When Are Elections with Few Candidates Hard to Manipulate?

VINCENT CONITZER

Duke University

TUOMAS SANDHOLM

Carnegie Mellon University

AND

JÉRÔME LANG

IRIT, Université Paul Sabatier

Abstract. In multiagent settings where the agents have different preferences, preference aggregation is a central issue. Voting is a general method for preference aggregation, but seminal results have shown that all general voting protocols are manipulable. One could try to avoid manipulation by using protocols where determining a beneficial manipulation is hard. Especially among computational agents, it is reasonable to measure this hardness by computational complexity. Some earlier work has been done in this area, but it was assumed that the number of voters and 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. We give such an algorithm for an individual agent to manipulate the Single Transferable

V. Conitzer and T. Sandholm were funded by the National Science Foundation under CAREER Award IRI-97031222, Grant IIS-9800994, ITR IIS-0081246, ITR IIS-0121678, and IIS-0427858, a Sloan Fellowship and an IBM Ph.D Fellowship.

This article combines and expands on two early, shorter conference papers: V. Conitzer and T. Sandholm, "Complexity of manipulating elections with few candidates", In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI-02)*, pp. 314–319, and V. Conitzer, J. Lang, and T. Sandholm, "How many candidates are needed to make elections hard to manipulate?" In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-03)*, pp. 201–214.

Authors' addresses: V. Conitzer, Duke University, Computer Science Department, Levine Science Research Center, Box 90129, Durham, NC 27708, e-mail: conitzer@cs.duke.edu; T. Sandholm, Carnegie Mellon University, Computer Science Department, 5000 Forbes Avenue, Pittsburgh, PA 15213, e-mail: sandholm@cs.cmu.edu; J. Lang, IRIT, Université Paul Sabatier, 31062 Toulouse Cedex, France, e-mail: lang@irit.fr.

Vote (STV) protocol, which has been shown hard to manipulate in the above sense. This motivates the core of this article, which derives hardness results for realistic elections where the number of candidates is a small constant (but the number of voters can be large).

The main manipulation question we study is that of *coalitional* manipulation by *weighted* voters. (We show that for simpler manipulation problems, manipulation cannot be hard with few candidates.) 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 the *plurality*, *Borda*, *STV*, *Copeland*, *maximin*, *veto*, *plurality with runoff*, *regular cup*, and *randomized cup* protocols. We also show 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 (but not vice-versa). To our knowledge, these are the first results on the hardness of manipulation when there is uncertainty about the others' votes.

Categories and Subject Descriptors: F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms: Algorithms, Economics, Theory

Additional Key Words and Phrases: Computational social choice, hardness of manipulation, voting

ACM Reference Format:

Conitzer, V., Sandholm, T., and Lang, J. 2007. When are elections with few candidates hard to manipulate? J. ACM 54, 3, Article 14 (June 2007), 33 pages. DOI = 10.1145/1236457.1236461 http://doi.acm.org/ 10.1145/1236457.1236461

1. *Introduction*

In multiagent systems, a group of agents often has to make a common decision, yet they have different preferences about which decision is made. In such settings, it is of central importance to be able to aggregate the preferences, that is, to make a socially desirable decision as to which *candidate* is chosen from a set of candidates. Candidates could be potential presidents, joint plans, allocations of goods or resources, etc. Voting is the most general preference aggregation scheme, and has been used in several multiagent decision making problems in computer science, such as collaborative filtering (e.g., Pennock et al. [2000]) and planning among multiple automated agents (e.g., Ephrati and Rosenschein [1991, 1993]).

One key problem voting mechanisms are confronted with is that of *manipulation* by the voters. An agent is said to manipulate or *vote strategically* when it does not rank the alternatives according to its true preferences, but rather so as to make the eventual outcome most favorable to itself. For example, if an agent prefers Nader to Kerry to Bush, but knows that Nader has too few other supporters to win, while Kerry and Bush are close to each other, the agent would be better off by declaring Kerry as its top candidate. Manipulation is an undesirable phenomenon because social choice schemes are tailored to aggregate preferences in a socially desirable way, and if the agents reveal their preferences insincerely, a socially undesirable candidate may be chosen.

The issue of strategic voting has been studied extensively. A seminal negative result, the *Gibbard–Satterthwaite theorem*, shows that if there are three or more candidates, then in any nondictatorial voting scheme, there are preferences under which an agent is better off voting strategically [Gibbard 1973; Satterthwaite 1975]. (A voting scheme is called dictatorial if one of the voters dictates the social choice no matter how the others vote.)

This is a significant problem in elections where voters are human. In automated group decision making where the voters are *software agents*, the manipulability of protocols is even more problematic, for at least three reasons. First, the algorithms they use to decide how to vote must be coded explicitly. Given that the voting algorithm needs to be designed only once (by an expert), and can be copied to large numbers of agents (even ones representing unsophisticated human voters), it is likely that rational strategic voting will increasingly become an issue, unmuddied by irrationality, emotions, etc. Second, software agents have more computational power and are more likely to find effective manipulations. Third, the settings where software agents are used for voting are likely to be more anonymous, and therefore the voters are likely to be less scrupulous. For example, they might not worry about the norm that manipulation, in itself, is considered inapproriate (if the voter's neighbor knows his preferences on an issue and sees him vote differently, then the neighbor will consider him a liar). As another example, anonymous voters are freed from the social pressures of voting in a particular way (e.g., voting for a school rather than a cigar factory), so the space of viable votes—and thus viable manipulations—is larger.

Nevertheless, there are several avenues for trying to avoid the possibility of manipulation. First, social choice theorists and economists have studied settings where the voters' preferences are restricted. Under certain restrictions, such as *single-peaked preferences* or *quasilinear preferences*, nonmanipulable protocols exist (e.g., [Mas-Colell et al. 1995; Ephrati and Rosenschein 1991, 1993]). The weakness of this approach is that in practice the protocol designer cannot be sure that the agents' preferences fall within the restriction. If they do not, it is *impossible* to vote truthfully, so the protocol *forces* the agents to manipulate. A second approach to avoiding manipulation is randomization. For example, a dictator could be chosen at random among the voters, in which case there would be no incentive for manipulation. The randomization approach is undesirable if it introduces too much noise into the election process, and it turns out that almost complete randomization (as in the example above) is required in order to obtain a nonmanipulable protocol [Gibbard 1977, 1978]. Furthermore, randomization can introduce manipulation possibilities even when none would have existed (for the preferences that the agents happen to have) under a deterministic protocol [Zeckhauser 1969].

We take a third tack toward avoiding manipulation: *ensuring that finding a beneficial manipulation is so hard computationally that it is unlikely that voters will be able to manipulate*. So, unlike in most of computer science, here high computational complexity is a desirable property. The harder it is to manipulate, the better. Especially in the context of software agents, this computational complexity is best measured with the usual tools from theoretical computer science. The approach of using computational complexity to avoid manipulation has not received much attention and many problems are still open. Some hardness results have been proven under the assumption that not only the number of voters but also *the number of candidates* is unbounded [Bartholdi et al. 1989a; Bartholdi and Orlin 1991; Conitzer and Sandholm 2003].[1] Such hardness results lose relevance when the number of

[1]For an unbounded number of candidates, voting rules can be made hard to manipulate by only slightly tweaking them and thus preserving many of their desirable properties. Conitzer and Sandholm [2003]

candidates is small, because manipulation algorithms that are exponential only in the number of candidates (and only slightly so) might be available. In the appendix, we give such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) protocol, 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 article, 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 1. *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 non-colluders are known, evaluating the result of a (coalitional) vote is as easy as determining the winner of an election, which must be in P for practical voting mechanisms.[2] 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

showed that by adding a preround of pairwise elections (where the winner of each pair proceeds to the original voting rule), a broad class of voting rules (including the most common voting rules) become 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. Elkind and Lipmaa [2005] showed that many hybrid voting rules (where one rule is used to curtail the set of candidates and another rule is used to select from among them) are NP-hard to manipulate, even if each of the component rules is easy to manipulate.

[2] However, there exist voting mechanisms 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; Conitzer et al. 2006; Conitzer 2006]—but only for large numbers of candidates.

there are reasonable settings where *evaluating* a manipulation is NP-hard. For instance, if 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 article 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* protocols. It turns out that the voting protocols under study become hard to manipulate at 3 candidates, 4 candidates, 7 candidates, or never. The remainder of this article 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*.

2. *Voting Protocols*

We now define the voting setting. Let $\mathcal{V} = \{v_1, \ldots, v_n\}$ be the finite set of *voters*. Let $\mathcal{X} = \{1, \ldots, m\}$ be the finite set of *candidates*. The *preferences* of voter v_i are given by a linear order O_i on $\mathcal{X}$. Such a linear order is represented as $(c_1, c_2, \ldots, c_m)$, meaning that c_1 is preferred to c_2, c_2 is preferred to c_3, etc. A *preference profile* is a vector $P = \langle O_1, \ldots, O_n \rangle$ of individual preferences.

A *voting protocol* is a function from the set of all preference profiles to the set of candidates $\mathcal{X}$.[3] The following list reviews the most common voting protocols. In the protocols that are based on scores, the candidate with the highest score wins. In each of the listed protocols (even the ones that have multiple rounds), the voters submit their preferences up front. That is, the voters are not allowed to change their preference revelations during the execution of the protocol.

—*Positional Scoring Protocols*. Let $\vec{\alpha} = \langle \alpha_1, \ldots, \alpha_m \rangle$ be a vector of integers such that $\alpha_1 \geq \alpha_2 \cdots \geq \alpha_m$. For each voter, a candidate receives α_1 points if it is ranked first by the voter, α_2 if it is ranked second, etc. The *score* $s_{\vec{\alpha}}$ of a candidate is the total number of points the candidate receives. The *Borda* protocol is the scoring protocol with $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$. The *plurality* protocol (aka majority rule) is the scoring protocol with $\vec{\alpha} = \langle 1, 0, \ldots, 0 \rangle$. The *veto* protocol is the scoring protocol with $\vec{\alpha} = \langle 1, 1, \ldots, 1, 0 \rangle$.

—*Maximin* (aka *Simpson*). For any two distinct candidates i and j, let $N(i, j)$ be the number of voters who prefer i to j. The *maximin score* of i is $s(i) = \min_{j \neq i} N(i, j)$.

—*Copeland*. For any two distinct candidates i and j, let $C(i, j) = +1$ if $N(i, j) > N(j, i)$ (in this case we say that i beats j in their pairwise election), $C(i, j) = 0$ if $N(i, j) = N(j, i)$ and $C(i, j) = -1$ if $N(i, j) < N(j, i)$. The *Copeland score* of candidate i is $s(i) = \sum_{j \neq i} C(i, j)$.

[3]Some of the voting protocols require tie-breaking rules (at different stages of the execution of the protocol). These rules are usually left undefined. The results in this article do not depend on tie-breaking rules.

—*Single Transferable Vote* (STV). The protocol proceeds through a series of $m-1$ rounds. In each round, the candidate with the lowest plurality score (that is, the least number of voters ranking it first among the remaining candidates) is eliminated (and each of the votes for that candidate "transfer" to the next remaining candidate in the order given in that vote). The winner is the last remaining candidate.

—*Plurality with Run-off*. In this protocol, a first round eliminates all candidates except the two with the highest plurality scores. Then votes are transferred to these (as in the *STV* protocol), and a second round determines the winner from these two.

—*Cup* (sequential binary comparisons). The cup is defined by a balanced[4] binary tree T with one leaf per candidate, and a *schedule*, that is, an assignment of candidates to leaves (each leaf gets one candidate). Each non-leaf node is assigned the winner of the pairwise election of the node's children; the candidate assigned to the root wins. The *regular cup* protocol assumes that the assignment of candidates to leaves is known by the voters before they vote. In the *randomized cup* protocol, the assignment of candidates to leaves is chosen uniformly at random after the voters have voted. Note that the randomized cup protocol differs from all the other protocols under discussion in the sense that all the others are deterministic.

In some settings, the voters are *weighted*. A *weight function* is a mapping $w : \mathcal{V} \rightarrow N$. When voters are weighted, the above protocols are applied by simply considering a voter of weight k to be k different voters.[5] Different possible interpretations can be given to weights. They may represent the decision power of a given agent in a voting setting where not all agents are considered equal. The weight may correspond to the size of the community that the voter represents (such as the size of the state). Or, when agents vote in partisan groups (e.g., in parliament), the weights may correspond to the size of the group (each group acts as one voter).

3. *Manipulating an Election*

In this section, we define our computational problem precisely. We lead into the definition by discussing the different dimensions of the election manipulation problem:

(I) *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 article, hardness results for manipulation by

[4] "Balanced" means that the difference in depth between two leaves can be at most one.

[5] This is why we assume that weights are integers. The results also apply to all rational weights because they can be converted to integers by multiplying all the weights by all the weights' denominators.

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.

(II) *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 very 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.[6] (Manipulation by individuals in the complete information setting is hard for some voting protocols if one allows the number of voters *and the number of candidates* to be unbounded [Bartholdi et al. 1989a; Bartholdi and Orlin 1991; Conitzer and Sandholm 2003]. However, in the appendix of this article, we give a manipulation algorithm for such a protocol that scales to fairly large numbers of candidates.) (3) Again, as we will demonstrate later in this article, hardness results for manipulation *by coalitions* in the complete information setting also imply hardness of manipulation *by individuals* in the incomplete information setting.

(III) *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 in the Introduction. (We recall that the number of effectively different vote vectors is polynomial due to the interchangeability of the different equiweighted voters, see Proposition 1.) (2) As we will demonstrate later in this article, 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.

(IV) *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 protocols.) 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

[6]This assumes that the voting protocol is easy to execute—as most protocols are (including all the ones under study in this article).

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 protocols 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 (which, in our setting, is a constant; however, the same argument applies even when the number of candidates is not constant).

Third, the complexity of destructive manipulation is directly related to the complexity of determining 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 2002].

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

Definition 1 (*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 that are still open (the manipulators' votes), and a preferred candidate p. For deterministic protocols, we are asked whether there is a way to cast the votes in T so that p wins the election. For randomized protocols, we are additionally given a distribution over instantiations of the voting protocol, 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 2 (*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 that are still open (the manipulators' votes), and a disliked candidate h. For deterministic protocols, we are asked whether there is a way to cast the votes in T so that h does *not* win the election. For randomized protocols, we are additionally given a distribution over instantiations of the voting protocol, 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 protocols, we do not consider a manipulation successful if it leaves candidate p or h tied for the win.[7] One way of viewing this is as follows. If ties are broken randomly, then technically, we are dealing with a randomized protocol. Then, we can set r so that a tie for the win does not give p enough probability of winning (e.g., $r = 2/3$), or so that a tie gives h too much probability of winning (e.g., $r = 1/(m + 1)$).

As additional justification for requiring hardness even with few candidates, in the appendix, we present an algorithm for an individual voter to manipulate the STV protocol that is exponential only in the number of candidates, and scales to reasonably large numbers of candidates. (Reading this appendix is not necessary to comprehend the rest of the article.)

4. *Complexity of Manipulation*

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 protocol 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 protocols (the lower this number, the less manipulable the protocol). For each protocol 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.)

4.1. CONSTRUCTIVE MANIPULATION.

We first present our results for constructive manipulation.

4.1.1. *Easiness Results.*

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

THEOREM 1. *For the plurality protocol,* 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 protocol, the proof is a little more involved:

THEOREM 2. *For the cup protocol* (*given the assignment of candidates to leaves*), CONSTRUCTIVE CW-MANIPULATION *can be solved in polynomial time* (*for any number of candidates*).

[7] Our proofs do not depend on this specification, with the exception of Theorem 7.

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 $O(m^3n)$ in accordance with the Master Theorem [Cormen et al. 1990]. □

The remaining easiness results that we show in this section 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 protocols considered in this article (and indeed all "reasonable" voting protocols) are equivalent to the plurality protocol, and hence both types of manipulation (constructive and destructive) are in P for all of the protocols. However, some protocols are still easy to manipulate constructively with more than two 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 protocols.

THEOREM 3. *If the Copeland protocol with three 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 three 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 that 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, that is, 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 4. *If the maximin protocol with three 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 three candidates be p, a, and b. We are given the nonmanipulators' votes S, and the weights for the manipulators' votes T. Again, it is easy to show that all the manipulators can rank p first without harm. Now, let us suppose that there exists a way for the manipulators to make p win (where each manipulator votes (p, a, b) or (p, b, a)). We will show that if this is so, then also, either it is the case that all manipulators voting (p, a, b) will make p win, or it is the case that all manipulators voting (p, b, a) will make p win. Let us consider two cases for the outcome of the whole election when the manipulators cast their successful votes (potentially including both (p, a, b) and (p, b, a) votes):

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, WLOG, 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 (contrary to assumption) p does not win. So this case cannot occur.

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 5. *If the randomized cup protocol with six 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 whose order might differ across manipulators' votes. As in the case of two 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 three candidates in G.

If there are four 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 three 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 four candidates in G.

Finally, if there are five 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. □

4.1.2. *Hardness Results.* We are now ready to prove hardness results that match the bounds given by the easiness results above. (That is, for every protocol that 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 3 (*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 6. *For any positional scoring protocol other than the plurality protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for three candidates.*

PROOF. First, note that when there are only three candidates, a positional scoring protocol 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 protocol is either meaningless (if $\alpha_1 = 0$) or equivalent to the plurality protocol, so we can assume $\alpha_2 > 0$. Then, we can scale the α_i appropriately, which will not affect the protocol.) 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 three 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 positional scoring procedures satisfy monotonicity, we can assume without loss of generality that the voters in the coalition T rank p first. Let x (respectively, y) be the total weight of voters in T of the voters who vote (p, a, b) (respectively, (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 1. *For the* veto *and* Borda *protocols,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for three candidates.*

(On a historical note, we actually proved Corollary 1 before we discovered its generalization to Theorem 6; the generalization was independently discovered by us, Hemaspaandra and Hemaspaandra [2005], and Procaccia and Rosenschein [2006].)

THEOREM 7. *For the Copeland protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for four 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 four 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 8. *For the* maximin *protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for four 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 four 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 protocol, 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 9. *For the* STV *protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for three 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 three 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 p 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 protocol is hard:

THEOREM 10. *For the* plurality with *runoff protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for three candidates.*

PROOF. Showing the problem is in NP is easy. To show it is NP-hard, we observe that with three candidates, the plurality with runoff protocol coincides with the STV protocol, and CONSTRUCTIVE MANIPULATION for STV with three candidates is NP-hard. □

THEOREM 11. *For the randomized cup protocol,* CONSTRUCTIVE CW-MANIPULATION *is* NP*-complete for seven 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 seven 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

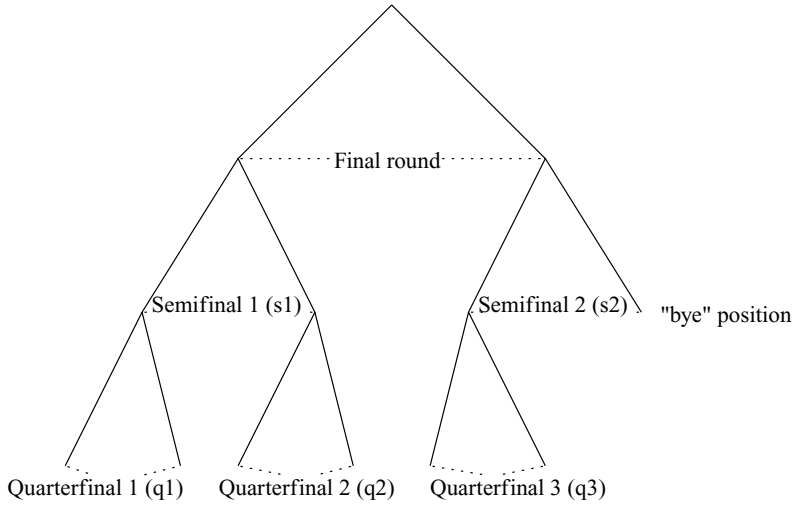

FIG. 1. The cup used in the proof.

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 (see Figure 1).

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

4.2. DESTRUCTIVE MANIPULATION. We now present our results for destructive manipulation.

4.2.1. *Easiness Results*. 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 protocols can be destructively manipulated in polynomial time, for any number of candidates. Interestingly, for most of the other protocols under study, destructive manipulation turns out to be drastically easier than constructive manipulation! The following theorem shows a sufficient condition for protocols to be easy to manipulate destructively.

THEOREM 12. *Consider any voting protocol 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 protocol, 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 2. *Destructive manipulation can be done in polynomial time for the veto, Borda, Copeland, and maximin protocols.*

4.2.2. *Hardness Results*. Theorem 12 does not apply to the STV and plurality with runoff protocols. We now show that destructive manipulation is in fact hard for these protocols, even with only three candidates.

THEOREM 13. *For the* STV *protocol with three 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 three 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 protocol:

THEOREM 14. *For the plurality with runoff protocol with three 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 three candidates, plurality with runoff coincides with STV, and DESTRUCTIVE MANIPULATION for STV with three candidates is NP-hard, as we proved in Theorem 13. □

5. *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.[8] Therefore, we should acknowledge that it is likely that the language used 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 section 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

[8]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.

manipulations, but all results apply to the destructive cases as well and the proofs are analogous. We restrict our attention to deterministic protocols.

5.1. WEIGHTED VOTERS. First, we show that with weighted voters, in protocols 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 4 (*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 15. *If* CONSTRUCTIVE CW-MANIPULATION *is* NP*-hard for a deterministic protocol* (*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 5 (*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.

While this definition appears natural and is easy to work with, one may wonder whether the following scenario is possible: it is always easy for the manipulator to determine its strategically optimal vote (or at least just a vote that is better than a truthful vote, if such a vote exists), but hard to determine whether this vote (or any other vote) makes p's probability of winning greater than r. If this were so, then the protocol would be hard to manipulate in the sense of the above definition, but would still always be manipulated. However, it seems that determining the strategically optimal vote (or even just a vote that is better than a truthful vote) would generally be hard if manipulation in the sense of the above definition is hard. For example, suppose that the manipulator has a utility of r for some candidate a, a utility of 1 for p, and a utility of 0 for any other candidate that has any possibility of winning. Additionally, suppose that the manipulator has the choice only between casting votes (including the truthful vote) that make a win with probability 1, or

casting votes that make a win with probability 0. (For example, under STV, there can be a situation where, if the manipulator (truthfully) votes for some third candidate b (that has no possibility of winning), this would rescue b from elimination in the first round, after which a is guaranteed to win due to how the votes transfer; but if the manipulator does not vote for b, then b is eliminated, after which a is immediately eliminated.) If this is the case, then the question of whether the manipulator can beneficially manipulate comes down to whether the manipulator can make p win with probability greater than r (the above definition). We note that this argument/proof sketch takes us beyond constructive and destructive manipulation (there are at least three different utilities for the manipulator); covering this in more detail would take us too far afield. Instead, we return to the above definition.

THEOREM 16. *If* WEIGHTED EVALUATION *is* NP*-hard for a protocol* (*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 15 and 16, we find that with weighted voters, if in some protocol 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 Section 4, this means that all of the protocols of this article 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 protocol:

—*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 protocol, 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 17. *In the approval protocol,* 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 two of the votes.*

PROOF. We reduce an arbitrary PARTITION instance to the following WEIGHTED EVALUATION instance. There are three 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. □

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

Definition 6. 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 protocols 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 (i.e., some votes are identical with probability one[9]).

In this section, we will assume that the manipulation instance is represented in a somewhat concise manner, namely, that if there are k perfectly correlated votes, then this is represented by a single description of the distribution of such a vote, plus the number k in *binary* representation. If k is given in unary representation, or a separate description is given for each vote, then the proofs do not work. Whether this assumption is reasonable presumably depends on the application setting.

THEOREM 18. *If* WEIGHTED EVALUATION *is* NP*-hard for a protocol* (*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*).

PROOF. For the reduction from WEIGHTED EVALUATION to its unweighted version, we replace each vote of weight k with k unweighted votes; we then make these k 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 16 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 16—setting the manipulator's weight to 0—does not work, because the weight of the manipulator must now be 1. (One may perhaps have felt uncomfortable with our use of a manipulator of weight 0 in the proof of Theorem 16; the below can also be used to avoid using such a manipulator in that setting.) 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 when we restrict our attention to instances where no ties can occur, then the CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY problem is also hard. (Formally, we say that a *tie occurs* if a voter of arbitrarily small weight (not necessarily an integer) can change the outcome of the election (i.e., whether p wins).)

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

THEOREM 19. *If* WEIGHTED EVALUATION *is* NP*-hard for a protocol even when restricting our attention to instances where ties cannot 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*).

PROOF. We reduce the EVALUATION instance to a MANIPULATION instance by first adding a single manipulator. Because ties cannot 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 k by k 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 19 applies to most of the protocols under study:

COROLLARY 3. *For each one of the following protocols,* CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY *is* NP*-hard: Borda (even with three candidates), veto (even with three candidates), STV (even with three candidates), plurality with runoff (even with three candidates), and maximin (even with four 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.*

PROOF. To show that we can apply Theorem 19 to each of these protocols, we first make the following observation. For each of these protocols, 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 15 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 19. □

Unfortunately, for the Copeland protocol, in the reduction given in Theorem 7, an unsuccessful manipulation may still leave p tied for winning the election. To show that CONSTRUCTIVE IU-MANIPULATION UNDER UNCERTAINTY is hard for this protocol as well, we need the following theorem:

THEOREM 20. *If* WEIGHTED EVALUATION *is* NP*-hard for a protocol 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*).

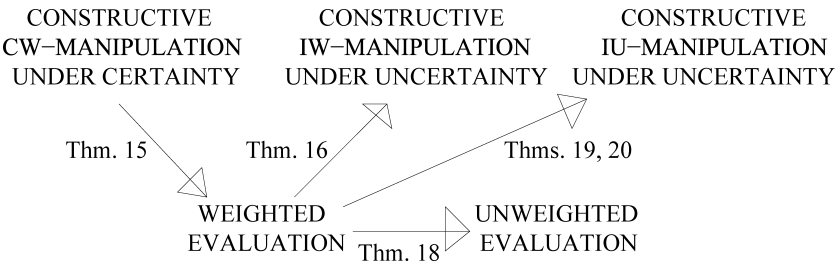

FIG. 2. The flow of the theorems in this section.

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 keeping the distributions over the other voters' votes the same. If there is nonzero probability of p winning in the EVALUATION instance, then there must exist 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 4. *For the Copeland protocol* (*even with four 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*.

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 7 has weight 1, which allows us to apply Theorem 20. □

As a final remark, we observe that the manipulation questions discussed in this section 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.

Figure 2 summarizes the flow of the theorems presented in this section.

6. *Conclusions and Future Research*

In multiagent settings where the agents have different preferences, preference aggregation is a central issue. Voting is a general method for preference aggregation, but seminal results have shown that all general voting protocols are manipulable.

One could try to avoid manipulation by using voting protocols where determining a beneficial manipulation is hard. Especially among computational agents, it is reasonable to measure this hardness by computational complexity. Some earlier work has been done in this area, but it was assumed that the number of voters and 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 the appendix, we give such an algorithm for an individual agent to manipulate the Single Transferable Vote (STV) protocol, which has been shown hard to manipulate in the above sense. The algorithm applies whether or not the voters are weighted.

This motivated the core of this article, which studied the complexity of manipulating realistic elections where the number of candidates is a small constant. We showed that with complete information about the others' votes, individual manipulation is easy, and coalitional manipulation is easy with unweighted voters. This left open two avenues for deriving high complexity results with few candidates. The first avenue is to 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. The second avenue is to ask whether there are reasonable settings where *evaluating* a manipulation is NP-hard. For instance, if 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 devoted most of this article to studying the first avenue. We studied both *constructive* manipulation (making a given candidate win) and *destructive* manipulation (making a given candidate not win). We studied not only *whether* any given protocol 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 protocols (the lower this number, the less manipulable the protocol). These results are summarized in the following tables:

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 devoted the remainder of this article to the second avenue. We showed that under weak assumptions, if weighted coalitional manipulation with complete information about the others' votes is hard in some voting protocol, then individual unweighted manipulation is hard when there is uncertainty (allowing for correlations) about the others' votes. We also showed that the converse is not true, that is, there exist protocols (such as approval) where weighted coalitional manipulation with complete information about the others' votes is easy, but individual unweighted manipulation is hard when there is uncertainty (allowing for correlations) about the others' votes. To our knowledge, these are the first results on the hardness of manipulation when there is uncertainty about the others' votes.

In summary, our results give a comparison of the relative hardness of manipulation in well-known voting protocols. Interestingly, the best-known protocol, plurality, is (together with the cup protocol) the easiest to manipulate (both constructive and destructive manipulation are in P for any number of candidates). In contrast, STV and plurality with runoff are the hardest to manipulate (both constructive and destructive manipulation are NP-complete, even with only three candidates). We also showed that randomizing over the instantiations of a protocol, after the votes are collected, can make manipulation harder (randomized cup is harder to manipulate than regular cup). Our results may also lead one to be generally less concerned about the possibility of manipulation as it is hard in the common case where not too much is known about how others will vote.

The results on complexity of manipulation to date (including ours) use NP-hardness as the complexity measure.[10,11] This is only a weak guarantee of hardness of manipulation. It means that there are infinitely many hard instances, but many instances may be easy to manipulate. Ideally, we would like to make *all*, or at least *most*, instances hard to manipulate. Future work includes studying other notions of hardness in the manipulation context, such as average-case completeness [Gurevich 1991] and instance complexity [Orponen et al. 1994]. Another interesting avenue is to try to embed a practically hard problem (e.g., factoring) in the manipulation

[10] One exception, which was published after the conference papers corresponding to this article, is the work by Conitzer and Sandholm on universal voting protocol tweaks [Conitzer and Sandholm 2003]. It uses not only NP-hardness, but also #P-hardness and PSPACE-hardness, as complexity measures. All of those measures are still worst-case complexity measures and the criticisms below apply.

[11] The same is true of studies of other complexity issues in voting settings, such as how complex it is for an election chairperson to influence the result by voter/candidate addition/suppression/partition [Bartholdi et al. 1992; Hemaspaandra et al. 2005], and how hard it is to bribe voters effectively and with minimal cost [Faliszewski et al. 2006].

problem. Future work also includes proving hardness of manipulation in more restricted protocols such as auctions, and with more restricted preferences.

6.1. SUBSEQUENT WORK: CAN MANIPULATION BE MADE HARD FOR *MOST* INSTANCES? Since the conference papers corresponding to this article were published, there has already been follow-on work on studying whether it is possible to make *most* instances hard to manipulate.

Ideally, one would like a voting rule where *every* instance is hard to manipulate. However, this is too much to ask for: 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.

Procaccia and Rosenschein [2006] show 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. (The distribution is chosen to have certain properties that would appear to make manipulation more difficult.)

Conitzer and Sandholm [2006] provide an impossibility result that does not require any restriction on the voting rule, number of candidates, or distribution over instances. It states that a voting rule/instance distribution pair cannot simultaneously be usually hard to manipulate, and have certain natural properties.

THEOREM 21 [CONITZER AND SANDHOLM 2006]. *For any $p \in [0, 1]$, there does not exist any combination of an efficiently executable voting rule and a distribution 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.*

Here, *weak monotonicity* is a very weak condition: A manipulation instance is weakly monotone if for every pair of candidates c_1, c_2, we have (i) c_2 does not win for any manipulator votes, or (ii) if all the manipulators rank c_2 first and c_1 last, then c_1 does not win. Weak monotonocity on most (if not all) instances is certainly a desirable property for a voting rule. A direct argument can also be made for why the remainder of Condition 1 (that is, part (b)) corresponds to a desirable property for a voting rule, but this argument is perhaps less convincing. Hence, this article proceeds to show that Condition 1 of the theorem is usually satisfied in practice.

The above impossibility result sheds negative light on the agenda of trying to construct voting rules that are usually hard to manipulate. Nevertheless, this article offers three possible directions for future work down this avenue:

—*Circumventing Property* (1*b*) *in the theorem by allowing low-ranked candidates to win with some probability*. Gibbard [1977] has shown that in order to make beneficial manipulation impossible by using randomization in a voting rule, quite extreme (undesirable) randomization is required. In contrast, here the idea is to combine a moderate amount of randomization with computational complexity to thwart manipulation.
—*Expanding the definition of a voting rule*. For example, if the pivotal voters are *banished* in such a way that they will not enjoy or suffer from the chosen

candidate, then truthful voting is a weakly dominant strategy. On the downside, for any responsive voting rule, more than half of the voters can be pivotal and hence may need to be banished. Future research could study expanding the definition of a voting rule less than what is needed to make truthful voting a dominant strategy, but enough to circumvent the theorem above. This way, computational complexity may yet have the potential to thwart manipulation.

—*Using voting rules that are hard to execute*. Manipulation algorithms usually use (and in particular, the one used to prove the theorem above uses) simulation of the voting rule to determine what outcome would come about under different insincere votes of the manipulators. If the voting rule itself is computationally hard to execute, these simulations may become prohibitively complex and thus complexity would thwart manipulation. On the downside, determining the election winner will become hard as a side effect. On the bright side, the winner needs to be determined only once, while the manipulation algorithm needs to simulate the voting rule multiple times.

Appendix: Algorithm for Individually Manipulating the Single Transferable Vote Protocol

When the number of candidates is unbounded, the STV protocol 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 appendix, 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 protocol uses some deterministic method for breaking ties (when choosing the loser to be eliminated at the end of a round), where the tie-breaking does not depend on aspects of the votes that the STV protocol 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 operator.

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:

$$T(k) \leq 1 + T(k-1) + T_0(k-1) \quad (1)$$

$$T_0(k) \leq 1 + T(k-1). \quad (2)$$

Combining the two, we get

$$T(k) \leq 2 + T(k-1) + T(k-2). \quad (3)$$

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 $\sum_{i=1}^{m} i = ((m+1)m)/2$ 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 $1 + \epsilon$.[12] 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 Eq. (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 Eq. (2) tight. Because the recurrences in Eqs. (1) and (2) are tight, the recurrence in Eq. (3) is tight.

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 $O(m+n)$ time, we have the following result.

THEOREM 22. *The algorithm MANIPULATE runs in time $O((m+n)(\frac{1+\sqrt{5}}{2})^m)$, where m is the number of candidates and n is the number of voters.*

So, MANIPULATE runs in $O((m+n) \cdot 1.62^m)$ time. While this function is exponential in m (which is to be expected given that the problem is NP-complete),

[12] 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.

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.

ACKNOWLEDGMENTS. We thank all of the reviewers of this article (including the earlier AAAI and TARK versions). One of the reviewers for this version gave especially detailed comments, for which we are especially grateful.

REFERENCES

BARTHOLDI, III, J., AND ORLIN, J. 1991. Single transferable vote resists strategic voting. *Social Choice Welf. 8*, 4, 341–354.

BARTHOLDI, III, J., TOVEY, C., AND TRICK, M. 1989a. The computational difficulty of manipulating an election. *Social Choice Welf. 6*, 3, 227–241.

BARTHOLDI, III, J., TOVEY, C., AND TRICK, M. 1989b. Voting schemes for which it can be difficult to tell who won the election. *Social Choice Welf. 6*, 157–165.

BARTHOLDI, III, J., TOVEY, C., AND TRICK, M. 1992. How hard is it to control an election? *Math. Comput. Modell. 16*, 8–9, 27–40. Formal theories of politics, II.

COHEN, W., SCHAPIRE, R., AND SINGER, Y. 1999. Learning to order things. *J. Artif. Intell. Res. 10*, 213–270.

CONITZER, V. 2006. Computing Slater rankings using similarities among candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Boston, MA).

CONITZER, V., DAVENPORT, A., AND KALAGNANAM, J. 2006. Improved bounds for computing Kemeny rankings. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Boston, MA).

CONITZER, V., AND SANDHOLM, T. 2002. Vote elicitation: Complexity and strategy-proofness. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Edmonton, Canada). 392–397.

CONITZER, V., AND SANDHOLM, T. 2003. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)* (Acapulco, Mexico). 781–788.

CONITZER, V., AND SANDHOLM, T. 2006. Nonexistence of voting rules that are usually hard to manipulate. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Boston, MA).

CORMEN, T., LEISERSON, C., AND RIVEST, R. 1990. *Introduction to Algorithms*. MIT Press, Cambridge, MA.

DAVENPORT, A., AND KALAGNANAM, J. 2004. A computational study of the Kemeny rule for preference aggregation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (San Jose, CA). 697–702.

DWORK, C., KUMAR, R., NAOR, M., AND SIVAKUMAR, D. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th World Wide Web Conference*. 613–622.

ELKIND, E., AND LIPMAA, H. 2005. Hybrid voting protocols and hardness of manipulation. In *Proceedings of the Annual International Symposium on Algorithms and Computation (ISAAC)*.

EPHRATI, E., AND ROSENSCHEIN, J. S. 1991. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Anaheim, CA). 173–178.

EPHRATI, E., AND ROSENSCHEIN, J. S. 1993. Multi-agent planning as a dynamic search for social consensus. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI)* (Chambery, France). 423–429.

FALISZEWSKI, P., HEMASPAANDRA, E., AND HEMASPAANDRA, L. 2006. The complexity of bribery in elections. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Boston, MA).

GIBBARD, A. 1973. Manipulation of voting schemes. *Econometrica 41*, 587–602.

GIBBARD, A. 1977. Manipulation of schemes that mix voting with chance. *Econometrica 45*, 665–681.

GIBBARD, A. 1978. Straightforwardness of game forms with lotteries as outcomes. *Econometrica 46*, 595–614.

GUREVICH, Y. 1991. Average case completeness. *J. Comput. Syst. Sci. 42*, 346–398.

HEMASPAANDRA, E., AND HEMASPAANDRA, L. 2005. Dichotomy for voting systems. Tech. Rep. 861, University of Rochester, Department of Computer Science.

HEMASPAANDRA, E., HEMASPAANDRA, L., AND ROTHE, J. 1997. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *J. ACM 44*, 6, 806–825.

HEMASPAANDRA, E., HEMASPAANDRA, L., AND ROTHE, J. 2005. Anyone but him: The complexity of precluding an alternative. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Pittsburgh, PA).

KARP, R. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, Eds. Plenum Press, New York, 85–103.

MAS-COLELL, A., WHINSTON, M., AND GREEN, J. R. 1995. *Microeconomic Theory*. Oxford University Press.

ORPONEN, P., KO, K.-I., SCHÖNING, U., AND WATANABE, O. 1994. Instance complexity. *J. ACM 41*, 1, 96–121.

PENNOCK, D. M., HORVITZ, E., AND GILES, C. L. 2000. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)* (Austin, TX). 729–734.

PROCACCIA, A. D., AND ROSENSCHEIN, J. S. 2006. Junta distributions and the average-case complexity of manipulating elections. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)* (Hakodate, Japan). 497–504.

ROTHE, J., SPAKOWSKI, H., AND VOGEL, J. 2003. Exact complexity of the winner problem for Young elections. In *Theory of Computing Systems*. Vol. 36(4). Springer-Verlag, New York, 375–386.

SATTERTHWAITE, M. 1975. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *J. Econ. Theory 10*, 187–217.

ZECKHAUSER, R. 1969. Majority rule with lotteries on alternatives. *Quart. J. Econ. 83*, 696–703.

RECEIVED MARCH 2007; ACCEPTED MARCH 2007