cannot be made to win, due to weak monotonicity—which is contrary to assumption. Hence it must be that $c = c'$. If the algorithm does not reach the iteration of the **for** loop in which $c_2 = c'$, it must have found a manipulation that produced a winner other than c_1 in an earlier iteration, and (by assumption) this other winner can only be c'. ■

It is not possible to extend the algorithm so that it also succeeds on all weakly monotone instances in which *three* candidates can be made to win. When three candidates can be made to win, even under monotone rules, it is possible that one of these candidates can only win if some manipulators vote differently from the other manipulators. In fact, as we saw in the previous section, the problem of deciding whether multiple weighted manipulators can make a given candidate win is NP-complete, even when there are only three candidates and the Borda or veto rule is used (both of which are monotone rules). Any algorithm that does succeed on all weakly monotone instances in which at most three candidates can be made to win would be able to solve this NP-complete problem, and thus cannot run in polynomial time (or it would show that P = NP).

The impossibility result now follows as a corollary.

Corollary 8 *For any $p \in [0, 1]$, there does not exist any combination of an efficiently executable voting rule R and a distribution d over instances such that*

1. *the probability of drawing an instance that is both a) weakly monotone, and b) such that either of exactly two candidates can be made to win, is at least p; and*

2. *for any computationally efficient manipulation algorithm, the probability that an instance is drawn on which the algorithm succeeds is smaller than p.*

This impossibility result is relevant only insofar as one expects a voting rule to satisfy Property 1 in the corollary (high probability of drawing a weakly monotone instance in which either of exactly two candidates can be made to win). Before we argue why one should in fact expect this, it is helpful to consider how a skeptic might argue that an impossibility result such as this one is irrelevant. At a minimum, the skeptic should argue that one of the properties required by the result is not sufficiently desirable or necessary to insist on it. The skeptic could make her case much stronger by actually exhibiting a voting rule that satisfies all properties except for the disputed one, and that still seems intuitively desirable. (For example, Arrow's impossibility result [Arrow, 1963] is often criticized on the basis that the *independence of irrelevant alternatives* property is unnecessarily strong, and this is the only property that common voting rules fail to satisfy.)

Conversely, we will first argue directly for the desirability of Property 1 in Corollary 8. We will then provide indirect evidence that it will be difficult to construct a sensible rule that does not satisfy this property, by showing experimentally that all common rules satisfy it very strongly (that is, a large fraction of manipulable instances are weakly monotone and such that only two candidates can be made to win).

8.4.3 Arguing directly for Property 1

In this subsection, we argue why one should expect many manipulable instances to be both a) weakly monotone and b) such that the manipulator(s) can make either of exactly two candidates win. We first make a simple observation: if manipulable instances are usually weakly monotone, and they usually allow the manipulator(s) to make either of exactly two candidates win, then a significant fraction of manipulable instances have both of properties a) and b). More precisely:

Proposition 10 *If the probability of drawing a weakly monotone instance is p, and the probability of drawing an instance in which either of exactly two candidates can be made to win is q, then the probability of drawing an instance with both properties is at least $p + q - 1$.*

Proof: The probability of drawing an instance that is *not* weakly monotone is $1 - p$, and the probability of drawing an instance in which more than two candidates can be made to win is $1 - q$. From this, it follows that the probability of drawing an instance with both properties is at least $1 - (1 - p) - (1 - q) = p + q - 1$. ■

With this in mind, we will now argue separately for each of the two properties a) and b).

The argument for Property a)—most manipulable instances should be weakly monotone—is easy to make. The reason is that if the manipulators rank certain candidates higher, this should, in general, benefit those candidates. If this were not the case, then the manipulators' votes would lose their natural interpretation that they support certain candidates over others, and we are effectively

asking the manipulators to submit a string of bits without any inherent meaning.[13] It should also be noted that most common voting rules are in fact monotone (on all instances), and the few rules for which nonmonotone instances can be constructed are often severely criticized because of this (even if the rule is in fact monotone on most instances).

Arguing for Property b)—most manipulable instances should be such that the manipulators can make either of exactly two candidates win—is somewhat more difficult. For simplicity, consider rules that produce a score for every candidate. As the number of voters grows, typically, candidates' scores tend to separate. This is especially the case if some candidates systematically tend to be ranked higher than others by voters, *e.g.* because these candidates are intrinsically better. (One interpretation of voting that dates back at least to Condorcet is the following: everyone has a noisy signal about the relative intrinsic quality of the candidates, and the purpose of an election is to maximize the probability of choosing the intrinsically best candidate [de Caritat (Marquis de Condorcet), 1785].) Thus, given a large number of nonmanipulators, it is unlikely that the scores of the two leading candidates will be close enough to each other that a few manipulators can make either one of them win; but it is significantly more unlikely that the scores of the *three* leading candidates will be close enough that the manipulators can make any one of them win. So, even given that some manipulation is possible, it is unlikely that more than two candidates can be made to win. This argument suggests that it is likely that most common voting rules in fact satisfy Property b). But it is also an argument for why we should *require* a voting rule to have this property, because, especially when we think of voting as being a process for filtering out the noise in voters' individual preferences to find the intrinsically best candidate, we *want* the candidates' scores to separate.

In the next subsection, we show experimentally that common voting rules in fact strongly satisfy properties a) and b).

8.4.4 Arguing experimentally for Property 1

In this subsection, we show experimentally that for all the common voting rules, most manipulable instances are in fact weakly monotone and such that either of exactly two candidates can be made to win. Because most of the rules that we study are in fact monotone on all instances, this mostly comes down to showing that at most two candidates can be made to win in most manipulable instances. Unfortunately, for quite a few of these rules, it is NP-hard to determine whether more than two candidates can be made to win (this follows from results in the previous section). Rather than trying to solve these NP-hard problems, we will be content to provide a *lower bound* on the fraction of manipulable instances in which either of exactly two candidates can be made to win. We obtain these lower bounds by characterizing, for each rule that we study, an easily computable sufficient (but not necessary) condition for an instance to be such that either of exactly two candidates can be made to win. For at least some rules, the lower bound is probably significantly below the actual fraction—which only strengthens the relevance of the impossibility result.

One useful property of these lower bounds is that they are independent of how the manipulators' total weight is distributed. Because of this, only the manipulators' total weight matters for the

[13] Incidentally, if this does not bother us, it is easy to design rules that are always hard to manipulate: for example, we can count an agent's vote only if part of its vote (represented as a string of bits) encodes the solution to (say) a hard factoring problem. Of course, this is not a very satisfactory voting rule.

purpose of our experiments, and we can assume, without loss of generality, that each manipulator has weight 1.

It should be noted that we can only show results for specific distributions of instances, because we need a specific distribution to conduct an experiment. Therefore, it cannot be said with certainty that other distributions would lead to similar results, although for reasonable distributions it appears likely that they would. One should keep in mind that the vote aggregator typically has no control over the distribution over voters' preferences, so that constructing an artificial distribution for which these results do not hold is unlikely to be helpful. We now present the specific distributions that we study.

For a given number of candidates, number of nonmanipulators, and number of manipulators, we generate instances as follows. (This is Condorcet's distribution, which we discussed in Chapter 3 because the maximum likelihood estimator of the correct ranking under this distribution is the Kemeny rule [Kemeny, 1959; Young, 1995].) We assume that there is a "correct" ranking t of the candidates (reflecting the candidates' unknown intrinsic quality), and the probability of drawing a given vote r is proportional to $p^{a(r,t)}(1-p)^{m(m-1)/2-a(r,t)}$, where $a(r,t)$ is the number of pairs of candidates on whose relative ranking r and t agree (they agree if either $c_1 \succ_r c_2$ and $c_1 \succ_t c_2$, or $c_2 \succ_r c_1$ and $c_2 \succ_t c_1$). p is a given noise parameter; if $p = 1$ then all voters will produce the correct ranking, and if $p = 0.5$ then we are drawing every vote (independently and uniformly) completely at random. This distribution is due to Condorcet [de Caritat (Marquis de Condorcet), 1785], and one way to interpret it is as follows. To draw a vote, for each pair of candidates c_1, c_2, randomly decide whether the vote is going to agree with the correct ranking on the relative ranking of c_1 and c_2 (with probability p), or disagee (with probability $1-p$). This may lead to cycles (such as $c_1 \succ c_2 \succ c_3 \succ c_1$); if so, restart.

These distributions often produce nonmanipulable instances. Ideally, we would discard all nonmanipulable instances, but this requires us to have an algorithm for detecting whether an instance is manipulable. If we know that the instance is weakly monotone, we can simply use algorithm Find-Two-Winners for this purpose. However, a few of the rules that we study (STV and plurality with runoff) are not monotone on all instances. In fact, for these rules, it is NP-hard to tell whether the instance is manipulable (this follows from results in the previous section). For these rules, we use simple sufficient (but not necessary) conditions to classify an instance as nonmanipulable. We will classify each nonmanipulable instance that does not satisfy this condition as having more than two potential winners, so that our results are still lower bounds on the actual ratio.

In the experiments below, we draw 1000 manipulable instances at random (by drawing and discarding instances as described above), and for each voting rule, we show our lower bound on the number of instances in which the manipulators can make either of exactly two candidates win. For rules that are not monotone everywhere, we also show a lower bound on the number of such instances that are *also* weakly monotone (indicated by "<rule> - monotone"). We also consider the Condorcet criterion—recall that a rule satisfies the Condorcet criterion if any candidate that wins all of its pairwise elections must win the overall election—and show a lower bound on the number of instances for which these properties are satisfied for *any* rule satisfying the Condorcet criterion.

In our first experiment (Figure 8.3), we have three candidates, one manipulator, and significant noise in the votes ($p = 0.6$). For all the rules under study, the fraction of instances satisfying the property approaches 1 as the number of nonmanipulator votes grows.

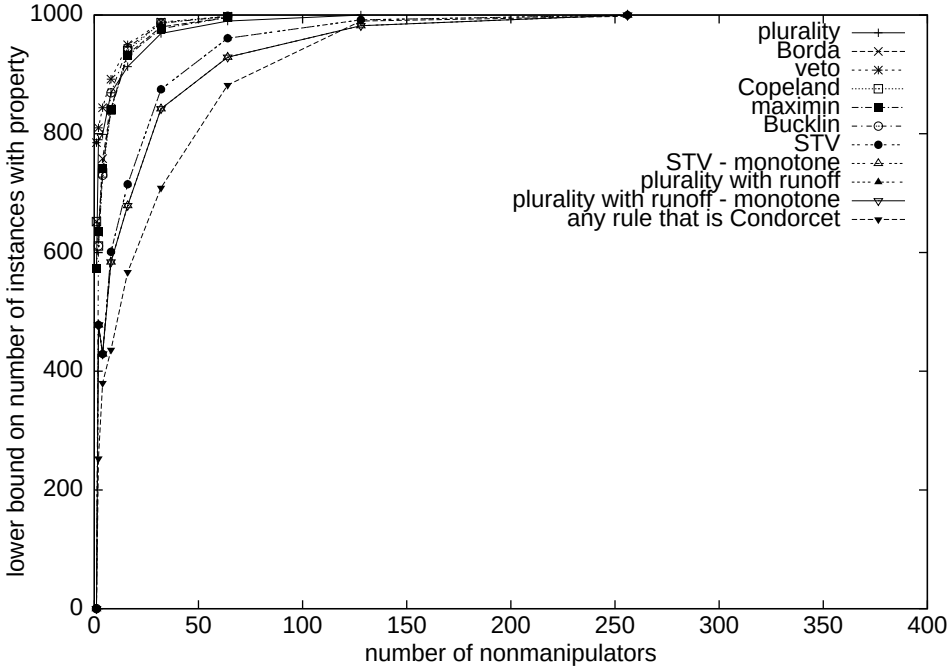

Figure 8.3: p = 0.6, one manipulator, three candidates.

Next, we show what happens when we maximize noise ($p = 0.5$), so that votes are drawn completely at random (Figure 8.4). Even under such extreme noise, the fraction of instances satisfying the property approaches 1 or at least becomes very large (> 0.7) for every one of the rules. However, it is no longer possible to say this for *any* rule satisfying the Condorcet criterion (although the specific common rules that do satisfy this criterion satisfy the property for a very large fraction of instances).

Next, we show results when there are multiple (specifically, 5) manipulators (Figure 8.5). The results are qualitatively similar to those in Figure 8.3, although for smaller numbers of nonmanipulators the fractions are lower. This makes sense: when the number of nonmanipulators is relatively small, a large coalition is likely to be able to make any candidate win.

Finally, we experiment with an increased number of candidates (Figure 8.6).

Now, the lower bound on the fraction of instances satisfying the property approaches 1 for all rules but STV. The lower fraction for STV is probably at least in part due to the fact that the lower bound that we use for STV is relatively weak. For example, any instance in which the manipulators can change the eliminated candidate in at least two rounds is counted as having more than two candidates that the manipulators can make win. This is extremely conservative because changes in which candidate is eliminated in a given round often do not change the winner.

8.4.5 Can the impossibility be circumvented?

One may wonder whether there are ways to circumvent the impossibility result presented in this section. Specifically, one may still be able to construct voting rules that are usually hard to ma-

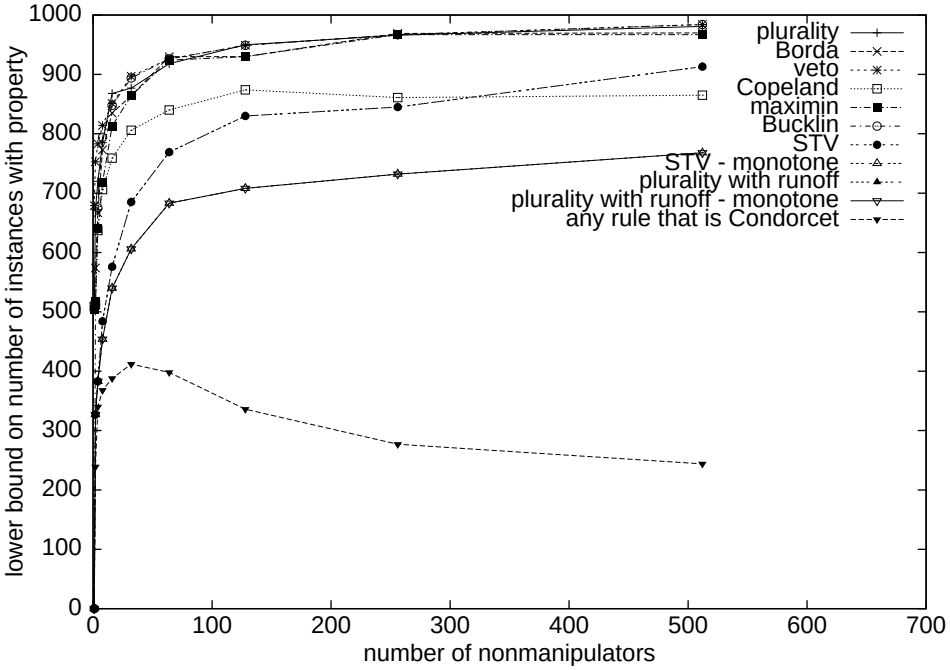

Figure 8.4: p = 0.5, one manipulator, three candidates.

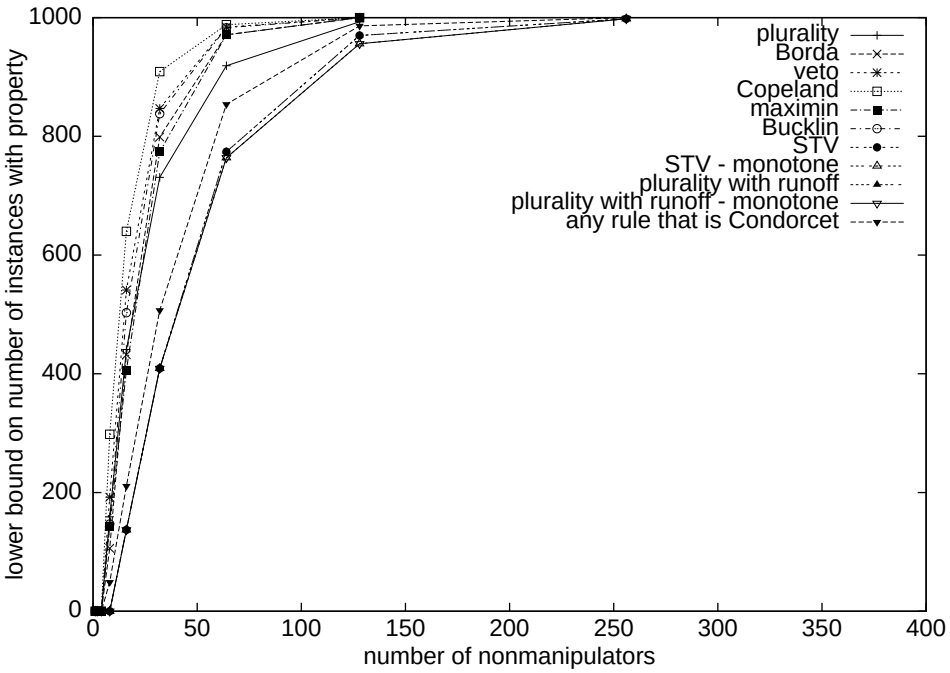

Figure 8.5: p = 0.6, five manipulators, three candidates.

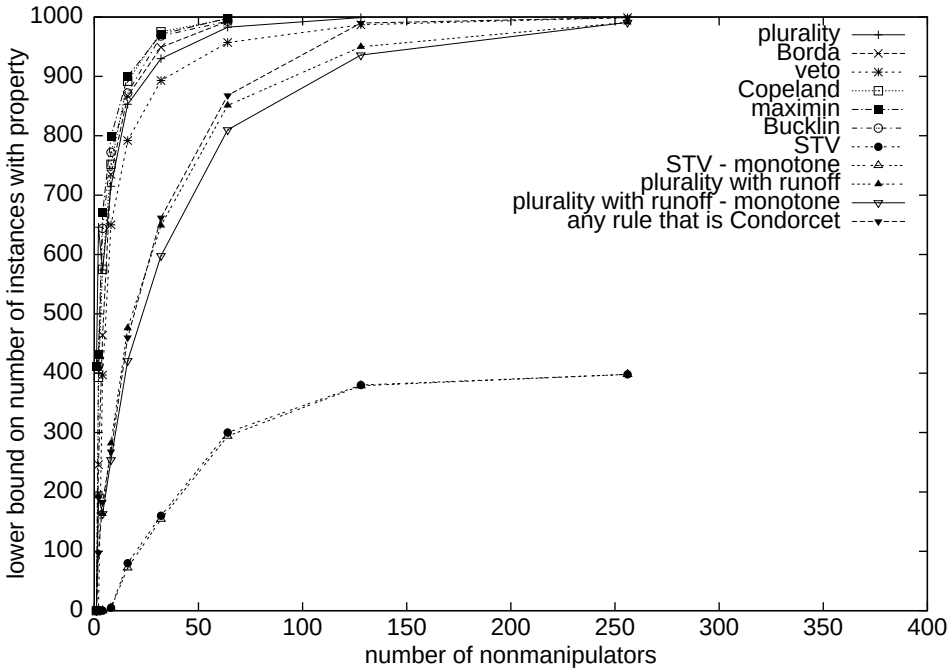

Figure 8.6: p = 0.6, one manipulator, five candidates.

nipulate by considering a larger class of voting rules, a class that contains rules that do not satisfy the preconditions of the impossibility result. In this subsection, we discuss various approaches for circumventing the impossibility result, and their prospects. (One approach that we will not discuss is that of constructing distributions over voters' preferences for which the impossibility result fails to hold, because, as we mentioned earlier, the distribution over voters' preferences is typically not something that the vote aggregator has any control over.)

Allowing low-ranked candidates to sometimes win

The impossibility result is only significant if in a sizable fraction of manipulable instances, only two candidates can be made to win. One may try to prevent this by using a voting rule that sometimes chooses as the winner a candidate that in fact did not do well in the votes (according to whatever criterion), thereby increasing the number of candidates that can be made to win in manipulable instances. Of course, having such a candidate win is inherently undesirable, but if it occurs rarely, it may be a price worth paying in order to achieve hardness of manipulation.

If we take this approach, and in addition allow for the rule to be *randomized*, then we can construct reasonable voting rules that are in fact strategy-proof (that is, no beneficial manipulation is ever possible). Consider, for example, the following voting rule:

Definition 50 *The* Copeland-proportional *rule chooses candidate c as the winner with probability proportional to c's Copeland score.*

An alternative interpretation of this rule is the following: choose a pair of candidates at random; the winner of their pairwise election wins the entire election. (If the pairwise election is tied, choose one of the two candidates at random.)

Theorem 90 *Copeland-proportional is strategy-proof.*

Proof: Suppose the manipulator knows which pair of candidates is chosen. Then, any vote in which he ranks his preferred candidate higher than the other candidate is strategically optimal. But the manipulator can guarantee that this is the case, even without knowing the pair of candidates, simply by voting truthfully. ■

Recall Gibbard [1977]'s result that a randomized voting rule is strategy-proof only if it is a probability mixture of unilateral and duple rules, where a rule is unilateral if only one voter affects the outcome, and duple if only two candidates can win. The Copeland-proportional rule only randomizes over duple rules (namely, pairwise elections).

Of course, the Copeland-proportional rule is still not ideal. For instance, even a Condorcet winner has a probability of only $(m-1)/(m(m-1)/2) = 2/m$ of winning under this rule. (However, this rule will at least never choose a candidate that loses every pairwise election.) Thus, it may be worthwhile to try to construct voting rules that are usually hard to manipulate, and that are more likely to choose a "good" winner than Copeland-proportional.

Expanding the definition of a voting rule

The impossibility result may cease to hold when the rule can choose from a richer outcome space. As in the previous subsubsection, this may prevent problems of manipulability completely, by allowing the construction of strategy-proof rules. For example, *if* payments are possible and the agents have quasilinear utility functions, then a payment scheme such as the VCG mechanism can induce strategy-proofness. As another example that does not require these assumptions, suppose that it is possible to exclude certain voters from the effects of the election—as an illustrative example, in an election for a country's president, suppose that it is possible to *banish* certain voters to another country, in which it will no longer matter to those voters who won the election. (It does not matter whether living in the other country is otherwise more or less desirable than in the original country.) Then, we can augment any voting rule as follows:

Definition 51 *For any voting rule (in the standard sense) R, the* banishing *rule $B(R)$ always chooses the same winner as R, and banishes every pivotal voter. (A voter is* pivotal *if, given the other votes, he can make multiple candidates win.)*

Theorem 91 *For any rule R, the banishing rule $B(R)$ is strategy-proof.*

Proof: A voter who is not pivotal has no incentive to misreport, because by definition, his vote does not affect which candidate wins, and he cannot affect whether he is pivotal. A voter who is pivotal also has no incentive to misreport, because he cannot affect whether he is pivotal, and the winner of the election will not matter to him because he will be banished. ■

However, this scheme also has a few drawbacks. For one, it may not always be possible to completely exclude a voter from the effects of the election. Another strange property of this scheme is that no voter is ever capable of affecting his own utility, so that *any* vote is strategically optimal. Finally, it may be necessary to banish large numbers of voters. In fact, the following lemma shows that for any rule, the votes may turn out to be such that more than half of the voters must be banished.

Theorem 92 *For any responsive voting rule R, it is possible that more than half the voters are simultaneously pivotal. (We say that a voting rule is* responsive *if there are two votes r_1, r_2 such that everyone voting r_1 will produce a different winner than everyone voting r_2.)*

Proof: Let there be n voters total. Denote by v^i the vote vector where i voters vote r_1, and the remaining $n-i$ vote r_2. Because v^0 and v^n produce different winners under R, there must be some i such that v^i and v^{i+1} produce different winners. Thus, in v^i, all $n-i$ voters voting r_2 are pivotal, and in v^{i+1}, all $i+1$ voters voting r_1 are pivotal. Since $(n-i)+(i+1) > n$, at least one of $n-i$ and $i+1$ must be greater than $n/2$. ■

Rules that are hard to execute

The impossibility result only applies when an efficient algorithm is available for executing the rule, because algorithm Find-Two-Winners makes calls to such an algorithm as a subroutine. Thus, one possible way around the impossibility is to use a rule that is hard to execute. Indeed, as we pointed out before, a number of voting rules have been shown to be NP-hard to execute [Bartholdi *et al.*, 1989b; Hemaspaandra *et al.*, 1997; Cohen *et al.*, 1999; Dwork *et al.*, 2001; Ailon *et al.*, 2005]. Of course, we do actually need an algorithm for executing the rule to determine the winner of the election; and, although we cannot expect this to be a worst-case polynomial-time algorithm, it should at least run reasonably fast in practice for the rule to be practical. But if the algorithm does run fast in practice, then it can also be used by the manipulators as the subroutine in Find-Two-Winners. Therefore, this approach does not look very promising.

8.5 Summary

In this chapter, we studied mechanism design for bounded agents. Specifically, we looked at how hard it is computationally for agents to find a best response to given opponent strategies in various expressive preference aggregation settings.

In Section 8.1, we showed that there are settings where using the optimal (social-welfare maximizing) truthful mechanism requires the center to solve an NP-hard computational problem; but there is another, non-truthful mechanism that can be executed in polynomial time, and under which the problem of finding a beneficial manipulation is hard for one of the agents. Moreover, if the agent manages to find the manipulation, the produced outcome is the same as that of the best truthful mechanism; and if the agent does not manage to find it, the produced outcome is strictly *better*.

In Section 8.2, we showed how to *tweak* existing voting rules to make manipulation hard, while leaving much of the original nature of the rule intact. The tweak studied in this section consists of

adding one preround to the election, where candidates face each other one against one. The surviving candidates continue to the original protocol. Surprisingly, this simple and universal tweak makes typical rules hard to manipulate! The resulting protocols are NP-hard, #P-hard, or PSPACE-hard to manipulate, depending on whether the schedule of the preround is determined before the votes are collected, after the votes are collected, or the scheduling and the vote collecting are interleaved, respectively. We proved general sufficient conditions on the rules for this tweak to introduce the hardness, and showed that the most common voting rules satisfy those conditions. These are the first results in voting settings where manipulation is in a higher complexity class than NP (presuming PSPACE $\neq$ NP).

In Section 8.3, we noted that all of the previous results on hardness of manipulation in elections required the number of candidates to be unbounded. Such hardness results lose relevance when the number of candidates is small, because manipulation algorithms that are exponential only in the number of candidates (and only slightly so) might be available. We gave such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) rule, which had been shown hard to manipulate in the above sense. To obtain hardness-of-manipulation results in settings where the number of candidates is a small constant, we studied *coalitional* manipulation by *weighted* voters. (We show that for simpler manipulation problems, manipulation cannot be hard with few candidates.) We studied both *constructive* manipulation (making a given candidate win) and *destructive* manipulation (making a given candidate not win). The following tables summarize our results.

Number of candidates	2	3	4,5,6	≥ 7
Borda	P	NP-complete	NP-complete	NP-complete
veto	P	NP-complete	NP-complete	NP-complete
STV	P	NP-complete	NP-complete	NP-complete
plurality with runoff	P	NP-complete	NP-complete	NP-complete
Copeland	P	P	NP-complete	NP-complete
maximin	P	P	NP-complete	NP-complete
randomized cup	P	P	P	NP-complete
regular cup	P	P	P	P
plurality	P	P	P	P

Complexity of CONSTRUCTIVE CW-MANIPULATION

Number of candidates	2	≥ 3
STV	P	NP-complete
plurality with runoff	P	NP-complete
Borda	P	P
veto	P	P
Copeland	P	P
maximin	P	P
regular cup	P	P
plurality	P	P

Complexity of DESTRUCTIVE CW-MANIPULATION

We also showed that hardness of manipulation in this setting implies hardness of manipulation

by an individual in unweighted settings when there is uncertainty about the others' votes.

All of the hardness results mentioned above only show hardness in the worst case; they do not preclude the existence of an efficient algorithm that *often* finds a successful manipulation (when it exists). There have been attempts to design a rule under which finding a beneficial manipulation is *usually* hard, but they have failed. To explain this failure, in Section 8.4, we showed that it is in fact impossible to design such a rule, if the rule is also required to satisfy another property: a large fraction of the manipulable instances are both weakly monotone, and allow the manipulators to make either of exactly two candidates win. We argued why one should expect voting rules to have this property, and showed experimentally that common voting rules satisfy it. We also discussed approaches for potentially circumventing this impossibility result, some of which appear worthwhile to investigate in future research.

The manipulation problems defined in this chapter did not involve sophisticated strategic reasoning: we simply assumed that the manipulator(s) knew the others' votes (or at least a distribution over them). Acting in a strategically optimal way becomes more difficult when this information is not available, and the manipulator(s) must reason over how the others are likely to act. This is the topic of the next chapter.