Ann Math Artif Intell (2016) 77:159–189
DOI 10.1007/s10472-015-9490-6

Manipulation complexity of same-system runoff elections

Zack Fitzsimmons[1] · **Edith Hemaspaandra**[2] ·
Lane A. Hemaspaandra[3]

Published online: 8 December 2015

Abstract Do runoff elections, using the same voting rule as the initial election but just on the winning candidates, increase or decrease the complexity of manipulation? Does allowing revoting in the runoff increase or decrease the complexity relative to just having a runoff without revoting? For both weighted and unweighted voting, we show that even for election systems with simple winner problems the complexity of manipulation, manipulation with runoffs, and manipulation with revoting runoffs are independent. On the other hand, for some important, well-known election systems we determine what holds for each of these cases. For no such systems do we find runoffs lowering complexity, and for some we find that runoffs raise complexity. Ours is the first paper to show that for natural, unweighted election systems, runoffs can increase the manipulation complexity.

Keywords Voting · Social choice · Runoff elections

Mathematics Subject Classification (2010) 91B12 · 91B14 · 91B10 · 68Q17

✉ Lane A. Hemaspaandra
lane@cs.rochester.edu
www.cs.rochester.edu/u/lane

Zack Fitzsimmons
zmf6921@rit.edu

Edith Hemaspaandra
eh@cs.rit.edu

[1] College of Computing and Information Sciences, Rochester Institute of Technology, Rochester, NY 14623, USA

[2] Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA

[3] Department of Computer Science, University of Rochester, Rochester, NY 14627, USA

1 Introduction

There is an extensive literature on two-stage and multistage voting. Although some of this study exists within economics, multistage elections and runoffs have been greatly influential in computational social choice during the past decade, due to such work as that of Elkind and Lipmaa [18] and Conitzer and Sandholm [12].

Particularly interesting recent work in this line has been done by Narodytska and Walsh [38]. They focus on manipulation of election systems of the form X THEN Y, i.e., an initial-round election under voting rule X, after which if there are multiple winners just those winners go on to a runoff election under voting rule Y, with the initial votes now restricted to the remaining candidates. The question at issue is whether a given manipulative coalition can vote in such a way as to make a distinguished candidate win (namely, win in the initial round if there is a unique winner in the initial round, or if not, then be a winner of the runoff).

Narodytska and Walsh [38] study the computational complexity of this question. They strongly address the issue of how the manipulation complexity of X and Y affect the manipulation complexity of X THEN Y. Viewing P as being easy and NP-hardness as being hard, they show that every possible combination of these manipulation complexities can be achieved for X, Y, and X THEN Y.

The present paper focuses on the complexity of X THEN X. That is, we are focused on the case where X is so valued as an election system that if X selects a unique winner, our election is over and we have our winner. However, if X in the initial round has tied winners, then we take just those winners and subject them to a runoff election, again using system X. (Votes in this second election will be over only the candidates who made it to the second round.) Like Narodytska and Walsh [38], we are interested in the case in which the second-round votes are simply the initial-round votes restricted to the remaining candidates, and the case, first raised by them, in which revoting is allowed in the second round.

Real-world examples exist of such same-system runoff elections. In general elections in North Carolina and many districts of California, election law specifies that if there are two or more candidates tied for being the winner in the initial plurality election, a plurality runoff election is held among just those candidates [9, 39]. (Curiously, in both states this approach is explicitly limited to general elections. For party-candidate-selection (so-called "primary") elections, perhaps to limit cost, both states break ties by lot, and in addition North Carolina breaks ties by lot if very few voters voted.) So (Plurality THEN Plurality)-with-revoting is being used.

Although Narodytska and Walsh [38] for X THEN Y elections showed that all combinations of P and NP-hardness for X, Y, and X THEN Y can be realized, their examples achieving that almost all have $X \neq Y$. Thus their broad results do not address the issue of whether all possibilities can be achieved if one seeks to use the same system for both the initial and the runoff election.

We show that every possibility can be achieved, even when the runoff is the same system as the initial election. Indeed, even in the three-way comparison of the complexity of X, the complexity of X with runoff (under X), and the complexity of X with a runoff (under X) with revoting, we show that every possibility of setting some or all of those to P or to NP-complete manipulation complexities can be realized. And we show that this can even be done while ensuring that the winner problem for X (i.e., determining whether a given candidate is a winner of a given election under X) remains in P, and can also be done both for the weighted and the unweighted cases.[1] For example, there are election systems X—having P winner problems—such that manipulation of X is NP-complete, manipulation of X THEN X is NP-complete, but manipulation of X THEN X with revoting is in P. And there are election systems X—having P winner problems—such that manipulation of X is in P, manipulation of X THEN X is NP-complete, but manipulation of X THEN X with revoting is in P. Briefly put, there is no inherent connection between these three complexities.

For the most important systems, however, it is very important to see what the effects of runoffs, and revoting runoffs are. For example, weighted plurality is easily seen to remain easy in all of our cases, e.g., manipulation of elections with runoffs, or with revoting runoffs, remains in P. However, that result itself is something of a fluke. We show that for every scoring protocol (a notion formally defined in Section 3, but we mention here that each scoring protocol regards a particular number of candidates) that is not Triviality, Plurality, or a disguised version of one of those, manipulation of elections with runoffs and manipulation of elections with revoting runoffs are NP-complete in the weighted case.

Although manipulation of unweighted veto is in P, we show that manipulation of unweighted veto elections with runoffs and manipulation of unweighted veto elections with revoting runoffs are NP-complete. For unweighted HalfApproval (the election system where each voter gives one point to his or her $\lceil \|C\|/2 \rceil$ top candidates and zero points to the rest), we prove that for both elections with runoffs and elections with revoting runoffs, the manipulation complexity, even when restricted to having at most one manipulator, is NP-complete. This contrasts with the nonrunoff manipulation complexity of unweighted HalfApproval, which is in P when there is one manipulator (and indeed for an unlimited number of manipulators, using the argument of [48, Corollary 4.2]).

Both Veto and HalfApproval are natural, unweighted cases where runoffs increase the manipulation complexity, and are the first such natural, unweighted cases in the literature. In fact, the previous literature did not find any such natural, unweighted example *even if one is allowed to use different election systems in the first and second rounds*. And for weighted manipulation, we provide the literature's first natural examples where a same-system runoff increases manipulation complexity.

For the case of one manipulator, a standard way of seeking to manipulate unweighted or weighted scoring protocols—pioneered for the unweighted case by Bartholdi, Tovey, and Trick [4], and extended in many papers since—is to use the natural greedy algorithm. However, we prove that for some scoring protocols X, the greedy approach fails on X THEN X.

---

[1] Why do we study weighted as well as unweighted cases? Weighted elections are quite important in human situations ranging from political settings such the US Electoral College to broad-participation settings such as stockholder elections, and are also useful in multiagent systems, such as combining the opinions of experts whose wisdom is not all considered equal. As the proofs of Section 4.1 of this paper show, weighted cases often require quite different proof approaches than their analogous unweighted cases, even when analogous results hold. In particular, regarding the "every possibility can be achieved" claim just made in the text, for weighted systems it is often particularly taxing to create settings that achieve the "P" parts of the claims. Also, in many settings, weighted and unweighted cases have different results altogether, e.g., for many election problems, for fixed numbers of candidates the problems are simple for the unweighted case but hard for the weighted case. This behavior is richly on display in Section 4.3, where the broad unweighted polynomial-time result of Theorem 22 (and the comment just before it) contrasts with our NP-completeness results in Theorems 23, 24, and 25.

Since we feel this tool will be useful elsewhere, we mention that Section 4.2 presents a tool, which we call the *AlwaysWinners* transformation, that is very helpful in bridging the gap caused by the fact that some people feel that the natural way to define the notion of an election rule is to allow any subset of the candidates to form the winner set, and some people feel that the natural way to define the notion of an election rule is to allow any *nonempty* subset of the candidates to form the winner set. Traditional social choice uses the latter definition, but many computationally oriented papers prefer the former for its symmetry. Section 4.2's *AlwaysWinners* tool is a construction that transforms an election system into a new one that will (except when the candidate set is empty) always have at least one winner. Crucially, the transformation is so tightly related to the original election system that it leaves unchanged the complexity of many election-attack problems.

2 Related work

There are quite a few papers whose focus is close to ours. Yet each differs in some important way. Centrally underpinning our study and framing is the work of Narodytska and Walsh [38] on manipulating X THEN Y elections. In a very real sense, our paper is merely about their diagonal—the case when one uses the same voting system in the original election and the runoff. However, since they were not specifically exploring the diagonals, their existence results in general don't address that case. (However, we must mention an important exception. They show that for STV′, a particular decisive form of STV, that STV′ and STV′ THEN STV′ are both NP-hard.[2]) Our constructions, which must work within a single system for both rounds, are quite different from theirs.

In contrast, the even earlier work of Elkind and Lipmaa [18] has a section on using the same system in each round, which is our focus also. However, their model (unlike Narodytska and Walsh and unlike our paper, which pass forward just the winners) is based on removing only the *least* successful candidate after a round. In particular, their model is of one or more initial pruning rounds, that in their examples use a "prune off the least successful candidate" (except in one case they prune off half the candidates) rule inspired by some election system X, after which there is a final round using some (potentially different) election system Y. So Elkind and Lipmaa's section on using the same system even in the final round (Section 5 of their paper) is about having one or more rounds using (a variant of) X to cut off the least popular candidate, and then a final round also using X. Other recent work on removing weakest candidates, usually sequentially, include that of Bag, Sabourian, and Winter [2] and Davies, Narodytska, and Walsh [16].

Related to the Elkind–Lipmaa work is the still earlier "universal tweaks" work of Conitzer and Sandholm [11], which shows that adding one pairwise (so-called) CUP-like

[2]To avoid confusing the literature's terminology, it is important for us to mention that there is a very slight, but arguably philosophically interesting, difference between the THEN we defined in the Introduction and the THEN operator as defined by Narodytska and Walsh [38]. Our and their definitions of X THEN Y can differ in outcome only on what happens if there is exactly one winner of the initial election. In our use of THEN (as given in this paper), in that decisive case the election is over. In their case, that one winner goes on to a one-person election under system Y. Their approach opens the door to having system Y in some cases kill off a single candidate who won the initial round. However, we stress that in their paper they absolutely never use that possibility, and so every result in their paper, including each one mentioned in this paper, holds equally well in both models. Indeed, for any election system that always has at least one winner when there is at least one candidate, the two models coincide, and almost all natural election systems have this property.

"preround," which cuts out about half the candidates, can tremendously boost a system's manipulation complexity over a broad range of systems.

Speaking more broadly, the problem that Narodytska and Walsh [38] and this paper are studying, for the case of runoffs and runoffs with revoting, is the manipulation problem. This asks whether a coalition of manipulators can ensure that a particular candidate is a winner of the overall election. The seminal work on the computational complexity of manipulation was that of Bartholdi, Tovey, and Trick [4] and Bartholdi and Orlin [3], and there have been many papers since studying manipulation algorithms for, and hardness results for, a variety of election systems, see, e.g., the surveys [6, 14, 19, 20, 24]. This entire stream exists within the area known as computational social choice [10].

Runoff elections can be viewed as a way of resolving at least some ties. There naturally are a number of papers in the literature focusing on aspects of ties and tiebreaking mechanisms, see, e.g., [1, 40, 41] and the references therein.

There is an interesting line of work of Meir et al. [36] and Lev and Rosenschein [34] studying in a fully game-theoretic setting iterated voting in the sense of seeing whether a Nash equilibrium is reached (see also [42, 43]). This work does not remove candidates after votes, and so is different in flavor and goal from our work.

Finally, we note that Guo and Shrestha [26], adopting the models of Narodytska and Walsh [38] and our work, have recently brought the study of the complexity of X THEN Y and X THEN X elections to the very different attack type known as electoral control.

3 Preliminaries

We first give a standard formalization of elections, the winner problem, and the manipulation problem. Each election instance will have a finite set, C, of candidates specified by their names, e.g., a particular election might have Obama and Romney as its candidates. Some election systems may treat candidates asymmetrically, and that will play a role in some of the proofs in this paper. Elections also have a finite collection of votes, which we will assume are input as a list of ballots, one per voter. Although social choice theory sometimes allows voters to have names, in this paper we study the most natural case—the one where votes come in nameless, and the election system's outcome depends on just C and what the multiset of votes is (and not on anything about voter names, and also the system cannot be dependent on the order in which the ballots happen to be listed in the input). We will refer to the collection of votes as V. The type of each vote will depend on the election system. Most systems require a linear ordering of the candidates, and that will be the case for all systems discussed in this paper.

So-called scoring protocols such as Plurality, Veto, Borda, and so on will for us have votes cast as linear orders. And then from those orders we will assign points to each candidate based on the rules of that scoring system. For example, in a veto election, each voter casts zero points for his or her least favorite candidate, and one point for each other candidate. In a plurality election, each voter casts one point for his or her favorite candidate, and zero points for each other candidate. In Triviality, each candidate gets zero points from each voter, and so all candidates tie as winners. In HalfApproval, if there are m candidates, each voter gives one point to each of the $\lceil m/2 \rceil$ top candidates in his or her linear order, and gives zero points to each other candidate. In any scoring system, all points for each candidate are added up, and the candidate(s) who have the maximum score achieved by any candidate are the winner(s). (When we speak of scoring protocols in the abstract, each scoring protocol must have a fixed number of candidates. However, when we say Plurality or

HalfApproval or so on, we usually are referring to the protocol that on m-candidate inputs uses the m-candidate Plurality or HalfApproval or so on scoring protocol mentioned above.) In Copeland$^\alpha$ elections [15], where α is a rational number between 0 and 1, winners are computed as follows. We look at the pairwise election between every pair of candidates. The candidate that wins gets a point. In case of a tie, both candidates get α points. The winner(s) are the candidates with the most points. Copeland^1 is also known as Llull [27].

An election system, X, is a mapping that given C and V outputs a set of candidates W ("the winner(s)"). This is precisely the definition of a social choice correspondence, as given in Shoham and Leyton-Brown [46].[3] All election systems that we construct to realize claims of our theorems have $W \neq \emptyset$ (when $C \neq \emptyset$), which in social choice is often part of the definition of an election system (see also the paragraph immediately before the start of Section 4.1.1). Our Section 4.2 in fact gives a broadly applicable way of paving over the difference between having and not having this condition. The winner problem for an election system X is the language that contains exactly those triples C, V, and $p \in C$ such that p is a winner in the X election on C and V. Although some well-known election systems exist whose winner problems are not in P (unless P $=$ NP) [5, 30, 32, 45], all the systems we study in this paper have P winner problems.

We now define the classic unweighted and weighted (coalitional) election manipulation problems, due to Conitzer, Sandholm, and Lang [13] (generalizing the noncoalitional unweighted case raised by Bartholdi, Tovey, and Trick [4]). The unweighted version, called Constructive Unweighted Coalitional Manipulation (CUCM), is defined as follows for any given election system X.

Name: X-CUCM.
Given: A set C of candidates, a collection V_1 of the nonmanipulative votes (each specified by a linear ordering over the candidates), a set V_2 of manipulative voters (since our voters do not have names, these are specified by a nonnegative integer input in unary[4] of giving the number of manipulative voters), and a distinguished candidate $p \in C$.
Question: Is there a way to set the votes of the manipulators, V_2, so that under the election system X, p is a winner of the election over candidate set C with the vote set being the ballots of the manipulators and the nonmanipulators?

The analogous weighted version, X-CWCM, is the same except each member of V_1 has both a weight and a linear order, and V_2 is specified as a list giving the weight of each manipulator (in binary). (The list is specifying a multiset.) The allowed range of weights is the positive integers. For each of the election systems we deal with in a weighted context, it will be immediately apparent what it means to use the election system on weighted voters.

Our interest here is in runoff elections. So in addition to the above classic versions, let us define versions with runoffs and with revoting runoffs. The "runoff" problems X-

[3] That definition and this paper allow, as do many papers in computational social choice theory, the case in which an election has no winners. We find that natural for symmetry with the case in which everyone wins. Also, there are real-world cases in which having no winner is natural, e.g., the system for electing players to the Baseball Hall of Fame is set up so that if the crop of candidates in a given year is weak no one will win. That has happened four times, most recently in the January 2013 vote, in which none of the 37 candidates were elected to the Hall.

[4] Unary is what best captures the flavor of each manipulator having a separate ballot (conceptually blank, initially). This also is consistent in flavor with the case—not ours—when voters are named, since there each of them has a separate entry in the input. Finally, purely as a matter of complexity, allowing the number of manipulators to be specified in binary would potentially exponentially distort sizes in an unnatural way.

CUCM-runoff and X-CWCM-runoff are the same as the above problems, except if after the X election there are two or more winners, a runoff election is conducted under X, with the candidates being just the winners of the initial election, and the votes of all voters (both manipulators and nonmanipulators) being their initial-election's preference-order vote, restricted to the remaining set of candidates. The "revoting runoff" (or for short, "revoting") problems X-CUCM-revoting and X-CWCM-revoting are the same as the above runoff problems, except if there is a runoff election, the manipulators may change their votes. And the question is, of course, whether in this setting there is a set of initial-round and, if needed, second-round manipulator votes that makes p a winner of the overall election.[5]

Note that all of these problems are defined as language problems, as is standard in the area. Typical complexities that they might take on are membership in P and NP-completeness. Those two cases are the focus of this paper and of most papers in this area. However, we mention in passing three related issues.

First, it has recently been pointed out that at least in some artificial cases, election decision problems can be in P even when their related search problems are NP-hard [29]. This worry does not infect any of this paper's results. Every result where we make a polynomial-time claim in this paper has the property that in polynomial time one can even produce the action(s) that achieve the desired outcome (such as making the given candidate win), i.e., our polynomial-time results are essentially what is sometimes called "certifiable," see Hemaspaandra, Hemaspaandra, and Rothe [31].[6]

Second, there has been much worry—and some empirical studies suggesting—that even NP-complete sets can be often easy. However, the following result shows that in a certain sense this cannot happen: If even one NP-hard set has a (deterministic) polynomial-time heuristic algorithm whose asymptotic error frequency is subexponential, then the polynomial hierarchy collapses (namely, to the class known as $\mathrm{S}_2^{\mathrm{NP}}$). The expository article of Hemaspaandra and Williams [33] provides a discussion of that result (which follows from combining results from [7] and [8]) and an attempt to reconcile the result with the good empirical results observed for hard problems. (Even for election problems, heuristics seem to often do very well, see, e.g., [24, 44, 47].) Our view is that the issue of proving rigorous results about the performance of heuristics on election problems is a highly difficult, highly important direction, but that NP-completeness results for a given problem are unquestionably an excellent indication that polynomial-time algorithms, and even polynomial-time heuristics with subexponential error rates, cannot be reasonably expected.

[5] To make this crystal clear, we give an in-math definition of the X-CUCM-runoff notion described in words above, namely, given C, V_1, V_2, and p, the question is whether $(\exists R)[\{p\} = X(C, V_1 \cup R) \vee (\|X(C, V_1 \cup R)\| > 1 \wedge p \in X(X(C, V_1 \cup R), (V_1 \cup R)_{X(C,V_1 \cup R)}))]$, where the subscript denotes masking those votes down to the specified candidates, the unions are multiset-like, and R is an assignment of votes to the manipulative voter set V_2. And similarly, X-CUCM-revoting is the same except with the question (and the new variable S is over assignments of manipulative voters to V_2) being $(\exists R)(\exists S)[\{p\} = X(C, V_1 \cup R) \vee (\|X(C, V_1 \cup R)\| > 1 \wedge p \in X(X(C, V_1 \cup R), (V_1 \cup S)_{X(C,V_1 \cup R)}))]$.

[6] For the case of revoting runoffs, the natural model here, in terms of seeking a polynomial-time certificate- (i.e., action-) yielding algorithms, is to allow the manipulative coalition, before the runoff election, a full view of all the initial votes and candidates, and of the outcome of that election, and to require that they set their votes in polynomial time, and of course to also require that their initial-election vote-setting be done in polynomial time. However, since all the election systems in this paper have polynomial-time winner problems, after a given set of initial-round votes the manipulators can themselves compute who the initial-round winner(s) are, and so for problems with polynomial-time winner algorithms, one can w.l.o.g. require the manipulative coalition to fork over at the same time both of its rounds of votes.

Third, we mention that in our model, as is standard in this area, the manipulators are given access to the votes of the nonmanipulators. This is a strong though standard assumption, and admittedly is a model for study rather than a perfect image of the real world. The model actually makes the NP-hardness results stronger (since they say that even with full information the problem remains intractable) and most of our results are NP-hardness results.

4 Results

We now turn to our results regarding the complexity of the manipulation problem for elections, for elections with runoffs, and for elections with revoting runoffs.

Our results are of two basic sorts. First, we are interested in what *can* happen. That is, for those three manipulation complexities, what is the relationship between them? Is there any connection at all?

We show that there is no connection that holds globally. Even when limiting ourselves just to election systems with P winner problems, we prove that every possible case of P-or-NP-complete can simultaneously hold for these three complexities: Each of the eight weighted and eight unweighted possibilities can be realized. The reason we want to know what *can* happen is because it is important to know the universe of behaviors that one may face. To us, the importance of understanding what can happen and what, if anything, is inherently impossible, is so great that such results are interesting and valuable even if one has to—as we at times do—use artificial systems to show that a case can happen. Note that since our runoff and revoting problems must have the same system used in the initial and runoff rounds, the result we mention (Theorem 1) does not follow from the important work of Narodytska and Walsh [38] realizing all possibilities for X THEN Y.[7]

Our second type of result regards what *does* happen for the most famous, important, natural systems. For example, although we show that, perhaps counterintuitively, runoffs and revoting runoffs can sometimes lower complexity and can have other bizarre relative complexities, for none of the natural, concrete systems we have looked at do we find this behavior to occur. For each concrete, natural system we have studied, runoffs and revoting runoffs either leave the manipulation complexity unchanged, or increase the manipulation complexity. Of course, our results on what *does* happen for concrete systems prove some of the cases of our claims regarding what *can* happen.

4.1 Realizability theorem

The following theorem states our result about what can happen, namely, regarding P and NP-completeness, any possible triple of complexities can occur.

Theorem 1 *Let* NPC *denote* "NP-*complete.*" *Let* W $=\{$ (P, P, P), (P, P, NPC), (P, NPC, P), (P, NPC, NPC), (NPC, P, P), (NPC, P, NPC), (NPC, NPC, P), (NPC, NPC, NPC)$\}$.

[7] Our complexity results regarding revoting, which first appeared in a January 2013 technical report on this work [25], appeared after the 2012 paper of Narodytska and Walsh [37] that suggested the study of revoting but that didn't provide complexity results on that topic; our complexity results regarding revoting appeared before the 2013 version [38] that has complexity results on revoting; that is, those results were obtained separately.

1. *For each element w of W, there exists an election system X, whose winner problem is in* P*, such that the complexity of X*-CUCM*, X*-CUCM-runoff*, and X*-CUCM-revoting *is, respectively, the three fields of w.*
2. *The analogous result holds for the weighted case (where the three fields will capture the complexity of X*-CWCM*, X*-CWCM-runoff*, and X*-CWCM-revoting*, respectively).*

Recall that we promised that each election system that we construct to realize claims in our theorems will always have at least one winner (if there is at least one candidate). The election systems that we build in this theorem will satisfy that also. However, along the path to building such a system X, we will often first build an election system Y that is allowed to have everyone lose, and that satisfies the theorem *in the model where there is always a runoff, even where there is a unique winner in the initial round*. We then set $X = AlwaysWinners(Y)$, where *AlwaysWinners* is as described in Section 4.2. Then X, which always has at least one winner (when there is at least one candidate), also satisfies the conditions of the theorem in the model where there is always a runoff. And since X always has at least one winner, X also satisfies the conditions of the theorem in our standard model, where when there is a unique winner in the initial round, this candidate is a winner of the overall election and there is no runoff.

4.1.1 Unweighted cases of the realizability theorem

We briefly mention that the two most interesting unweighted cases in the realizability theorem are the ones realizing the cases (NPC, P, NPC) in Theorem 6 and, especially, (NPC, NPC, P) in Theorem 9. The key twist in these is that both create a setting in which an election system can in effect pass messages to its own second-round self through the winner set and with the help of the manipulators. In particular, in a certain set of circumstances, the election system can be made to, in effect, know that “If the input I’m seeing is taking place in a second round (although I cannot myself tell whether or not it is), then we are utterly certainly in a model in which revoting is allowed and indeed in which one of the manipulators has changed his or her vote since the initial round.”

We now present (some here and some deferred to Appendix A.1) the proofs of the eight unweighted cases of Theorem 1. Most of the constructions used in these cases make use of a function $f(\cdot)$ described below. $f(\ell)$ will give the least number of candidates such that the set of votes over $f(\ell)$ candidates can clearly express at least ℓ possibilities. Since there are $m!$ linear orderings over m candidates, and given an ordering it is easy to tell its lexicographic rank among all the linear orderings over m candidates, and given a rank it is easy to find which ordering (if any) has that lexicographic rank among all the linear orderings over m candidates, we have that $f(\ell)$ is the least integer k such that $k! \geq \ell$.

Definition 1 Let $f(\ell)$ be the least integer k such that $k! \geq \ell$.

Theorem 2 *There exists an election system X, with a polynomial-time winner problem, such that X*-CUCM*, X*-CUCM-runoff*, and X*-CUCM-revoting *are all* NP-*complete.*

Proof As mentioned earlier, Narodytska and Walsh [38] show that STV′-CUCM and STV′-CUCM-runoff are NP-hard, where STV′ is the decisive version of STV where ties are broken in favor of the manipulators. As pointed out in [38], this follows because STV′ =STV′ THEN STV′. It is also immediate that STV′ = STV′ THEN STV′ (with revoting).

However, note that STV′ is technically not an election system, since an election system is blind to who is the preferred candidate. Since our theorem requires X to be an election system, we will introduce and use a close cousin of STV′, which we will call STV″, that *is* an election system.

Let STV″ be the decisive version of STV where ties are broken in favor of the lexicographically least candidate. Note that it is easy to reduce STV-CUCM, which is NP-complete [3], to STV″-CUCM. We simply need to ensure that the preferred candidate's name is lexicographically first in the image of the reduction. Note that STV″ behaves like STV′ in that STV″ = STV″ THEN STV″ and STV″ = STV″ THEN STV″ (with revoting). This establishes Theorem 2 for $X = \text{STV}''$. □

Theorem 3 *There exists an election system X, with a polynomial-time winner problem, such that X*-CUCM*, X*-CUCM-runoff*, and X*-CUCM-revoting *are all in* P.

Proof Let X = Plurality. Since the manipulators want to maximize the plurality score of their preferred candidate, p, their optimal action is to put p at the top of their orderings. It does not benefit the manipulators to put any other candidate higher in their orderings or to change their vote in the runoff. □

Theorem 4 *There exists an election system X, with a polynomial-time winner problem, such that X*-CUCM *is in* P *and both X*-CUCM-runoff *and X*-CUCM-revoting *are* NP-*complete.*

Proof Let X = Veto. The result now follows directly from Theorem 18, since it is well-known that Veto-CUCM is in P. □

Theorem 5 *There exists an election system X, with a polynomial-time winner problem, such that X*-CUCM *and X*-CUCM-runoff *are in* P*, but X*-CUCM-revoting *is* NP-*complete.*

Proof Let $f(\cdot)$ be as specified in Definition 1. Let $X = \mathit{AlwaysWinners}(Y)$, where Y is defined as follows:

If $\|V\| \neq 1$ then everyone loses.[8]

If $\|C\| = 2$ and the lexicographically smaller is ranked above the lexicographically larger candidate by the voter then everyone wins.

If $\|C\| \geq 4$, the lexicographically smallest candidate's name encodes a formula ψ, and $\|C\| \geq f(2^{\#\text{vars}(\psi)}) + 1$ (the "+1" is as we'll not involve the lexicographically smallest candidate in encoding the assignment), then we do the following. If the voter ranks the lexicographically largest above the lexicographically smallest candidate and on all candidates other than the lexicographically smallest it codes an assignment that satisfies ψ then the lexicographically smallest and largest candidates win. Otherwise, the lexicographically smallest, second smallest, and largest win.

Otherwise, everyone loses.

[8] We could remove the restriction that we have a single voter by always allowing an arbitrary number of voters, but requiring that all but at most one of the voters cast a vote that encodes a message to ignore their vote.

That completes the specification of Y.

Election system Y is admittedly artificial, as are some of our other constructions. We explained in the third paragraph of Section 4 why results obtained through artificial constructions can be interesting and valuable.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly the winner problem for Y is in P.
- Y-CUCM is in P since a preferred candidate p can always be made a winner if $\|C\| = 2$ and there is one manipulator and no other voters. If $\|C\| \geq 4$, the optimal action for the manipulator is to rank the lexicographically smallest candidate above the lexicographically largest candidate. In all other cases, no candidate can win.
- Y-CUCM-runoff is in P by the following argument. It suffices to consider the case where there is one manipulator and no other voters. If the election has two candidates, the manipulator must rank the lexicographically smaller above the lexicographically larger candidate so that both candidates win the initial round and runoff. If $||C|| = 3$, no candidate can win. If $||C|| \geq 4$, then there are zero, two, or three winners in the initial round. If there are three winners in the initial round, there are no winners in the runoff. If there are two winners in the initial round, then the lexicographically largest candidate is ranked above the lexicographically smallest candidate. But then there are also no winners in the runoff without revoting.
- Y-CUCM-revoting is NP-complete by reducing from SAT. Observe that $\psi \in$ SAT reduces to Y-CUCM-revoting for the candidate set: p encoding ψ and $\max(3, f(2^{\#\mathrm{vars}(\psi)}))$ dummy candidates all lexicographically larger than p. Let the preferred candidate be p and let there be zero nonmanipulative voters and one manipulative voter. In order for p to win the runoff, there have to be two winners in the initial round. This implies that ψ is satisfiable. And if ψ is satisfiable, p can be made a winner by having the manipulator vote so that p is ranked last and its ordering over the dummy candidates codes a satisfying assignment to ψ. In the runoff the manipulator must change her vote so that p is ranked first.

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

The proofs of Theorems 6–9 are deferred to Appendix A.1.

Theorem 6 *There exists an election system X, with a polynomial-time winner problem, such that X-*CUCM *is* NP-*complete,* X-CUCM-runoff *is in* P*, and* X-CUCM-revoting *is* NP-*complete.*

Theorem 7 *There exists an election system X, with a polynomial-time winner problem, such that X-*CUCM *is* NP-*complete, but* X-CUCM-runoff *and* X-CUCM-revoting *are in* P.

Theorem 8 *There exists an election system X, with a polynomial-time winner problem, such that X-*CUCM *is in* P*,* X-CUCM-runoff *is* NP-*complete, and* X-CUCM-revoting *is in* P.

Theorem 9 *There exists an election system X, with a polynomial-time winner problem, such that X-*CUCM *and* X-CUCM-runoff *are* NP-*complete, but* X-CUCM-revoting *is in* P.

4.1.2 Weighted cases of the realizability theorem

We now present (some here and some deferred to Appendix A.2) the proofs of the eight weighted cases of Theorem 1. As is typical for NP-hardness proofs for weighted manipulation, we will usually reduce from the well-known NP-complete problem Partition: Given a nonempty set of positive integers $k_1, \dots, k_t$ that sums to $2K$, we ask if $k_1, \dots, k_t$ can be partitioned into two subsets of equal size. (Formally, Partition is taken as the set of strings that encode sets of integers having the above properties, using a natural coding scheme with the appropriate simplicity properties. However, as is typical and clear, we will simply write things such as $k_1, \dots, k_t \in$ Partition in our proofs.)

Theorem 10 *There exists an election system X, with a polynomial-time winner problem, such that X-*CWCM*, X-*CWCM-runoff*, and X-*CWCM-revoting *are all* NP-*complete.*

Proof Let X be Veto. The result now follows directly from [28] and Theorem 23. □

Theorem 11 *There exists an election system X, with a polynomial-time winner problem, such that X-*CWCM*, X-*CWCM-runoff*, and X-*CWCM-revoting *are all in* P.

Proof Let X be Plurality. The result now follows directly from Theorem 21. □

Theorem 12 *There exists an election system X, with a polynomial-time winner problem, such that X-*CWCM *is in* P *and both X-*CWCM-runoff *and X-*CWCM-revoting *are* NP-*complete.*

Proof Let X be defined as follows: If $\|C\| \leq 4$ then use Llull, otherwise everyone wins. The result now follows directly from [23] and Theorem 25. □

Theorem 13 *There exists an election system X, with a polynomial-time winner problem, such that X-*CWCM *and X-*CWCM-runoff *are in* P*, but X-*CWCM-revoting *is* NP-*complete.*

Proof Let $X = AlwaysWinners(Y)$, where Y is defined as follows:

If $\|C\| = 3$ then candidates with plurality scores $\geq 50\%$ win.
If $\|C\| = 2$ then candidates with plurality scores $> 50\%$ win.

Otherwise, everyone loses.
That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly the winner problem for Y is in P.
- Y-CWCM is in P since in all cases where a preferred candidate can win, the optimal action for the manipulators is to vote for that candidate.
- Y-CWCM-runoff is in P since the only way to have winners in a two-stage Y election is to have three candidates in the initial round and two candidates in the runoff. No candidates can win in the runoff unless revoting is allowed, so Y-CWCM-runoff is in P.

– Y-CWCM-revoting is NP-complete since to test if a set of positive integers $k_1, \ldots, k_t \in$ Partition we can ask if the candidate p can win the Y-CWCM-revoting instance with the candidate set $\{p, r, \ell\}$ and the voter set containing zero nonmanipulative voters and t manipulative voters. Let the t manipulators have weights that correspond to the Partition instance, i.e., $k_1, \ldots, k_t$ such that $\sum_{i=1}^{t} k_i = 2K$.

  To ensure that p is an overall winner, the manipulators must partition their votes between p and w.l.o.g. r, since the only way to have a runoff where a candidate can win is when two candidates win in the initial round. Then in the runoff the manipulators all vote for p. Thus if there is a way for the manipulators to partition their votes into two equal-weight subsets, then p can be made the overall winner. Otherwise, p cannot be made the overall winner. Therefore Y-CWCM-revoting is NP-complete.

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

The proofs of Theorems 14–17 are deferred to Appendix A.2.

Theorem 14 *There exists an election system X, with a polynomial-time winner problem, such that* X-CWCM *is* NP-*complete,* X-CWCM-runoff *is in* P*, and* X-CWCM-revoting *is* NP-*complete.*

Theorem 15 *There exists an election system X, with a polynomial-time winner problem, such that* X-CWCM *is* NP-*complete, but* X-CWCM-runoff *and* X-CWCM-revoting *are in* P.

Theorem 16 *There exists an election system X, with a polynomial-time winner problem, such that* X-CWCM *is in* P*,* X-CWCM-runoff *is* NP-*complete, and* X-CWCM-revoting *is in* P.

Theorem 17 *There exists an election system X, with a polynomial-time winner problem, such that* X-CWCM *and* X-CWCM-runoff *are* NP-*complete, but* X-CWCM-revoting *is in* P.

So we now (in concert with the proofs deferred to the Appendix) have completed both the unweighted and weighted cases of the realizability theorem, Theorem 1. That is, for both the unweighted and weighted cases, we have shown that there are election systems, having polynomial-time winner problems, such that the respective complexities of manipulation, manipulation with runoffs, and manipulation with revoting runoffs are (P, P, P), (P, P, NPC), (P, NPC, P), (P, NPC, NPC), (NPC, P, P), (NPC, P, NPC), (NPC, NPC, P), and (NPC, NPC, NPC). That is, regarding membership in P and NP-completeness, our eight-case unweighted analysis and our eight-case weighted analysis show that these three problems do take on every possible combination. We commend as an issue for future study a sixty-four-case analysis in which every sextuple of unweighted-plus-weighted cases is studied. Note that for that case, since each weighted case is at least as hard as its unweighted case, some situations will be impossible unless P = NP. For example, if the CUCM complexity of a system is NP-hard, then so is its CWCM complexity, and so the latter clearly cannot be in P unless P = NP. Thus, unless P = NP, not all sixty-four cases can be realized.

4.2 The *AlwaysWinners* construction

We now describe the transformation *AlwaysWinners* that was used in the proofs of many of the cases of Theorem 1. What this transformation will do is it will take an election system and will transform it into a new election system that will (except when the candidate set is empty) always have at least one winner, yet that is so closely related to the original election system that the complexity of the original system is unchanged, with respect to the three problems we are concerned with here *in the model where there is always a runoff, even where there is a unique winner in the initial round* (and in fact, with respect to many, many other types of manipulative actions). Note that once we have an election system that will always have at least one winner, the model in which there is always a runoff coincides with our standard model.

This transformation thus relatively broadly addresses a persistently annoying issue. Many people, including the authors, feel that it is ugly and asymmetric to require nonempty winner sets yet to allow all of C to be the winner set. And indeed a number of papers in *computational* social choice do allow the winner set to be any member of 2^C. On the other hand, traditionally in *social choice*, the requirement that the winner set be nonempty is part of the definition of elections. Our transformation shows that the difference in models isn't as large as one might think; one can often adopt the more symmetric model, yet by this transformation one will know that one's results also hold in the more restrictive model.

Although *AlwaysWinners* outputs systems that are artificial, we feel the transformation is of real interest. This is so because this transformation allows the better understanding of what is and is not possible on an issue that as we mention above, and as we discussed in the final paragraph of the Introduction, is important and has not previously been addressed.

We now give the transformation *AlwaysWinners*. First let us describe informally how it works, and then we'll describe it and its properties more formally. Let $\mathscr{E}$ be the election system we want to transform under *AlwaysWinners*. Informally, suppose that we have a candidate name, new, that is not part of our universe of legal candidate names. (Of course this is untrue; the universe of names is as it is. But please indulge us for a few more lines, and we'll then avoid this problem in our more formal approach.) Then under *AlwaysWinners*, if new is not an input candidate then all candidates win, and if new is an input candidate then new wins and also all candidates win who under $\mathscr{E}$ would win if in our input election new is removed and all votes are masked down to remove new. Note that this system always has a winner (if the set of candidates is not empty). Also, it can be easily seen to retain most of the manipulative-action complexities of $\mathscr{E}$, as we'll discuss later.

As admitted above, we can't just make a new name appear. But we can get the same effect formally, by shifting all the names in the universe up by one spot to open up a space for our new name, and then when we're using those other names in simulating the original system, by shifting them back down again. And that is precisely what we will do.

So we now more carefully and correctly specify the transformation *AlwaysWinners*.

Definition 2 Let $\mathscr{E}$ be an election system (that perhaps has no winners even on inputs on which C is nonempty). Let the set of legal names for candidates (i.e., the set of all strings) be enumerated in lexicographic order by $s_0, s_1, s_2, \ldots$ (e.g., "ε, 0, 1, 00, 01, ..." if names are taken to be binary strings). Let $++$ denote a one-step increase in this order, i.e., $s_i{++} = s_{i+1}$. The $++$ operator naturally applies to sets of candidates, namely as defined by $A{++} = \{a{++} \mid a \in A\}$. And for any set A of candidates such that $s_0 \notin A$, we similarly define the decrement of the set, namely by $A{-}{-} = \{a \mid a{++} \in A\}$. On candidate set C and voter set V, *AlwaysWinners*($\mathscr{E}$) does the following. If $s_0 \notin C$, then the winner set is C. If $s_0 \in C$ then the winner set is $\mathscr{E}((C - \{s_0\})\text{--}, V')\text{++} \cup \{s_0\}$, where V' is V with s_0 masked out of each preference order and then each candidate name decremented in each order and where $\mathscr{E}(\hat{C}, \hat{V})$ denotes the winner set, under $\mathscr{E}$, of the election over candidate set $\hat{C}$ and voter set $\hat{V}$.

The crucial things to notice about *AlwaysWinners*($\mathscr{E}$) are the following, which hold for all election systems $\mathscr{E}$ (including ones that allow there to be no winners on some inputs for which the input candidate set of the $\mathscr{E}$ instance is nonempty) *in the model where there is always a runoff*. *AlwaysWinners*($\mathscr{E}$) always has at least one winner (when the input candidate set of the *AlwaysWinners*($\mathscr{E}$) instance is nonempty). For $\mathscr{E}$-CUCM (respectively, $\mathscr{E}$-CUCM-runoff, $\mathscr{E}$-CUCM-revoting) it holds that if the problem is in P then *AlwaysWinners*($\mathscr{E}$)-CUCM (respectively, *AlwaysWinners*($\mathscr{E}$)-CUCM-runoff, *AlwaysWinners*($\mathscr{E}$)-CUCM-revoting) is in P. For $\mathscr{E}$-CUCM (respectively, $\mathscr{E}$-CUCM-runoff, $\mathscr{E}$-CUCM-revoting) it holds that if the problem is NP-complete then *AlwaysWinners*($\mathscr{E}$)-CUCM (respectively, *AlwaysWinners*($\mathscr{E}$)-CUCM-runoff, *AlwaysWinners*($\mathscr{E}$)-CUCM-revoting) is NP-complete. Part of the easy task of seeing that these complexity connections hold is noticing that given an instance of one of these problems under $\mathscr{E}$, one can increment all candidate names both within the candidate set and the voter preferences, can then add in a new candidate s_0 and extend voter preferences arbitrarily to include that new candidate (e.g., putting it last in each voter's preferences), and then we can note that a candidate p can be made a winner in the initial election under $\mathscr{E}$ exactly if p++ can be made a winner under *AlwaysWinners*($\mathscr{E}$) in the transformed election. (And we mention in passing that the analogous claim holds for the so-called "destructive" case in which we seek to preclude p from being a winner, though destructive cases are not a focus of this paper.)

The observations above are what we need to conclude that, for the election systems Y built in the proofs of many of the cases of Theorem 1, *AlwaysWinners*(Y) satisfies each theorem in the model where there is always a runoff, and has the property that it always has a winner (when the candidate set is nonempty). Since our standard model and the model where there is always a runoff are the same for election systems that always have a winner, *AlwaysWinners*(Y) also satisfies the conditions of the theorem in our standard model.

However, we comment that the above transformation will be useful in the exact same way for many types of manipulative attacks other than the three discussed above. It in fact will similarly work (keeping in mind that we are always in the so-called nonunique-winner model—aka the co-winner model—which focuses on whether a given candidate is/is not *a* winner) for all standard types of voter control (adding/deleting/partitioning), all standard types of manipulation, and all standard types of bribery. Thus, the above transformation goes quite far in paving over the divide between those who feel that requiring nonempty winner sets is unnatural and those who feel that failing to require nonempty winner sets is unnatural.

4.3 Specific voting rules and scoring protocols

Theorem 1 showed that regarding P and NP-completeness, any possible triple of complexities can occur, in both the unweighted case and the weighted case. In this section, we look at what happens for the most famous, important, natural systems. For each concrete, natural system we have studied, runoffs and revoting runoffs either both leave the manipulation complexity unchanged, or both increase the manipulation complexity.

4.3.1 *Unweighted specific voting rules where the complexity increases*

The following result provides—if we keep in mind that it is well-known that Veto-CUCM is in P—a natural, unweighted case where the classic manipulation problem is simple but the runoff and revoting runoff versions are hard. Even Narodytska and Walsh's [38] work, in which they allowed themselves the freedom to use different systems in the first and second round, did not obtain any natural, unweighted example of runoffs or revoting runoffs increasing complexity.

Theorem 18 Veto-CUCM-runoff *and* Veto-CUCM-revoting *are each* NP-*complete.*

Proof We will reduce from the well-known NP-complete Exact Cover by 3-Sets Problem (X3C): Given a set $B = \{b_1, \ldots, b_{3k}\}$, and a collection $\mathscr{S} = \{S_1, \ldots, S_n\}$ of three-element subsets of B, we ask if $\mathscr{S}$ has an exact cover for B, i.e., if there exists a subcollection $\mathscr{S}'$ of $\mathscr{S}$ such that every element of B occurs in exactly one member of $\mathscr{S}'$. Without loss of generality, we assume that $k \geq 1$ and $n \geq 3$. We will denote which elements of B are in a given S_i by some new i_j variables: $S_i = \{b_{i_1}, b_{i_2}, b_{i_3}\}$.

Since Veto-CUCM is in P (simply greedily veto all candidates that score higher than p), the only place where hardness can come in is in the selection of the set of winners in the initial round.

Our election has the following set C of candidates: p (the preferred candidate), $b_1, \ldots, b_{3k}$ and $s_1, \ldots, s_n$ (candidates corresponding to the X3C instance), $r_1, \ldots, r_k$ (candidates that will be vetoed in the runoff), d (a buffer candidate), and ℓ (a candidate that always loses in the initial round). We have k manipulators. We have the following nonmanipulators. (When we use "$\cdots$" in a vote, this denotes that the order of the remaining candidates does not matter, and for specificity, let us say we will put them in lexicographic order.)

- For every i, $1 \leq i \leq n$,
  - one nonmanipulator voting $\cdots > p > b_{i_1} > s_i$,
  - one nonmanipulator voting $\cdots > p > b_{i_2} > s_i$,
  - and one nonmanipulator voting $\cdots > p > b_{i_3} > s_i$.
- Three nonmanipulators voting $\cdots > p$.
- For every $c \in B \cup \{r_1, \ldots, r_k\} \cup \{d\}$, three nonmanipulators voting $\cdots > p > c$.
- One nonmanipulator voting $\cdots > p > \ell$.
- For every i, $1 \leq i \leq n$, one nonmanipulator voting $\cdots > p > d > s_i > \ell$.

Note that every candidate other than ℓ receives three vetoes from the nonmanipulators in the initial round. And ℓ receives $n + 1 > 3$ vetoes from the nonmanipulators.

Let $\mathscr{S}' = \{S_{j_1}, \ldots, S_{j_k}\} \subseteq \mathscr{S}$ be an exact cover for B. For $1 \leq i \leq k$, let the ith manipulator vote $\cdots > r_i > s_{j_i}$. We claim that p is a winner of the overall election (even without revoting). It is immediate that the winner set of the initial round is $C - \{\ell\} - \{s_j \mid S_j \in \mathscr{S}'\}$. Since ℓ does not participate in the runoff, p gains one veto from the nonmanipulator voting $\cdots > p > \ell$. Since every candidate in $B \cup \{r_1, \ldots, r_k\} \cup \{d\}$ participates in the runoff, this is the only veto that p gains. Since ℓ does not participate in the runoff, each s_i that participates in the runoff gains one veto from the nonmanipulator voting $\cdots > d > s_i > \ell$. d gains $k \geq 1$ vetoes from the nonmanipulators voting $\cdots > d > s_i > \ell$ such that $S_i \in \mathscr{S}'$ and every $b \in B$ gains one veto from the nonmanipulator voting $\cdots > p > b > s_i$ such that $b \in S_i$ and $S_i \in \mathscr{S}'$. Every candidate r_i gains a veto from the manipulator voting $\cdots > r_i > s_{j_i}$. It follows that p is a winner in the runoff.

For the converse, we will show the manipulations described above are the only way to make p a winner. Suppose the manipulators can vote (in the initial round and the runoff) in such a way that p becomes a winner of the overall election. Recall that in the initial round, every candidate other that ℓ receives three vetoes from the nonmanipulators and that ℓ receives $n + 1 > 3$ vetoes. Since there are k manipulators, ℓ does not participate in the runoff and at most k other candidates (those vetoed by at least one of our k manipulators) do not participate in the runoff. Since ℓ does not participate in the runoff, p gains one veto from the nonmanipulator voting $\cdots > p > \ell$.

Suppose there is a candidate $c \in B \cup \{r_1, \ldots, r_k\} \cup \{d\}$ that does not participate in the runoff. Then p gains three vetoes from the nonmanipulators voting $\cdots > p > c$, and thus p receives at least seven vetoes in the runoff. There are at least $2k$ candidates from B that participate in the runoff and each of these candidates is vetoed three times in the initial round and does not gain any vetoes from deleting ℓ. Since p receives at least seven vetoes in the runoff, each candidate in B that participates in the runoff needs to gain at least four vetoes, so these candidates need to gain a total of at least $8k$ vetoes. But the most vetoes that these candidates can gain in total is three vetoes for each candidate s_i that does not participate in the runoff plus k vetoes from the manipulators. Since fewer than k s_i candidates do not participate in the runoff, the B candidates that participate in the runoff gain a total of at most $4k$ vetoes, which is not enough.

It follows that the only candidates other than ℓ that do not participate in the runoff are s_i candidates. Note that candidates in $\{r_1, \ldots, r_k\}$ will not gain vetoes from the nonmanipulators in the runoff, and so each manipulator needs to veto exactly one r_i in the runoff. To make sure that every candidate $b \in B$ gains at least one veto, we need to delete a set of s_i candidates corresponding to a cover. Since we can delete at most k such candidates, these candidates will correspond to an exact cover. □

The case of just one manipulator is a natural and important case. It also can often be surprisingly well handled, thanks to the lovely result—initially stated for the unweighted case in the seminal work of Bartholdi, Tovey, and Trick [4] and since then used or extended in various settings, e.g., [17, 49]—that the natural (polynomial-time) greedy manipulation algorithm (giving one's highest point value to p and then giving, in turn, the highest remaining value to the candidate who has the lowest point total among those not yet assigned points by the manipulative voter) is optimal (i.e., finds a successful manipulation when one exists) for both weighted and unweighted scoring protocols, for the case when there is just one manipulator. The following theorem implies that that result does not carry over to runoff elections.

Theorem 19 *The standard one-manipulator polynomial-time greedy algorithm for scoring protocols is not optimal for* HalfApproval-CUCM-runoff *and* HalfApproval-CUCM-revoting*, restricted to at most one manipulator.*

Proof Consider the election with candidate set $\{p, a, b, c\}$, a nonmanipulator voting $a > p > b > c$, a nonmanipulator voting $a > b > p > c$, and one manipulator. The scores of p, a, b, c from the nonmanipulators are $1, 2, 1, 0$. The greedy algorithm would give the following vote for the manipulator: $p > c > b > a$. Then p and a are the winners of the initial round, and there is no way for p to win the runoff. However, if the manipulator votes

$p > b > c > a$, then p, a, and b are the winners of the initial round, and p is a winner of the runoff (even without revoting). $\square$

Theorem 18 gave a case where a simple-to-manipulate unweighted scoring protocol became hard for runoffs, with or without revoting. The following result gives a new example of runoffs increasing complexity, this time for the one-manipulator case. It is natural to wonder whether the following theorem itself implies Theorem 19. The answer is that the following theorem does not imply that, but it does imply something a bit weaker, namely, it says that Theorem 19 holds unless P = NP. (Of course, Theorem 19 holds absolutely; it doesn't require a P $\neq$ NP hypothesis.) HalfApproval-CUCM for one manipulator is clearly in P—for example by the greedy algorithm we mentioned above—and so the following result does express a raising of complexity. Even HalfApproval-CUCM (i.e., with an unbounded number of manipulators) is in P (this follows from the argument of [48, Corollary 4.2]). So this gives even more of a contrast between classic and runoff manipulation complexity.

Theorem 20 HalfApproval-CUCM-runoff *and* HalfApproval-CUCM-revoting *are each* NP-*complete, even when restricted to having at most one manipulator.*

Proof This construction operates similarly to the construction from Theorem 18. Note that we have fewer candidates in the runoff than in the initial round, and so in contrast to the proof of Theorem 18, the manipulator has fewer vetoes to contribute in the runoff and some candidates may have fewer vetoes in the runoff than they had in the initial round. We must be careful of how these two points affect our construction. We reduce from Exact Cover by 3-Sets (X3C): Given a set $B = \{b_1, \ldots, b_{3k}\}$, and a collection $\mathscr{S} = \{S_1, \ldots, S_n\}$ of three-element subsets of B, we ask if $\mathscr{S}$ has an exact cover for B, i.e., if there exists a subcollection $\mathscr{S}'$ of $\mathscr{S}$ such that every element of B occurs in exactly one member of $\mathscr{S}'$. We will denote which elements of B are in a given S_i by some new i_j variables: $S_i = \{b_{i_1}, b_{i_2}, b_{i_3}\}$. Without loss of generality we assume that $n \geq 3$ and $n \geq k \geq 1$.

We pad the election from the proof of Theorem 18. We have the following candidates: p (the preferred candidate), $B = \{b_1, \ldots, b_{3k}\}$ and $S = \{s_1, \ldots, s_n\}$ (corresponding to the X3C instance), $R = \{r_1, \ldots, r_{n+2k}\}$ (candidates that will pad the votes in the runoff), d (a buffer candidate), and $L = \{\ell_1, \ldots, \ell_{2n+5k}\}$ and $\hat{\ell}$ (candidates that will lose in the initial round). Summarizing, this gives us the candidate set $C = \{p, d, \hat{\ell}\} \cup B \cup S \cup R \cup L$. We have $\|C\| = 4n + 10k + 3$, so each vote approves $\lceil \|C\|/2 \rceil = 2n + 5k + 2$ candidates and vetoes $2n + 5k + 1$ candidates. Observe that the number of vetoes contributed by a vote is equal to $\|L\| + 1$, which is crucial to pad our construction. We have the following nonmanipulators. (A set in the specification of a vote, e.g., R in the first vote specified below, denotes that candidates in that set are in arbitrary order in that part of the vote.)

- For every i, $1 \leq i \leq n$,
  - one nonmanipulator voting $\cdots > B - \{b_{i_1}\} > p > b_{i_1} > R > s_i > L$,
  - one nonmanipulator voting $\cdots > B - \{b_{i_2}\} > p > b_{i_2} > R > s_i > L$,
  - one nonmanipulator voting $\cdots > B - \{b_{i_3}\} > p > b_{i_3} > R > s_i > L$,
  - and one nonmanipulator voting $\cdots > B > d > R > s_i > \hat{\ell} > L$.
- For every i, $1 \leq i \leq 3k$,
  - two nonmanipulators voting $\cdots > B - \{b_i\} > p > R > b_i > L$.

- For every i, $1 \leq i \leq n+2k$,
  - two nonmanipulators voting $\cdots > B > p > d > R - \{r_i\} > r_i > L$.
- Two nonmanipulators voting $\cdots > B > R > d > L$.
- Three nonmanipulators voting $\cdots > B > R > p > L$.
- One nonmanipulator voting $\cdots > B > R > p > \hat{\ell} > L$.

The votes of the nonmanipulators result in the following vetoes for each of our candidates. (We count vetoes since the main idea of this proof is similar to the proof of Theorem 18 and since we specify enough of each of the votes to clearly show all possibilities for the initial round and the runoff.) Recall that the number of vetoes contributed by each vote in the initial round is $\|L\| + 1$.

- p has three vetoes.
- Each candidate in S has three vetoes.
- Each candidate in $B \cup R \cup \{d\}$ has two vetoes.
- Each candidate in L has more than three vetoes.
- $\hat{\ell}$ has more than three vetoes (since $n \geq 3$).

Let $\mathscr{S}' \subseteq \mathscr{S}$ be an exact cover for B. We will show that if the manipulator votes

$$\cdots > L > \{s_i | S_i \in \mathscr{S}'\} > R > d > B$$

then p will be a winner of the overall election. Note that the candidate set of the runoff is $C' = \{p, d\} \cup B \cup \{s_i | S_i \notin \mathscr{S}'\} \cup R$ and $\|C'\| = 2n + 4k + 2$. Again we have a set of candidates that is crucial to pad the votes in the runoff. Now $\|R\|+1$ is the number of vetoes in the runoff not unlike when $\|L\| + 1$ was the number of vetoes in the initial round. In the runoff, because of the absence of the candidates in $L \cup \{\hat{\ell}\}$ and the candidates corresponding to the exact cover, and because of the vote of the manipulator, the remaining candidates have the following vetoes.

- p has four vetoes.
- Each candidate in S that participates in the runoff has four vetoes.
- Each candidate in B has four vetoes.
- Each candidate in $\{d\} \cup R$ has more than four vetoes.

So, p is a winner in the runoff (even without revoting).

For the converse suppose that the manipulator votes (in the initial round and the runoff) such that p is a winner of the overall election. We will show that this is only possible if $\mathscr{S}$ has an exact cover.

Clearly, each candidate in $L \cup \{\hat{\ell}\}$ will not participate in the runoff since the manipulator can only possibly add one additional veto to each candidate. So there is no way for any of the candidates in $B \cup R \cup \{d\}$ to have more than three vetoes in the initial round. Let C' be the set of candidates that participate in the runoff. Note that $C' = \{p, d\} \cup B \cup \widehat{S} \cup R$, where $\widehat{S} \subseteq S$.

If $||C'|| < 2n + 4k + 2$, then each voter vetoes at least one and at most $n + 2k = ||R||$ candidates in the runoff. This causes d to receive two vetoes while p receives four vetoes from the nonmanipulators in the runoff. Since there is only one manipulator, p will not win the runoff.

If $||C'|| \geq 2n + 4k + 4$, then each voter vetoes at least $||R|| + 2$ and certainly no more than $||R|| + ||B||$ candidates in the runoff. Note that $\widehat{S} \neq \emptyset$ and that every candidate in $\widehat{S}$ is vetoed four times by the nonmanipulators in the runoff while p clearly has more than five vetoes. Since there is only one manipulator, p will not win the runoff.

It remains to handle the case where $2n+4k+2 \leq ||C'|| \leq 2n+4k+3$. In this case, each voter vetoes $||R||+1$ candidates in the runoff. p has four vetoes from the nonmanipulators in the runoff. So, every $b \in B$ needs at least three vetoes from the nonmanipulators in the runoff. The only way b can get more than two vetoes from the nonmanipulators is if there exists an i such that (a) $b \in S_i$ and (b) s_i does not participate in the runoff. It follows that $S - \widehat{S}$ corresponds to a cover for B. So, since $||S - \widehat{S}|| \leq k$ and any cover must have size at least k, it follows that $S - \widehat{S}$ corresponds to an exact cover for B. □

4.3.2 Scoring protocols

It is easy to argue, in contrast with the result of Theorem 18 regarding Plurality's close cousin Veto, that Plurality is easy, even in the weighted case, since throwing all one's votes to p is always optimal.

It is important to make the distinction between our result that Plurality-CWCM-runoff and Plurality-CWCM-revoting are each in P and Narodytska and Walsh's claim that computing a weighted coalition manipulation for Plurality with Runoff is NP-hard with or without revoting [38]. For their results they use the definition TopTwo THEN Plurality, instead of the arguably more natural approach of using the same system at each stage, which is what we are considering here. (We will define TopTwo later, namely on page 20.)

Theorem 21 Plurality-CWCM-runoff *and* Plurality-CWCM-revoting *are each in* P.

We mention the following result, which holds because by brute-force partitioning of the integer $\|V_2\|$ (the number of manipulative voters) into $(\|C\|!)^2$ named buckets (one for each pair of possible votes, though a second-round decrease in candidates could make the numbers even smaller than this), one can solve in polynomial time even the revoting runoff manipulation question (and of course polynomial time similarly holds for plain runoffs and, as is well known, for CUCM). This holds since the number of ways to partition $\|V_2\|$ identical items into $(\|C\|!)^2$ named buckets is $\binom{\|V_2\|-1}{(\|C\|!)^2-1}$. One should keep in mind that the claims being made regard systems in which the voters are indistinguishable and the integer $\|V_2\|$ is given in unary.

Theorem 22 *For any election system X having a* P *winner problem, and for any integer k,* X-CUCM-revoting *restricted to k candidates is in* P.

There is a dichotomy result for the complexity of CWCM for scoring protocols [28]. The following claim transfers that result to CWCM-runoff and CWCM-revoting.

Theorem 23 *For every scoring protocol X,* X-CWCM-runoff *and* X-CWCM-revoting *are in* P *if X is Plurality or Triviality (or a direct transform of one of those, in a sense that can be made formal, see* [28]*), and otherwise are* NP-*complete.*

Proof For Plurality, this follows from Theorem 21 and for Triviality, this is trivial. For every other weighted scoring protocol X, Hemaspaandra and Hemaspaandra [28] give a reduction f from the NP-complete problem Partition to X-CWCM with the property that for all x, if $x \in$ Partition, then p can be made the unique winner in $f(x)$, and if $x \notin$ Partition, then

p can not be made a winner in $f(x)$. So, if $x \in$ Partition, then p can be made the unique winner of the initial round, and thus the unique winner of the overall election. And if $x \notin$ Partition, then p will never make it to the final round. $\square$

4.3.3 Weighted specific voting rules where the complexity increases

Theorems 18 and 20 give examples of natural unweighted systems where runoffs increase the complexity of manipulation. What about the weighted case? This is harder, since there are far fewer examples of natural weighted election systems for which manipulation is easy. And those examples tend to be so easy that they remain easy with runoffs. For example, weighted manipulation for scoring protocols is easy if and only if the system is in effect Plurality or Triviality, and those remain easy (see Theorem 21). Narodytska and Walsh [38, Proposition 7] show that manipulation of Condorcet THEN Plurality with weighted votes is NP-complete with three or more candidates. Their definition of Condorcet selects as winners the Condorcet winner (i.e., the candidate that beats every other candidate in a pairwise election) and all candidates if there is no Condorcet winner. However, it is immediate from their proof that Condorcet-CWCM (in the definition of [38]) with at least three candidates is already NP-complete, and so the second round doesn't increase the complexity.

Narodytska and Walsh [38] also show that manipulation of TopTwo THEN Plurality with weighted votes is NP-complete, even when restricted to three candidates; this result holds with or without revoting. TopTwo selects the two candidates with the highest plurality scores (ties are broken lexicographically, and the candidates are renamed so that the preferred candidate is lexicographically first) [38]. Note that, since—as mentioned in the proof of Theorem 2—election systems are blind to who the preferred candidate is, TopTwo is technically not an election system. Now consider the election system TopTwo_m, where the winner is the Majority winner (if it exists) and otherwise the two candidates with the highest Plurality scores (ties are broken in favor of the lexicographically least candidate). TopTwo_m-CWCM is clearly in P; the optimal strategy for the manipulators is to vote for p. It is easy to see that, for elections in which the preferred candidate is lexicographically first, TopTwo_m-CWCM-runoff is the same problem as weighted manipulation of TopTwo THEN Plurality (a system that in the literature is often referred to as Plurality with Runoff) and that TopTwo_m-CWCM-revoting is the same problem as weighted manipulation of TopTwo THEN Plurality with revoting. Thus the following holds.

Theorem 24 TopTwo_m-CWCM *is in* P*, while* TopTwo_m-CWCM-runoff *and* TopTwo_m-CWCM-revoting *are each* NP*-complete, even when restricted to three candidates.*

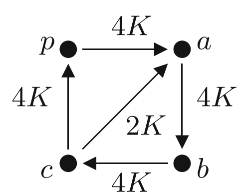
Is there an example of an election system *that is already in the literature* for which runoffs increase the complexity of weighted manipulation? There is one known case where weighted manipulation is easy, but in a very nontrivial way. This is Llull-CWCM, restricted to four candidates [23]. This is in contrast to Copeland^α for other values of α (recall that Llull is Copeland^1), which are hard for three or more candidates [21, 22]. The reason that we don't get hardness in the same way in the case for Llull, is that we get hardness by enforcing ties (which allows us to encode Partition), but in the case of Llull, a tie is never better than a non-tie for making p a winner. However, if we have a second round, it does not just matter if p wins, but also if other candidates win, since these candidates participate in the second round. It turns out that to make candidates other than p win, we sometimes need to enforce a tie, which allows us to encode Partition, and we get NP-completeness.

This already happens for four candidates, and so we have, in four-candidate weighted Llull, found an example of a case where runoffs make the problem go from easy to hard.[9]

Theorem 25 Llull-CWCM-runoff *and* Llull-CWCM-revoting *are each* NP-*complete, even when restricted to four candidates.*

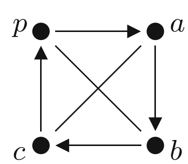
Proof When looking at Llull elections, it is often convenient to think of an election as its induced weighted majority graph. Given an election $E = (C, V)$, E's induced weighted majority graph is the directed graph that has C as its vertices and for each pair of vertices c, d, there is an edge from c to d with weight $w > 0$ if c beats d by a margin of w in the pairwise election between c and d. If we leave off the weights, we get the induced majority graph. Note that we can determine the winners of a Llull election from its induced majority graph.

As in Section 4.1.2, we will reduce from the well-known NP-complete problem Partition: Given a nonempty set of positive integers $k_1, \ldots, k_t$ that sums to $2K$, we ask if there exists a subset that sums to K. Let $C = \{p, a, b, c\}$. Construct a set of nonmanipulators V_1 such that the induced weighted majority graph of (C, V_1) looks like this:



Such a V_1 exists (since all weights have the same parity) and can be computed in polynomial time using McGarvey's construction [35]. The manipulators have weights $k_1, \ldots, k_t$. Since the total weight of the manipulators is $2K$, no matter how the manipulators vote, the induced majority graph contains the cycle $p \to a \to b \to c \to p$ and does not contain the edge $a \to c$.

If $k_1, \ldots, k_t$ can be partitioned into two sets of equal weight (i.e., weight K), the manipulators vote so that K of the manipulator weight votes $p > b > a > c$ and K of the manipulator weight votes $b > p > a > c$. This gives the following induced majority graph (drawing an undirected edge for a tie):



Note that all candidates have a score of 2. So, all candidates proceed to the second round. Without revoting, all candidates win the second round. So, even without revoting p is a winner of the election.

[9]Runoffs do not increase the complexity for three-candidate weighted Llull, since in that case p can be made a winner of the initial round if and only if p can be made a winner of the election with runoff. If this were not true, there would be a candidate a such that a defeats p in their pairwise election and the winners of the initial round are p and a. But then the score of p in the initial round is 1, and the only way p can be a winner of the initial round is if the third candidate, b, defeats a in their pairwise election. But then b is also a winner of the initial round.

For the converse, suppose that p is a winner of the election. c always makes it to the second round. If the set of candidates in the second round is $\{p, c\}$ or $\{p, a, c\}$, c is the unique winner. So, in order for p to be a winner, both p and b need to make it to the second round. This implies that p and b tie in the first round. Consider the set of weights of the manipulators for which $p > b$ in the first round. This set sums to K and so we have a partition. □

Note that the construction above also gives a natural election system for which it is not always better for the manipulators to put p first in the first round (although of course doing so would be optimal in classic manipulation).

We realize that upon seeing results such as Theorems 18–21, 23, and 20–25 it might be natural to wonder whether NP-hardness of X-CWCM-runoff automatically implies NP-hardness of X-CWCM-revoting; Theorem 1 however shows that no such universal implication holds.

5 Conclusions and open problems

This paper has explored the relative manipulation complexity of runoff elections, with and without revoting. We have seen that there is no general relation between the manipulation complexity of either of those with each other or with the manipulation complexity of the underlying election system. Sometimes revoting can even lower complexity, for example. Yet for the natural, concrete systems we studied, runoffs and revoting runoffs never lowered complexity and sometimes raised complexity.

Important open directions include the study of runoffs and revoting runoffs for the case of bribery rather than manipulation, for which we have some preliminary results; seeking to find a natural system for which the complexity of X-CWCM-runoff and X-CWCM-revoting differ (even if one is allowed to use different first- and second-round systems, this question is open in the literature); and the study of what role heuristics, especially in light of Theorems 19 and 20, can play.

Acknowledgments This work was supported in part by NSF grants CCF-0915792, CCF-1101452, and CCF-1101479, and by NSF graduate research fellowship DGE-1102937. The observation that STV′ and TopTwo are technically not election systems is due to an anonymous referee. We are very grateful to the anonymous referees for helpful comments and suggestions that we feel much improved this paper.

Appendix A: Deferred proofs

This appendix provides the proofs that the body of the paper deferred.

A.1 Deferred proofs from Section 4.1.1

Proof of Theorem 6 Let $f(\cdot)$ be as specified in Definition 1. Let $X = AlwaysWinners(Y)$, where Y is defined as follows:

If $\|V\| \neq 1$ or $\|C\| = 1$ then everyone loses. Otherwise, we'll handle things based on the number of candidates and the single voter v, as described below.

If $\|C\| = 2$ and the lexicographically larger candidate is more preferred than the lexicographically smaller candidate by the voter then everyone wins, else everyone loses.

If $\|C\| \geq 3$, the lexicographically smallest candidate's name encodes a formula ψ, $\|C\| \geq f(2^{\#\text{vars}(\psi)}) + 1$ (the "+1" is as we'll not involve the lexicographically smallest candidate in encoding the assignment), v's vote on all candidates other than the lexicographically smallest candidate codes an assignment that satisfies ψ, and the lexicographically smallest candidate is more preferred than the lexicographically largest candidate by v, then the lexicographically largest and smallest candidates win, else everyone loses.
That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly the winner problem for Y is in P.
- Y-CUCM is NP-complete by reducing from SAT. Observe that $\psi \in$ SAT reduces to Y-CUCM for the candidate set: p encoding ψ and $\max(2, f(2^{\#\text{vars}(\psi)}))$ dummy candidates all lexicographically larger than p. Let the preferred candidate be p and let there be zero nonmanipulative voters and one manipulative voter. The manipulator must vote with p at the top of her preference order and code a satisfying assignment to ψ over the dummy candidates. If $\psi \notin$ SAT then no one wins. Therefore Y-CUCM is NP-complete.
- Y-CUCM-runoff is in P since in the initial round we always have zero or two winners. However, when we have two winners (and more than two candidates), they always lose in the second round unless the manipulator can change her vote. So in the second round scheme (runoff without revoting), no one ever can win if we have more than two candidates.
- Y-CUCM-revoting is NP-complete by the same argument that we used to show that Y-CUCM is NP-complete. Here, things works as follows. When the ψ encoded by the lexicographically smallest candidate is satisfiable, the manipulator in the initial round conveys (in the dummy candidates) a satisfying assignment, and she puts the lexicographically smallest candidate as her most preferred candidate. In the runoff the manipulator puts the lexicographically smallest candidate as her least preferred candidate. (So we'll go to the $\|C\| = 2$ case of Y in the runoff and we will have two winners, including the lexicographically smallest candidate.)

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

Proof of Theorem 7 Let $X = \textit{AlwaysWinners}(Y)$, where Y is defined as follows:

If $\|C\| \geq 2$ then STV$''$ (where STV$''$ is as defined in Theorem 2). Otherwise, everyone loses.

That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly Y has a polynomial-time winner problem.
- Y-CUCM is NP-complete since Y-CUCM on at least two candidates corresponds to STV$''$-CUCM.
- Y-CUCM-runoff and Y-CUCM-revoting are each in P since the initial round results in at most one winner, and so the runoff will never give a winner (even with revoting).

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

Proof of Theorem 8 Let $X = \mathit{AlwaysWinners}(Y)$, where Y is defined as follows.

We will utilize several different special candidate names in our proof. The candidate names are as follows:

- $\langle \text{Shiva}_1, \varepsilon \rangle$, which we refer to as Shiva_1.
- $\langle \text{Shiva}_2, \psi \rangle$, where ψ is a boolean formula, which we refer to as Shiva_2-like.
- $\langle \text{Angel}, 1 \rangle$, which we refer to as Angel_1.
- $\langle \text{Angel}, 2 \rangle$, which we refer to as Angel_2.
- More angels are used as needed.

Let $f(\cdot)$ be as specified in Definition 1. For our election Y, the candidate set is expected to be one of the following forms:

(a) One Shiva_2-like candidate with formula ψ, the Shiva_1 candidate, and enough angel candidates to encode $2^{\#\text{vars}(\psi)} + 1$ possibilities ($2^{\#\text{vars}(\psi)}$ assignments to ψ and one special "Begone-2" ordering), i.e., $f(2^{\#\text{vars}(\psi)} + 1)$ angel candidates.
(b) One Shiva_2-like candidate with formula ψ and enough angel candidates to encode $2^{\#\text{vars}(\psi)} + 1$ possibilities, i.e., $f(2^{\#\text{vars}(\psi)} + 1)$ angel candidates.

Let Y be defined as follows:

If $\|V\| \neq 1$ or C is not of an expected form then everyone loses.

If C is of form (a) and the angel candidates' restriction of the voter's vote encodes an assignment to ψ, then the Shiva_2-like candidate and all of the angel candidates win, else everyone loses.

If C is of form (b) and the angel candidates' restriction of the voter's vote encodes the special "Begone-2" ordering or a satisfying assignment to ψ then all of the angels win, else everyone loses.

That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly the winner problem for Y is in P.
- Y-CUCM is in P by looking at both of the allowed candidate set forms:
    - For form (a), when the voter casts a vote that encodes an assignment to ψ over the angels then everyone wins who can possibly win.
    - For form (b), when the voter casts a vote that encodes the special "Begone-2" ordering over the angels then everyone wins who can possibly win.
- Y-CUCM-runoff is NP-complete by reducing from SAT. Observe that to see if $\psi \in$ SAT ask if Angel_1 can win the Y-CUCM-runoff instance with the candidate set: Shiva_1, $\langle \text{Shiva}_2, \psi \rangle$, and $f(2^{\#\text{vars}(\psi)} + 1)$ angels. Let the voter set contain zero nonmanipulative voters and one manipulative voter.

  If $\psi \in$ SAT, then the manipulator casts the initial-round vote that corresponds to a satisfying assignment to ψ, so the Shiva_2-like candidate and all of the angels win in the initial round and all of the angels win in the runoff. Conversely, if $\psi \notin$ SAT then Angel_1 would not be able to win in the runoff.
- Y-CUCM-revoting is in P by looking at both of the allowed candidate set forms:
    - For form (a), when the voter casts a vote that codes an assignment to ψ over the angels in the initial round and then changes her vote to encode the special

Table 1 Allowed candidate types for the election Y used in the proof of Theorem 9

Candidate form	Role in our proof
$\langle 1, 1\rangle$	Seeks to make the initial round hard.
$\langle 1, 2\rangle$	Seeks to make the initial round easy.
$\langle 2, \psi\rangle$	Candidate coding a formula intended as part of a hard initial round.
$\langle 3, \psi\rangle$	Candidate coding a formula intended as part of a hard runoff.
$\langle 4, \text{(any string)}\rangle$	"Type-4" dummy candidate, used to make votes so big as to encode assignments.
$\langle 5, 1\rangle$, $\langle 5, 2\rangle$	Special dummy candidates to allow vote changes to show through in some cases.

"Begone-2" ordering over the angels then everyone wins who can possibly win.

– For form (b), everyone loses by the end of the runoff.

So Y satisfies the theorem in the model where there is always a runoff, and $AlwaysWinners(Y)$ satisfies the theorem in our standard model. □

Proof of Theorem 9 Let $f(\cdot)$ be as specified in Definition 1. Let $X = AlwaysWinners(Y)$, where Y is defined as follows. The allowed (all others will cause everyone to lose) candidate types for Y are described in Table 1.

For our election Y, the candidate set is expected to be in one of the following forms (we assume w.l.o.g. that every formula has at least one variable):

(a) $\langle 1, 1\rangle$, $\langle 2, \psi\rangle$, and enough type-4 dummy candidates so that a vote including them can encode an assignment to ψ, i.e., at least $f(2^{\#\mathrm{vars}(\psi)})$ dummy candidates.

(b) $\langle 1, 2\rangle$, $\langle 3, \psi\rangle$, $\langle 5, 1\rangle$, $\langle 5, 2\rangle$, and enough type-4 dummy candidates so that a vote including them can encode an assignment to ψ, i.e., at least $f(2^{\#\mathrm{vars}(\psi)})$ dummy candidates.

(c) $\langle 3, \psi\rangle$, $\langle 5, 1\rangle$, $\langle 5, 2\rangle$, and enough type-4 dummy candidates so that a vote including them can encode an assignment to ψ, i.e., at least $f(2^{\#\mathrm{vars}(\psi)})$ dummy candidates.

Let Y be defined as follows:

If $\|V\| \neq 1$ or C is not of an expected form then everyone loses. Otherwise, we'll handle things as described below (we will refer to the one voter in this case as v below).

If C is of form (a) and the type-4 dummy candidates' restriction of v's vote encodes a satisfying assignment to ψ, then $\langle 2, \psi\rangle$ wins. Otherwise, everyone loses.

If C is of form (b) and $\langle 5, 2\rangle > \langle 5, 1\rangle$ in v's vote then $\langle 3, \psi\rangle$, $\langle 5, 1\rangle$, $\langle 5, 2\rangle$,and all of the type-4 dummy candidates win. In all other cases where C is of form (b), everyone loses.

If C is of form (c) and $\langle 5, 1\rangle > \langle 5, 2\rangle$ in v's vote or the type-4 dummy candidates' restriction of v's vote encodes a satisfying assignment to ψ, then $\langle 3, \psi\rangle$ wins. Otherwise, everyone loses.

This completes the specification of Y.

Now we must ensure that Y meets all of the requirements of this theorem in the model where there is always a runoff.

– Clearly the winner problem for Y is in P.
– Y-CUCM is NP-complete since to test if $\psi \in$ SAT, we can ask if $\langle 2, \psi\rangle$ can win the Y-CUCM instance with candidate set $\{\langle 1, 1\rangle, \langle 2, \psi\rangle, f(2^{\#\mathrm{vars}(\psi)})$ type-4 dummy candidates$\}$ and the voter set containing zero nonmanipulative voters and one manipulative

voter. For $\langle 2, \psi \rangle$ to win the manipulator must code a satisfying assignment to ψ in her ordering over the type-4 dummy candidates. If $\psi \notin$ SAT, then no one can win.

- Y-CUCM-runoff is NP-complete since to test if $\psi \in$ SAT, we can ask if $\langle 3, \psi \rangle$ can win the Y-CUCM-runoff instance with candidate set $\{\langle 1, 2\rangle, \langle 3, \psi\rangle, \langle 5, 1\rangle, \langle 5, 2\rangle, f(2^{\#\text{vars}(\psi)})$ type-4 dummy candidates$\}$ and voter set containing zero nonmanipulative voters and one manipulative voter.
  Note that we need $\langle 5, 2\rangle > \langle 5, 1\rangle$ in the manipulator's vote in order for $\langle 3, \psi\rangle$ to make it to the runoff. If $\psi \notin$ SAT, then $\langle 3, \psi\rangle$ cannot win. Otherwise, if $\psi \in$ SAT, voting so that $\langle 5, 2\rangle > \langle 5, 1\rangle$ in the manipulator's vote and with the type-4 dummy order giving a satisfying assignment to ψ, makes $\langle 3, \psi\rangle$ a winner.
- Y-CUCM-revoting is in P by looking at each of the allowed candidate set forms:
  - For form (a), everyone loses by the end of the runoff.
  - For form (b), when voter v casts a vote where $\langle 5, 2\rangle > \langle 5, 1\rangle$ in the initial round and a vote where $\langle 5, 1\rangle > \langle 5, 2\rangle$ in the runoff, then everyone wins who can possibly win.
  - For form (c), everyone loses by the end of the runoff.

  All other cases have everyone lose immediately. Thus Y-CUCM-revoting is in P.

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

A.2 Deferred proofs from Section 4.1.2

Proof of Theorem 14 Let $X = \textit{AlwaysWinners}(Y)$, where Y is defined as follows:
If $\|C\| = 3$ then candidates with plurality scores of exactly 50% win.
If $\|C\| = 2$ then candidates with plurality scores $> 50\%$ win.
Otherwise, everyone loses.
That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly the winner problem for Y is in P.
- Y-CWCM is NP-complete since to test if a set of positive integers $k_1, \ldots, k_t \in$ Partition we can ask if the candidate p can win the Y-CWCM instance with the candidate set $\{p, r, \ell\}$ and the voter set containing zero nonmanipulative voters and t manipulative voters. Let the t manipulators have weights that correspond to the Partition instance, i.e., $k_1, \ldots, k_t$ such that $\sum_{i=1}^{t} k_i = 2K$.
  Since $\|C\| = 3$, p is a winner if and only if p has a score of exactly 50% of the available vote weight. So, p can be made a winner if and only of the manipulators can partition their votes into two equal-weight subsets.
- Y-CWCM-runoff is in P since no one can win in the runoff without revoting. If the initial round has three candidates where two attain exactly 50% of the vote to move on to the runoff, no one will win in the runoff since neither will have greater than 50% of the available vote weight. If the initial round has only two candidates, the winner will lose the runoff since when there is only one candidate, that candidate always loses. In all other cases the candidates all lose immediately. Therefore Y-CWCM-runoff is in P.
- Y-CWCM-revoting is NP-complete since to test if a set of positive integers $k_1, \ldots, k_t \in$ Partition we can ask if the candidate p can win the Y-CWCM-revoting instance with the candidate set $\{p, r, \ell\}$ and the voter set containing zero nonmanipulative voters and

t manipulative voters. Let the t manipulators have weights that correspond to the Partition instance, i.e., $k_1, \ldots, k_t$ such that $\sum_{i=1}^{t} k_i = 2K$.
To ensure that p is an overall winner, the manipulators must partition their votes between p and w.l.o.g. r, since the only way to have a runoff where a candidate can win is when two candidates win in the initial round. Then in the runoff the manipulators all vote for p to increase p's score to be greater than 50%. Thus if there is a way to partition the vote weight of the manipulators into two equal-weight subsets, then p can be made the overall winner. Otherwise, p cannot be made the overall winner. Therefore Y-CWCM-revoting is NP-complete.

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

Proof of Theorem 15 Let $X = AlwaysWinners(Y)$, where Y is defined as follows:

If $\|C\| \geq 2$ then the candidate with the highest veto score wins as long as the winner is unique. Otherwise (if there is no unique winner or $\|C\| = 1$), everyone loses.
That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly Y has a polynomial-time winner problem.
- Y-CWCM is NP-complete since Y-CWCM restricted to three candidates corresponds to Veto-CWCM restricted to three candidates in the unique winner model, which is known to be NP-complete [13].
- Y-CWCM-runoff and Y-CWCM-revoting are each in P since the initial round results in at most one winner, and so the runoff will never give a winner (even with revoting).

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

Proof of Theorem 16 Let $X = AlwaysWinners(Y)$, where Y is defined as follows:

If $\|C\| = 3$ and there is a candidate with plurality score 0 then candidates with plurality scores greater than or equal to half of the highest occurring score win.

If $\|C\| = 2$ then candidates with plurality scores less than or equal to half of the highest occurring plurality score win.

If $\|C\| = 1$ then that candidate wins.

Otherwise, everyone loses.
That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

- Clearly Y has a polynomial-time winner problem.
- Y-CWCM is in P since the optimal action for the manipulators is to vote for their preferred candidate p (when $\|C\| = 3$ or $\|C\| = 1$) to maximize p's score or to not vote for p (when $\|C\| = 2$) to minimize p's score.
- Y-CWCM-runoff is NP-complete since to test if a set of positive integers $k_1, \ldots, k_t \in$ Partition we can ask if the candidate p can win the Y-CWCM-runoff instance with the candidate set $\{p, r, \ell\}$ and the voter set containing one nonmanipulative voter and t manipulative voters. Let the nonmanipulator have weight K and vote $r > p > \ell$ and

let the t manipulative voters have weights that correspond to the Partition instance, i.e., $k_1, \ldots, k_t$ such that $\sum_{i=1}^{t} k_i = 2K$.
To ensure that the preferred candidate p wins, ℓ needs to score 0 and p needs to score at least K to advance to the runoff. Then p and r will proceed to the runoff. In order for p to win the runoff, p needs to score at most K. So, p needs to score exactly K in order to become a winner.

– Y-CWCM-revoting is in P by the following argument. If $V = \emptyset$, simply check if p is a winner of the two-stage election. Now assume that $V \neq \emptyset$. If $\|C\| = 1$, p will always be a winner. If $\|C\| > 3$, there are no winners. If $\|C\| = 2$, the runoff will consist of at most one candidate. So, the optimal action is for all manipulators to not vote for p in the initial round, to maximize p's chances of participating in the runoff. Finally, let $\|C\| = 3$. If p can be made the unique winner of the initial round, then p wins the overall election. If that is not possible, and p makes it to the runoff, there are two participants in the runoff. In both cases, the optimal action for the manipulators in the initial round is to vote for p. And if the runoff has two participants, the optimal action in the runoff is to not vote for p.

So Y satisfies the theorem in the model where there is always a runoff, and *AlwaysWinners*(Y) satisfies the theorem in our standard model. □

The following proof adds an additional case to the election system used in the proof of Theorem 16, to raise the complexity of manipulation to be NP-complete while keeping the complexity of manipulation with revoting runoffs in P.

Proof of Theorem 17 Let $X = \mathit{AlwaysWinners}(Y)$, where Y is defined as follows:

If $\|C\| = 5$ and there are exactly four candidates that have plurality scores of exactly 25%, then these four candidates win.

If $\|C\| = 3$ and there is a candidate with plurality score 0 then candidates with plurality scores greater than or equal to half of the highest occurring score win.

If $\|C\| = 2$ then candidates with plurality scores less than or equal to half of the highest occurring plurality score win.

If $\|C\| = 1$ then that candidate wins.

Otherwise, everyone loses.

That completes the specification of Y.

Now we explain why this system Y meets all the requirements of the theorem in the model where there is always a runoff.

– Clearly Y has a polynomial-time winner problem.
– Y-CWCM is NP-complete since to test if a set of positive integers $k_1, \ldots, k_t \in$ Partition we can ask if the candidate p can win the Y-CWCM instance with the candidate set $C = \{a, b, c, p, \ell\}$, one weight-K nonmanipulator voting for a, one weight-K nonmanipulator voting for b, and t manipulative voters. Let the t manipulative voters have weights that correspond to the Partition instance, i.e., $k_1, \ldots, k_t$ such that $\sum k_i = 2K$. It is immediate that p is a winner if and only if exactly half of the manipulator weight votes for p. This is possible if and only if $k_1, \ldots, k_t \in$ Partition.
– Y-CWCM-runoff is NP-complete by exactly the same argument as in the proof of Theorem 16.
– Y-CWCM-revoting is in P by the same argument as used above for the previous case (Theorem 16). Observe that the addition of the $\|C\| = 5$ case does not increase the complexity of Y-CWCM-revoting: There are never five candidates in the runoff and if

there are five candidates in the initial round, there will be zero or four candidates in the runoff, and so there will be no winners in the runoff

So Y satisfies the theorem in the model where there is always a runoff, and $AlwaysWinners(Y)$ satisfies the theorem in our standard model. □

References

1. Aziz, H., Gaspers, S., Mattei, N., Narodytska, N., Walsh, T.: Ties matter: Complexity of manipulation when tie-breaking with a random vote. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence, pp. 74–80. AAAI Press (2013)
2. Bag, P., Sabourian, H., Winter, E.: Multi-stage voting, sequential elimination and Condorcet consistency. J. Econ. Theory **144**(3), 1278–1299 (2009)
3. Bartholdi III, J., Orlin, J.: Single transferable vote resists strategic voting. Soc. Choice Welf. **8**(4), 341–354 (1991)
4. Bartholdi III, J., Tovey, C., Trick, M.: The computational difficulty of manipulating an election. Soc. Choice Welf. **6**(3), 227–241 (1989)
5. Bartholdi III, J., Tovey, C., Trick, M.: Voting schemes for which it can be difficult to tell who won the election. Soc. Choice Welf. **6**(2), 157–165 (1989)
6. Brandt, F., Conitzer, V., Endriss, U.: Computational social choice. In: Weiss, G. (ed.) Multiagent Systems. 2nd edn. MIT Press, Cambridge (2013)
7. Buhrman, H., Hitchcock, J.: NP-hard sets are exponentially dense unless coNP $\subseteq$ NP/poly. In: Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, pp. 1–7. IEEE Computer Society Press, Los Alamitos (2008)
8. Cai, J., Chakaravarthy, V., Hemaspaandra, L., Ogihara, M.: Competing provers yield improved Karp–Lipton collapse results. Inf. Comput. **198**(1), 1–23 (2005)
9. California Elections Code, Division 15/Chapter 10/Statute 15650-15673: Tie Votes. http://leginfo.legislature.ca.gov/ (2013)
10. Chevaleyre, Y., Endriss, U., Lang, J., Maudet, N.: A short introduction to computational social choice. In: Proceedings of the 33rd International Conference on Current Trends in Theory and Practice of Computer Science, Springer-Verlag Lecture Notes in Computer Science #4362, pp. 51–69 (2007)
11. Conitzer, V., Sandholm, T.: Universal voting protocol tweaks to make manipulation hard. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 781–788. Morgan Kaufmann, San Mateo (2003)
12. Conitzer, V., Sandholm, T.: Nonexistence of voting rules that are usually hard to manipulate. In: Proceedings of the 21st National Conference on Artificial Intelligence, pp. 627–634. AAAI Press (2006)
13. Conitzer, V., Sandholm, T., Lang, J.: When are elections with few candidates hard to manipulate? J. ACM **54**(3) (2007). Article 14
14. Conitzer, V., Walsh, T.: Barriers to manipulation in voting. In: Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A. (eds.) Handbook of Computational Social Choice. Cambridge University Press. To appear
15. Copeland, A.: A "reasonable" social welfare function. Mimeographed notes from Seminar on Applications of Mathematics to the Social Sciences, University of Michigan (1951)
16. Davies, J., Narodytska, N., Walsh, T.: Eliminating the weakest link: Making manipulation intractable? In: Proceedings of AAAI-12, pp. 1333–1339. AAAI Press (2012)
17. Elkind, E., Erdélyi, G.: Manipulation under voting rule uncertainty. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 624–634 (2012)
18. Elkind, E., Lipmaa, H.: Hybrid voting protocols and hardness of manipulation. In: Proceedings of the 16th Annual International Symposium on Algorithms and Computation, Springer-Verlag Lecture Notes in Computer Science #3872, pp. 206–215 (2005)
19. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L.: Using complexity to protect elections. Commun. ACM **53**(11), 74–82 (2010)
20. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: A richer understanding of the complexity of election systems. In: Fundamental Problems in Computing, pp. 375–406. Springer, Berlin (2009)

21. Faliszewski, P., Hemaspaandra, E., Schnoor, H.: Copeland voting: Ties matter. Tech. Rep. TR-926, Department of Computer Science, University of Rochester, Rochester, NY (2007)
22. Faliszewski, P., Hemaspaandra, E., Schnoor, H.: Copeland voting: Ties matter. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 983–990 (2008)
23. Faliszewski, P., Hemaspaandra, E., Schnoor, H.: Weighted manipulation for four-candidate Llull is easy. In: Proceedings of the 20th European Conference on Artificial Intelligence, pp. 318–323. IOS Press, Amsterdam (2012)
24. Faliszewski, P., Procaccia, A.: AI's war on manipulation: Are we winning? AI Mag. **31**(4), 53–64 (2010)
25. Fitzsimmons, Z., Hemaspaandra, E., Hemaspaandra, L.: X THEN X: Manipulation of same-system runoff elections. Tech. Rep., Computing Research Repository (2013). arXiv:1301.6118v1 [cs.GT]
26. Guo, J., Shrestha, Y.: Controlling two-stage voting rules. In: Proceedings of the 21st European Conference on Artificial Intelligence, pp. 411–416 (2014)
27. Hägele, G., Pukelsheim, F.: The electoral writings of Ramon Llull. Studia Lulliana **41**(97), 3–38 (2001)
28. Hemaspaandra, E., Hemaspaandra, L.: Dichotomy for voting systems. J. Comput. Syst. Sci. **73**(1), 73–83 (2007)
29. Hemaspaandra, E., Hemaspaandra, L., Menton, C.: Search versus decision for election manipulation problems. In: Proceedings of the 30th Annual Symposium on Theoretical Aspects of Computer Science, Leibniz International Proceedings in Informatics (LIPIcs), pp. 377–388 (2013)
30. Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. J. ACM **44**(6), 806–825 (1997)
31. Hemaspaandra, E., Hemaspaandra, L., Rothe, J.: Anyone but him: The complexity of precluding an alternative. Artif. Intell. **171**(5–6), 255–285 (2007)
32. Hemaspaandra, E., Spakowski, H., Vogel, J.: The complexity of Kemeny elections. Theor. Comput. Sci. **349**(3), 382–391 (2005)
33. Hemaspaandra, L., Williams, R.: An atypical survey of typical-case heuristic algorithms. SIGACT News **43**(4), 71–89 (2012)
34. Lev, O., Rosenschein, J.: Convergence of iterative voting. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 611–618 (2012)
35. McGarvey, D.: A theorem on the construction of voting paradoxes. Econometrica **21**(4), 608–610 (1953)
36. Meir, R., Polukarov, M., Rosenschein, J., Jennings, N.: Convergence to equilibria in plurality voting. In: Proceedings of AAAI-10, pp. 823–828. AAAI Press (2010)
37. Narodytska, N., Walsh, T.: Manipulating two stage voting rules. In: Proceedings (Workshop Notes) of the 4th International Workshop on Computational Social Choice, pp. 323–334 (2012)
38. Narodytska, N., Walsh, T.: Manipulating two stage voting rules. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 423–430 (2013)
39. North Carolina General Statues, Chapter 163/Article 15A/Statute 163-182.8: Elections and Election Laws. http://www.ncleg.net/ (2013)
40. Obraztsova, S., Elkind, E.: On the complexity of voting manipulation under randomized tie-breaking. In: Proceedings of IJCAI-11, pp. 319–324. AAAI Press (2011)
41. Obraztsova, S., Elkind, E., Hazon, N.: Ties matter: Complexity of voting manipulation revisited. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 71–78 (2011)
42. Reijngoud, A., Endriss, E.: Voter response to iterated poll information. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 635–644 (2012)
43. Reyhani, R., Wilson, M.: Best reply dynamics for scoring rules. In: Proceedings of the 20th European Conference on Artificial Intelligence, pp. 672–677. IOS Press, Amsterdam (2012)
44. Rothe, J., Schend, L.: Challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey. Ann. Math. Artif. Intell. **68**(1–3), 161–193 (2013)
45. Rothe, J., Spakowski, H., Vogel, J.: Exact complexity of the winner problem for Young elections. Theory Comput. Syst. **36**(4), 375–386 (2003)
46. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge (2009)
47. Walsh, T.: Where are the hard manipulation problems? J. Artif. Intell. Res. **42**, 1–29 (2011)
48. Xia, L., Conitzer, V., Procaccia, A.: A scheduling approach to coalitional manipulation. In: Proceedings of the 11th ACM Conference on Electronic Commerce, pp. 275–284. ACM Press (2010)
49. Zuckerman, M., Procaccia, A., Rosenschein, J.: Algorithms for the coalitional manipulation problem. Artif. Intell. **173**(2), 392–412 (2009)