Auton Agent Multi-Agent Syst
DOI 10.1007/s10458-016-9349-1

The complexity of online voter control in sequential elections

Edith Hemaspaandra[1] · Lane A. Hemaspaandra[2] · Jörg Rothe[3]

Abstract Previous work on voter control, which refers to situations where a chair seeks to change the outcome of an election by deleting, adding, or partitioning voters, takes for granted that the chair knows all the voters' preferences and that all votes are cast simultaneously. However, elections are often held sequentially and the chair thus knows only the previously cast votes and not the future ones, yet needs to decide instantaneously which control action to take. We introduce a framework that models *online voter control in sequential elections*. We show that the related problems can be much harder than in the standard (non-online) case: For certain election systems, even with efficient winner problems, online control by deleting, adding, or partitioning voters is PSPACE-complete, even if there are only two candidates. In addition, we obtain (by a new characterization of coNP in terms of weight-bounded alternating Turing machines) completeness for coNP in the deleting/adding cases with a bounded deletion/addition limit, and we obtain completeness for NP in the partition cases with an additional restriction. We also show that for plurality, online control by deleting or adding voters is in P, and for partitioning voters is coNP-hard.

Keywords Computational social choice · Voter control of elections · Sequential elections · Online control

A preliminary version of parts of this paper has appeared in the Proceedings of the 20th European Conference on Artificial Intelligence [26].

✉ Jörg Rothe
rothe@cs.uni-duesseldorf.de

1 Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA

2 Department of Computer Science, University of Rochester, Rochester, NY 14627, USA

3 Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany

1 Introduction

Elections are important not just in the human world. They also can function as an important way of aggregating the preferences of (often electronic) agents, in our world that is increasingly networked and in which people and institutions will increasingly be spoken for by automated agents.

In the field of multiagent systems, voting has been suggested for tasks as varied as, for example, recommender systems, collaborative spam filtering, and planning [9,13,21]. And not surprisingly, the study of the computational properties of voting systems has been an exceedingly active area within computational social choice.

In particular, various types of manipulation, electoral control, and bribery in voting have been classified in terms of their computational complexity (see [15,19]). This paper focuses on *voter control*, a model introduced by Bartholdi, Tovey, and Trick [5], where a chair attempts to alter the outcome of an election via changing its structure by deleting, adding, or partition of voters. These types of control seek to model such real-world behaviors as targeted vote suppression, bring-out-the-vote drives, and districting/gerrymandering. Bartholdi, Tovey, and Trick's paper was in the bounded-rationality spirit of Simon [40], and was in part making the point that computational complexity is important in decision-making.

There have been many papers analyzing the (non-online) control complexity of election systems, and seeking to find natural systems that make many types of control attack difficult. See, for example, the surveys [15,19], the book chapters [4,20,29], and the references therein. Control is such a natural model that it has also been applied in settings quite different than standard elections, e.g., it has been applied in judgment aggregation [2] and group identification [45]. To the best of our knowledge, all previous work on control (see, e.g., [5,11,12,18,23,24]) takes for granted that the chair has full knowledge of all the voters' preferences and that all votes are cast simultaneously.[1] However, in many settings voters vote sequentially and the chair's task in such a setting may often be quite different: Knowing only the already cast votes but not the future ones, the chair must decide *online* (i.e., in that moment) whether there exists a control action that guarantees success, no matter what votes will be cast later on. We introduce a framework to model *online voter control in sequential elections*. Our approach is inspired by the area of "online algorithms" [1]—algorithms running and performing computational actions based only on the input data seen thus far.

In our framework of online voter control, the chair's task stated above is based on a "maximin" idea (although here, due to the time effects, that can involve more than two quantifiers; this hints at a "game-like" flavor, and that is further explored in our Preliminaries section, see also footnote 3), a typical online-algorithmic theme; in that framing of the chair's task we are following the approach that has been used for online manipulation and online candidate control [25,28]. Note that another central online-algorithmic theme, a strictly numerical ratio approach to so-called "competitive analysis," would not apply very naturally here; the reason is that in its general setting, voting (in social-choice theory) is most typically based on an ordinal notion of preferences, and those don't convey cardinal strength-of-affinity information regarding the outcome (that is, they don't provide any fixed numerical valuation of different outcomes, e.g., if our preference is $a > b > c > d$ that does not provide a numerical value to the relative desirability to us of c winning versus d winning). (For some

[1] An exception is a paper by Fitzsimmons et al. [16] that is, regarding their earliest appearing versions, more recent than the present paper, and studies a mixed model involving both a chair and manipulators, in which the manipulative voters set their votes after control action by the chair. (We mention in passing that if one looks beyond the study of control, uncertainty appears in many election, selection, and preference-aggregation settings, see, e.g., the book [39] and, as one among many possible examples, the work of Mattei et al. [33].)

specific voting systems such as so-called scoring systems one can interpret them as giving cardinal information, and we commend as an interesting open issue a future, general control-complexity study for such systems in terms of a competitive-ratio analysis; see [35], which takes that approach for the issue of selecting a bundle of goods.) Sequential (or otherwise "dynamic") voting has been studied in other contexts as well, e.g., from a game-theoretic perspective as "Stackelberg voting games" [44] (see also [8,10,41]), or using an axiomatic approach [43] or Markov decision processes [37]. None of this work has considered the issue of voter control.

What our results show is that such online control problems can be much harder than in the standard (non-online) case. We show that for certain election systems, even with efficient winner problems, online control by deleting, adding, or partitioning voters is PSPACE-complete, even if there are only two candidates. In addition, we obtain completeness for coNP in the deleting/adding cases with a bounded deletion/addition limit. We do this by establishing a complexity-theoretic result (Theorem 3) that is of interest in its own right: Polynomial-time alternating Turing machines that on each accepting path make a constant number of "Yes" guesses accept only coNP languages, and in fact this completely characterizes coNP. We also show that for plurality, online control by deleting or adding voters is in P, and for partitioning voters is coNP-hard.

The comments made in the previous paragraph put into context our investigation, and its framing, but let us make this explicit: This paper is primarily trying to discover the limits of complexity that can possibly exist for any election system (that has a polynomial-time winner problem) regarding these problems—even if realizing the limits involves an artificial system. We do so because to understand a problem one needs to understand the complexities that it can take on. For example, note that knowing that a problem is PSPACE-complete for some systems having polynomial-time winner problems makes it clear that we should not be seeking a result of the form: For all systems having polynomial-time winner problems this problem is in NP. But, despite discovering that high complexities can occur, one naturally can then hope to look at specific, popular systems to show that they may take on far lower levels of complexity, as we do here for the most important of all systems, plurality. (We say that plurality is the most important of all election systems as it is in extremely widespread use, both in political elections and across a broad range of other contexts.) And in our conclusion section we commend the study of additional concrete systems as a natural, important future direction. (These comments mostly focus on the issue of the role of election systems in our results; please see also the final two paragraphs of the Preliminaries section for discussion somewhat related to this and, far more so, discussion related to the use of and role of maximin.)

2 Motivation

The coming sections will give our definitions, results, and proofs. However, before that, the present section will very informally present some motivation and examples. In particular, we give example settings in which it is natural to study sequential action, in which the election's "chair" has a use-it-or-lose-it ability to do addition/deletion/partition-choice for each voter as the voter votes, and the chair knows the votes of the voters seen so far, but not of future voters. Of course, theoretical models don't capture the many interactions and subtleties of the real world, and so our models don't perfectly capture the full richness of even these sample situations. Nonetheless, we feel that for many cases, such as those we are about to mention, the theoretical models we develop in this paper are far closer to capturing the real-world situation than are existing models of simultaneous voting or even existing models where votes are sequential but all voters' preferences are known ahead of time.

As a concrete example (and let us for the moment not worry about what the particular election system is), consider a College faculty meeting at which, going right around the room, the faculty members hand their handwritten paper ballots to the Dean, who then passes them on to her administrative assistant, who quietly adds them to the totals he is keeping. But let us further assume that the Dean is a shifty person, and can, for a certain number of ballots, slip the piece of paper into her pocket after reading the vote, without that being noticed, and without the people in the room being likely to notice that there aren't quite enough votes in the totals (let's suppose it is a big college). And the question is, given that we are at some particular point in going around the table (and know what votes have been cast so far and what actions the Dean—or whoever was standing in for her—has taken so far): Can the Dean ensure, using at most her remaining amount of vote-to-pocket shifting, that the winner(s) of the election will include at least one of the alternatives she favors? This setting loosely corresponds to our sequential version of control by deleting voters. For vividness, our examples are about humans voting and a human chair (in the above, the Dean), and in the case just given, paper ballots. However, our model applies also to more electronically focused cases of preference aggregation, e.g., the "Dean" in the above example could be a doctored voting machine that can only suppress so many ballots before seriously risking detection.

The above example is about deleting voters, but there are also natural examples for adding or partitioning voters. For partitioning voters, imagine that a school's undergraduate admissions office is going to use a panel, whose members will each be assigned to one of two faculty committees, to vet applying students (perhaps with the committees purportedly looking for different things, e.g., one is looking for traditional smartness and the other is looking for unusual levels of passion and creativity), with all applicants' folders given to both committees, and with each committee using voting to select its favorite proposals, and then with only the winners of those two vetting elections moving on to a final election in which all the panel members vote. Suppose a particular admissions office staff member (who is the chair in this example), with all the faculty members lined up and coming into the room, as each faculty member steps to the doorway briefly chats with the faculty member well enough to discern the likely votes he or she will cast, and then right there assigns the person to either the smartness committee or the passion committee. If the admissions staff member does so with the goal of ensuring that at least one of a certain set of students (perhaps the students who are great football quarterbacks, or the students whose parents might fund a new admissions building) will be admitted, that very loosely put would be captured by our sequential version of control by partition of voters.[2] For adding voters, a natural model might be a political candidate (who is the chair in this example) going door to door through her district in a preset order, and knowing from public records which voters are registered voters and which are not, and at each door meeting and learning the voter's preferences among the candidates, and then for those voters who are not registered deciding whether to use charisma to convince them to register and vote, with the limitation that the candidate has only so much charisma to use.

The above are a few very informal examples of settings where sequential action is natural, and one knows the votes cast so far but not those to be cast in the future (except who will be casting them and in which order). Let us finish this informal section by briefly giving a mini-example of the flavor of the goal we have for our chairs, and how that affects their actions.

[2] Actually, as our previous example suggested, our model is a bit more flexible and allows one to ask such questions starting at an intermediate point at which some actions have already been taken, potentially by a different admissions staff member.

We are assuming that chairs are very pessimistic: What a chair wants to know is whether there is some action she can take at the given moment so that one of her preferred candidates will win no matter what the value is of all the currently unknown-to-her future votes—but assuming that her own future decisions are (of course) aimed at supporting her goal. To make this more concrete, let us discuss the most important real-world election system: plurality. In our addition-of-voters example above, suppose the candidate going door to door has only one preferred candidate in the election, namely, herself. Then it is quite clear and simple what she should do. Until she runs out of charisma, she should for each unregistered voter she meets for whom she is the favorite candidate expend her charisma to have that person become a registered voter. That is an "operational" approach that would work perfectly. But more must be said. The question our pessimistic candidate (and our decision problems) wants answered at each point is whether, whatever the preferences still to come after the current point are, that candidate will win. And it is also clear how to judge that. The candidate, as she starts speaking with a given unregistered voter who likes our candidate the most (we can similarly describe how to reason in the other cases), reasons as follows: I need to assume that all future voters (whose preferences I don't currently know!) concentrate their votes on a candidate other than me who currently has the most votes (in the tally I have been building in my canvas so far), and that I use my charisma to (if it is not expended) add the current unregistered voter, and then I suppress those hypothetical, unregistered, against-me voters, and would that leave me a winner of this election? If the answer is yes, then the candidate should be very happy, as she knows she can guarantee herself victory as long as she doesn't later do anything overtly stupid with her charisma.

The example we just gave is in effect explaining why it holds that (so-called constructive) control by adding voters is in polynomial time for sequential plurality elections. Now, one might assume that plurality is such a simple system that for all types of sequential control we will obtain polynomial-time algorithms. However, as Theorem 9 we will show that that is not the case (unless P = NP). In particular, Theorem 9 is about sequentially partitioning voters—the control setting we described above in our example about college admissions. Theorem 9 states that for plurality both the constructive and destructive cases are coNP-hard. The proof of that result is in effect giving a somewhat complex example of transforming a coNP-complete problem, namely the complement of Hitting Set, into an election-control instance about sequentially partitioning voters in a plurality election.

3 Preliminaries

We assume a good grasp of standard complexity-theoretic notions such as the complexity classes P, NP, coNP, and PSPACE, polynomial-time many-one reductions ($\leq_m^p$), $\leq_m^p$-hardness, and $\leq_m^p$-completeness [30,36]. A standard NP-complete problem is the satisfiability problem (SAT) from propositional logic, a standard coNP-complete problem is the tautology problem, and the quantified boolean formula problem (QBF) is a standard PSPACE-complete problem.

We will later use the famous result, due to Chandra, Kozen, and Stockmeyer [7], that PSPACE is exactly the class of languages that can be accepted in polynomial time by what are known as alternating Turing machines. (Alternating Turing machines are, very loosely put, much like nondeterministic Turing machines, except rather than being allowed only existential branching nodes, they are allowed both existential and universal branching nodes.) Such alternations have a very "game-like" feel to them, and indeed the generalized versions of such games as checkers [14], chess [42], and GO [32] are known to be complete for PSPACE. Our paper's setting also has a very game-like feel, with the chair in some sense playing against whatever information potentially is going to be revealed, and we in fact will ourselves obtain a number of PSPACE-completeness results. A standard fact about PSPACE that we will tacitly draw on at a few points is that PSPACE $=$ NP $\iff$ PSPACE $=$ NP $\cap$ coNP; the "if" direction follows from the fact that NP $\cap$ coNP $\subseteq$ NP $\subseteq$ PSPACE and the "only if" direction follows from the fact that PSPACE is closed under (set-wise) complementation, i.e., PSPACE $= \{L \mid \overline{L} \in$ PSPACE$\}$.

This paper provides both polynomial-time algorithms and NP-completeness results. The latter are worst-case results, and so it is possible that for certain distributions heuristics might do well (see [38] for a survey of this in the context of elections). We commend this direction as an area for future research. However, such studies are quite dependent on distributions, and by relatively recent work, it is known that for the uniform distribution heuristic algorithms cannot asymptotically have subexponential error frequency on any NP-hard problem unless the polynomial hierarchy collapses to (and indeed, slightly further than) its third level [3,6,31]. (Note: An algorithm is said to have subexponential error frequency if for every $\epsilon > 0$ the number of errors the algorithm makes at length n is $O(2^{n^{\epsilon}})$; see [31] for a more detailed explanation.)

3.1 Voter control types in simultaneous elections

A pair (C, V) is called a *(standard or simultaneous) election* if C is a set of candidates and V a list of voters that all have cast their votes simultaneously. We assume that each vote in V has the form (v, p), where v is the name of this voter and p is v's (total) preference order over C. For example, if $C = \{c, d, e\}$ then $(\text{Bob}, d > e > c) \in V$ indicates that Bob (strictly) prefers d to e and e to c (or, to be more precise, it indicates that that is the ballot Bob has cast).

The standard types of (constructive) voter control in simultaneous elections are as follows. (These are as introduced by Bartholdi, Tovey, and Trick [5], except here we will follow the now more standard model—called the nonunique-winner model—of asking whether a candidate can be made *a* winner, rather than their approach—called the unique-winner model—of asking whether a candidate can be made the one and only winner.) An election system is a mapping from elections (votes/candidates) to a winner set. Let $\mathscr{E}$ be a given election system. In *control by deleting voters* ($\mathscr{E}$-CCDV), given an election (C, V), a distinguished candidate $c \in C$, and a nonnegative integer $k \leq \|V\|$, we ask whether there exists a set of at most k voters from V such that c is an $\mathscr{E}$ winner of the election in which that set of voters is removed. In *control by adding voters* ($\mathscr{E}$-CCAV), we are given a candidate set C, a list V of registered voters with preferences over C, a list V' of as yet unregistered voters with preferences over C, a distinguished candidate $c \in C$, and a nonnegative integer $k \leq \|V'\|$, and the question is whether there exists a set of at most k voters from V' such that c is an $\mathscr{E}$ winner of the election where the voters are that set and all of V. Finally, in *control by partition of voters*, we are given an election (C, V) and a distinguished candidate $c \in C$, and we ask whether V can be partitioned into two sublists, V_1 and V_2, such that c is an $\mathscr{E}$ winner of the election $(W_1 \cup W_2, V)$, where W_i for $i \in \{1, 2\}$ is the (possibly empty) set of winners of subelection (C, V_i) that have survived the tie-handling rule used and by V here we implicitly mean V masked down to just those candidates in $W_1 \cup W_2$. Of the two tie-handling models introduced by Hemaspaandra et al. [23] we focus on the *ties-promote* (*TP*) model only, where all winners of a subelection proceed to the runoff, since that model fits more naturally with the nonunique-winner model in which we will define our online control problems. The resulting problem is denoted by $\mathcal{E}$-CCPV.

The destructive variants of these three problems, denoted by $\mathcal{E}$-DCDV, $\mathcal{E}$-DCAV, and $\mathcal{E}$-DCPV, are obtained by requiring that the distinguished candidate c is *not* a winner of the election resulting from the control action at hand [23].

3.2 Online voter control in sequential elections

We study *online voter control in sequential elections*, where we assume that the voters vote in order, one after the other, each expressing preferences over all the candidates. If u is the current voter and C the given candidate set, an *election snapshot for C and u* is specified by a triple $V = (V_{<u}, u, V_{u<})$, where the earlier voters $V_{<u}$ have already cast their votes, each a preference order over C, and now it is u's turn to cast a vote, and the future voters $V_{u<}$ will cast their votes in the order listed. ($V_{<u}$ and u of course list the votes cast and who cast them, but $V_{u<}$ just gives the order of the voters following u.) This snapshot approach is natural for studying online attacks on elections, and was used previously to study the different type of attack known as online manipulation in sequential elections [27,28].

We now define our notions of online voter control for the standard voter control types stated above, and the related problems. They all will start from a basic *online voter control setting* (an *OVCS*, for short), augmented by appropriate additional information according to the control type at hand. A basic OVCS (C, u, V, σ, d) consists of a set C of candidates, the current voter u (which isn't strictly needed here, as u is clearly singled out within V anyway), an election snapshot V for C and u, the chair's preference order σ on C, and a distinguished candidate $d \in C$. Let $\mathcal{E}$ be a given election system and let $W_{\mathcal{E}}(C, V)$ denote the $\mathcal{E}$ winner set of (standard) election (C, V). For each online voter control type we will define, the question the chair faces is: Does there exist a control-action choice of our considered type regarding the current voter (e.g., whether or not to delete u) such that if the chair takes that action, then no matter what votes the remaining voters after u cast, the chair's goal can be reached by the current decision regarding u and by using the chair's future decisions (if any), each being made using the chair's then-in-hand knowledge about what votes have been cast by then?[3] By *the chair's goal* we mean to ensure $W_{\mathcal{E}}(C, V') \cap \{c \mid c \geq_\sigma d\} \neq \emptyset$ for each possible ultimate election (C, V') (i.e., each V' is a possible vote list resulting from the control type at hand after all decisions have been made by the chair and all voters have cast their votes) in the constructive case, and to ensure that $W_{\mathcal{E}}(C, V') \cap \{c \mid d \geq_\sigma c\} = \emptyset$ in the destructive case (i.e., that neither d nor any candidate the chair likes even less than d is a winner).[4] Note that

[3] Note that this maxi-min-inspired (but with more quantifiers) approach is really about alternating quantifiers. We are asking if there exists a current action of the chair, such that for all potential revealed vote values that come between now and the next time the chair has to decide on an action, there exists a next action by the chair, such that for all ...... the chair reaches her goal.

[4] Why do we provide an ordering σ rather than just providing as a list the set of candidates who are good enough to count as reaching our goal? For the decision-problem version of online voter control, which is our formulation here, providing such a set would be just as good. But by making σ a part of the input, we make the model compatible, for the future, with the interesting optimization problem of trying to find the most preferred candidate within σ for which the chair can ensure that there is among the winner set one of the candidates in the segment from that candidate to the top candidate in σ.
Also, to avoid any confusion, we note that in our "d chooses an upper (constructive case) or lower (destructive case) segment of the candidates" approach, the non-online version's situation that the destructive goal "opposing" a constructive goal is specified in the same way not longer holds (although we could have defined things in a less natural way so that that would hold). That is, in the non-online setting, the distinguished candidate d in the constructive case is saying who the chair wants to win, and in the destructive case is saying who the chair wants to not win; d in one case is defined in the problem definition to denote the beloved candidate

the conditions $W_{\mathscr{E}}(C, V') \cap \{c \mid c \geq_\sigma d\} \neq \emptyset$ and $W_{\mathscr{E}}(C, V') \cap \{c \mid d \geq_\sigma c\} = \emptyset$ defining the chair's goal have the flavor, give or take the fact that we are focusing on a top segment of σ, of the nonunique-winner model, e.g., as long as $W_{\mathscr{E}}(C, V') \cap \{c \mid c \geq_\sigma d\} \neq \emptyset$ we call it success even if more than one candidate ties as winner. To formally define our problems, it remains to specify for each control type the information by which the basic OVCS is augmented. What kind of decisions the chair is to make in the course of a sequential election will always be clear from the control type at hand (e.g., whether or not to delete a voter in "online control by deleting voters").

Let $B = (C, u, V, \sigma, d)$ be a given basic OVCS. For *online control by deleting voters*, B is augmented by the following additional information: A nonnegative integer k (the deletion upper bound); for each voter v before u, there is a flag saying whether v was deleted and the vote cast by v (if not deleted)—at most k voters can be marked as deleted for the input to be syntactically legal; and the vote the current voter u will cast (if not selected for deletion). We denote these problems by online-$\mathscr{E}$-CCDV (constructive) and online-$\mathscr{E}$-DCDV (destructive). (We certainly could equivalently formulate the problem in a way that masks out all earlier deleted voters, and so removes the need for the flagging; but we prefer the above version since it allows the actual history of the voting situation to be part of the instance.)

For *online control by adding voters*, B is augmented by the following additional information: A nonnegative integer k (the addition upper bound); each voter v in V has a flag saying if v is unregistered (i.e., can be added) or registered—u must be unregistered; each unregistered voter v before u has another flag saying if v was added—at most k voters may have that flag set in any syntactically legal input; the vote cast is given for each registered or added unregistered voter before u; and also given is the vote u will cast (if it is added). We denote these problems by online-$\mathscr{E}$-CCAV (constructive) and online-$\mathscr{E}$-DCAV (destructive).

For *online control by partition of voters*, B is augmented by the following additional information: Each voter v before u has a flag saying which part of the partition v was assigned to ("left" or "right") and the vote cast by v, and also u's vote is given. We denote these problems by online-$\mathscr{E}$-CCPV (constructive) and online-$\mathscr{E}$-DCPV (destructive). As a reminder, the two preliminary elections are conducted under the convention that "ties promote" (i.e., all winners of the preliminary elections move forward to the final election).

A natural worry about our maxi-min approach to online voter control is that it is always possible that all the future voters are hostile to one's goals. And in that case, one may be, depending on the election system, powerless to reach one's goal in the worst case, and so the maxi-min outcome is easily seen to be failure to reach one's goal. This worry exists in a weaker form for online manipulation and online bribery; that is because, for those, if one is allowed almost no vote-changing one is in many cases obviously in trouble, at least in those settings where one can do whatever one wants to those votes one does manipulate or bribe.

Footnote 4 continued
and in the other case is defined to denote the despised candidate. However, in our case, we are giving an order σ, and it would be perverse and confusing to have $>$ mean one thing for constructive and another for destructive. And so, as we have defined things, if the chair's stated ordering σ is $v_1 > v_2 > v_3 > v_4 > v_5$ and $d = v_2$, in the constructive case that means that the chair wants at least one of v_1 or v_2 to win. To state the destructive-case goal—which in some sense is the "flip" of that constructive-case goal—of having neither v_1 nor v_2 be a winner, one would give as the chair's ordering $v_5 > v_4 > v_3 > v_2 > v_1$ and $d = v_2$, since this specifies that v_2 and v_1 are the chair's two most despised candidates and are the ones the chair wants to prevent from being winners.
These comments simply refer to the way various "opposite" goals happen to be expressed. None of the above is saying that the constructive *problem* (viewed as a set) and the destructive *problem* (viewed as a set) are each other's complements. Due to the quantification involved regarding the actions being taken such as by the chair, that is not true.

However, in control one doesn't get to set the value of a single vote, and that is quite extreme indeed.

This is a valid worry, but there are some things that keep it in perspective. Primarily, our paper is trying to find out the very greatest complexity that online control in sequential elections can ever have (when restricted to election systems having polynomial-time winner problems). And so we can look at election systems that sidestep the above worry, due to their properties simply not matching the intuition above (which assumed that we are using an election system in which having a lot of bad-for-us votes results in a bad-for-us outcome). In effect, we are seeking to understand the limits of behavior, in order to set a bounding box on the behaviors that can be realized. Of course, for many natural election systems, the effect mentioned in the previous paragraph will hold, and for many inputs that fact can be exploited to help achieve polynomial-time algorithms for the control problem; indeed, in this paper itself, we give examples of achieving polynomial-time algorithms for the most important of election systems: plurality. Of course, problems may start with some votes already cast, and this may itself make for interesting "endgame" decision issues. We also very much hope further studies will be conducted employing a range of models, including ones beyond maxi-min.

4 General upper and lower bounds

Theorem 1 *For each election system $\mathscr{E}$ with a polynomial-time winner problem,*[5] *online-$\mathscr{E}$-*CCDV*, online-$\mathscr{E}$-*DCDV*, online-$\mathscr{E}$-*CCAV*, online-$\mathscr{E}$-*DCAV*, online-$\mathscr{E}$-*CCPV*, and online-$\mathscr{E}$-*DCPV *are in* PSPACE.

Proof The upper bounds follow from the observation that each of these problems can be solved by an alternating Turing machine in polynomial time, and thus by a deterministic polynomial-space Turing machine, by the characterization due to Chandra, Kozen, and Stockmeyer [7] that was mentioned in the Preliminaries.

We won't provide the detailed programming of the alternating Turing machines but, taking online-$\mathscr{E}$-CCDV as an example, the alternating Turing machine's alternations will be that it will ask whether there exists a (legal) decision by the chair at the current point, such that for every (legal) next revelation as to voter preference, there exists a (legal) decision by the chair at that point, such that for every (legal) next revelation as to voter preference, ..., such that the chair's goal is obtained. This is an at most polynomial-in-the-input-size long sequence of alternating existential and universal blocks—precisely the action of a polynomial-time alternating Turing machine. □

Theorem 1 settles all general (i.e., regarding any voting system for which winner determination is easy) upper bounds for our online voter control problems. We now turn to exploring their lower bounds.

4.1 Control by deleting and by adding voters

Theorem 2 *There exist election systems $\mathscr{E}$ and $\mathscr{E}'$ with polynomial-time winner problems such that online-$\mathscr{E}$-*CCDV*, online-$\mathscr{E}$-*CCAV*, online-$\mathscr{E}'$-*DCDV*, and online-$\mathscr{E}'$-*DCAV *are* PSPACE*-complete, even when limited to two candidates.*

[5] The statement of Theorem 1 holds even for election systems whose winner problems are in PSPACE.

Proof This is a rather difficult construction, and those not very comfortable with the construction of reductions showing PSPACE-hardness might want to skip over it, especially during a first reading. But let us at least quickly, in this paragraph, describe very informally what is going on in the proof. Let us take online constructive control by deleting voters as our example here. The key idea is to take a problem that is already known to capture the full power of PSPACE (in our case below, a problem called QBF$'$, which is a rather technical variant of the most famous PSPACE-complete problem, QBF), and show how we can create an election system, having a polynomial-time winner problem, such that the online constructive control by deleting voters problem for that election system can in effect solve QBF$'$ problems. To do this, we create an election system that itself internally interprets its own input—the election instance (C, V)—in a way that crosses the gap between formulas and elections. This does involve a rather complex interpretation scheme. The key things being done below thus are the creation of such an election system ($\mathscr{E}$), and the definition of the transformation (in particular, a polynomial-time many-one reduction) that converts a question of the form "Is F in QBF$'$?" into an election instance that under that election system, for the online constructive control by deleting voters problem, is going to in effect perfectly capture the issue that "$F \in$ QBF$'$?" is framing.

We define election system $\mathscr{E}$ as follows. We will describe what that election system does on election instance (C, V). $\mathscr{E}$ interprets—in some fixed, natural encoding—the lexicographically least candidate name in C as a boolean formula, Φ, whose variable names must be the strings $x_1, x_2, \ldots, x_{2\ell}$ for some ℓ, where $x_{2\ell}$ actually appears in Φ (the other variables don't have to; no variables other than $x_1, x_2, \ldots, x_{2\ell}$ are allowed). If these syntactic requirements fail to hold, everyone loses in $\mathscr{E}$. Otherwise, if any two voters in V have the same name, everyone loses in $\mathscr{E}$. Otherwise, order the voters in V lexicographically by name of the voter, and let $v_1, v_2, \ldots, v_z$ be the voter names in this order. If $z < 2\ell$ or if there are fewer than two candidates, everyone loses in $\mathscr{E}$. Otherwise, if for some odd i, $1 \leq i \leq 2\ell - 1$, the two lowest order bits of v_i are not 00 or 01, or if for some even i, $2 \leq i \leq 2\ell$, the two lowest order bits of v_i are not 10 or 11, everyone loses in $\mathscr{E}$. Otherwise, assign the variables of $\Phi(x_1, x_2, \ldots, x_{2\ell})$ as follows. For each odd i, $1 \leq i \leq 2\ell - 1$, set x_i to *true* if the two lowest order bits of v_i are 01, and set x_i to *false* otherwise (i.e., the two lowest order bits of v_i are 00). For each even i, $2 \leq i \leq 2\ell$, set x_i to *true* if the name of the least preferred candidate in the vote of v_i is lexicographically less than the name of the next to least preferred candidate in the vote of v_i, and set x_i to *false* otherwise. If this assignment satisfies Φ, everyone wins in $\mathscr{E}$, and otherwise everyone loses. This ends the specification of $\mathscr{E}$. Since a boolean formula whose variables have all been assigned can be evaluated in polynomial time, $\mathscr{E}$ has a polynomial-time winner problem.

By Theorem 1, online-$\mathscr{E}$-CCDV is in PSPACE. To show PSPACE-hardness of online-$\mathscr{E}$-CCDV, we $\leq_m^p$-reduce the PSPACE-complete problem QBF$'$, a variant of QBF, to it. (Why does this show it is PSPACE-hard? A standard approach in complexity is to show that a problem A is PSPACE-hard by $\leq_m^p$-reducing a known PSPACE-complete—or even a known PSPACE-hard—problem to A. Due to the transitivity of $\leq_m^p$, building that one reduction establishes that every set in PSPACE $\leq_m^p$-reduces to A, i.e., it shows that A is PSPACE-hard.) QBF$'$ is the set of boolean formulas of the form $F(x_1, x_2, \ldots, x_{2\ell})$, for some ℓ, such that the variable $x_{2\ell}$ appears in F, all variables appearing in F are from the variable name collection "x_1", "x_2", ..., "$x_{2\ell}$", and

$$(\exists b_1)\,(\forall b_2)\,\cdots\,(\exists b_{2\ell-1})\,(\forall b_{2\ell})\,[F(x_1 := b_1, x_2 := b_2, \ldots, x_{2\ell} := b_{2\ell}) \text{ evaluates to } \textit{true}],$$

where $b_i \in \{0, 1\}$ and $x_i := b_i$ means that variable x_i is set to *true* if $b_i = 1$, and is set to *false* if $b_i = 0$, for $1 \leq i \leq 2\ell$.

Let $F(x_1, x_2, \ldots, x_{2\ell})$ be a given instance of QBF$'$, where $x_{2\ell}$ explicitly appears in F. (If our input is syntactically incorrect, we map it to a fixed no-instance of online-$\mathscr{E}$-CCDV.) We construct from F an instance of online-$\mathscr{E}$-CCDV, consisting of a basic OVCS (C, u, V, σ, d), augmented by the additional information of online control by deleting voters, as follows. Define $C = \{a, b\}$, where a encodes F (in our fixed, natural encoding of boolean formulas) and b is the string lexicographically immediately following a; the current voter is $u = v_1$; V will be specified below; the chair's preference order is $a >_\sigma b$; for specificity, we let $d = a$ be the distinguished candidate (though that does not matter, as all candidates win or all lose in $\mathscr{E}$); the deletion limit is $k = \ell$; and a vote $a > b$ to cast for u if not deleted (again, the vote doesn't matter, as $u = v_1$ will specify an assignment to x_1 by her name, not by her vote). There are $(3/2)\cdot 2\ell = 3\ell$ voters in V such that the name of the ith voter, v_i, is the binary string $u_i w_i$, where u_i is the binary representation of i and $w_i = 00$ if $i \equiv 1 \bmod 3$, $w_i = 01$ if $i \equiv 2 \bmod 3$, and $w_i = 10$ if $i \equiv 0 \bmod 3$, $1 \leq i \leq 3\ell$. This completes the description of our $\leq_m^p$-reduction from QBF$'$ to online-$\mathscr{E}$-CCDV, which clearly can be computed in polynomial time.

We claim that $F \in$ QBF$'$ exactly if the chair's goal can be reached by at most k deletions of voters. Why? By the definition of $\mathscr{E}$, everyone loses unless our $k = \ell$ deletions are used on *exactly one of* v_{3i-2} *and* v_{3i-1}, for each i, $1 \leq i \leq \ell$. No v_{3i}, $1 \leq i \leq \ell$, can be deleted if there is to be a winner. And the "exactly one of v_{3i-2} and v_{3i-1}" choices, $1 \leq i \leq \ell$, specify an assignment of truth values to the odd-numbered variables: For each i, $1 \leq i \leq \ell$, x_{2i-1} is set to *true* if v_{3i-2} is deleted and v_{3i-1} is not, and is set to *false* if v_{3i-1} is deleted and v_{3i-2} is not. On the other hand, for each i, $1 \leq i \leq \ell$, the truth value of x_{2i} is specified by the *vote* of voter v_{3i}, since after these ℓ deletions, v_{3i} will be the $2i$th voter name in the lexicographic order. It follows that the chair's goal can be reached by at most k deletions of voters exactly if $(\exists b_1)\,(\forall b_2) \cdots (\exists b_{2\ell-1})\,(\forall b_{2\ell})\,[F(x_1 := b_1, x_2 := b_2, \ldots, x_{2\ell} := b_{2\ell})$ evaluates to *true*]. In light of the definition of QBF$'$, that is equivalent to saying that the chair's goal can be reached by at most k deletions of voters exactly if $F \in$ QBF$'$. And that is precisely what this paragraph was seeking to establish.

PSPACE-hardness of online-$\mathscr{E}$-CCAV for the election system $\mathscr{E}$ defined above can be shown via essentially the same $\leq_m^p$-reduction from QBF$'$. The only difference is that we now map the given QBF$'$ instance F to an instance of online-$\mathscr{E}$-CCAV, which is defined exactly as the online-$\mathscr{E}$-CCDV instance constructed above, except that all voters v_i with $i \equiv 0 \bmod 3$ are specified as registered voters, and all other voters are unregistered. The correctness argument is analogous.

The destructive cases can be shown analogously, by modifying the election system $\mathscr{E}$ defined above as follows, yielding our modified system $\mathscr{E}'$: Whenever everyone loses (wins) in $\mathscr{E}$, everyone wins (loses) in $\mathscr{E}'$. It follows from Theorem 1 and the above $\leq_m^p$-reduction from QBF$'$ that online-$\mathscr{E}'$-DCDV and online-$\mathscr{E}'$-DCAV are both PSPACE-complete. $\square$

For control by deleting or adding voters, the deletion or addition limit k is—both in the non-online case and in our online definition (which is what is used in Theorem 2)—part of the problem instance. To better understand the source of the tremendous level of computational hardness Theorem 2 showed that these problems can have, let us now consider restrictions of these problems in which the deletion or addition limit is bounded by a constant. For a given election system $\mathscr{E}$ and a fixed k, let online-$\mathscr{E}$-CCDV$[k]$ be the restriction of online-$\mathscr{E}$-CCDV to those inputs whose deletion limit is at most k, and define the problem variant online-$\mathscr{E}$-CCAV$[k]$ analogously. We will show in Theorem 4 that this change in the definition—bounding the deletion/addition bound—brings the complexity of these problems from PSPACE down to coNP. (In contrast, limiting the number of *candidates* to two was shown by Theorem 2 to leave these two problems PSPACE-complete.)

The coNP upper bound follows immediately from the following theorem about restricted polynomial-time alternating Turing machines, which is of interest in its own right. If we define the weight of a path of an alternating Turing machine to be the number of 1's in the existential guesses along the path, what Theorem 3 says is that the class of languages accepted by polynomial-time alternating Turing machines whose accepting paths are weight-bounded is precisely coNP.

Theorem 3 *Let $k \geq 0$. The class of languages accepted by polynomial-time alternating Turing machines that satisfy the property that on each accepting computation path the number of existential guesses on which the bit is guessed as 1 is at most k is precisely* coNP.

Proof We will show this by induction on k.

The $k = 0$ case is precisely coNP. This is so simply because $k = 0$ fixes each existential guess (namely, of choosing between 0 and 1) to be a guess choosing 0, which in effect makes it not an existential guess at all.

To prove the inductive step, let $k > 0$ and let A be a language accepted by a polynomial-time alternating Turing machine that satisfies the property that on each accepting computation path the number of existential guesses on which the bit is guessed as 1 is at most k. That is, any path that contains at least $k + 1$ guessed 1's in its existential guesses must have as its (leaf) value Reject rather than Accept.[6] We will show that A is in coNP.

Throughout this proof, all x_i's and y_i's are over $\{0, 1\}$, i.e., are bits.

Let B be a polynomial-time computable ternary predicate and let $\ell(n)$ be a polynomial such that for all x, $x \in A$ if and only if

$$\forall x_1 \exists y_1 \ \forall x_2 \exists y_2 \ \ldots \forall x_{\ell(|x|)} \exists y_{\ell(|x|)} \left(B(x, x_1 \ldots x_{\ell(|x|)}, y_1 \ldots y_{\ell(|x|)}) \wedge \sum_{i=1}^{\ell(|x|)} y_i \leq k \right).$$

(To be perfectly clear about what x is in the displayed equation above, we mention again that the above equation is capturing whether a string x is a member of A, i.e., whether $x \in A$.) Such a polynomial, ℓ, and predicate, B, exist, since we can add extra quantifiers with dummy variables to make the quantifiers alternating and we can always guess an existentially-quantified dummy variable as 0.

We can rewrite the above as follows. For all x, $x \in A$ if and only if

$$\begin{aligned}
&\forall x_1 \ (\forall x_2 \exists y_2 \ \forall x_3 \exists y_3 \ \ldots \ \forall x_{\ell(|x|)} \exists y_{\ell(|x|)} \left(B(x, x_1 \ldots x_{\ell(|x|)}, 1y_2 \ldots y_{\ell(|x|)}) \wedge \textstyle\sum_{i=2}^{\ell(|x|)} y_i \leq k-1\right) \vee \\
&\quad \forall x_2 \ (\forall x_3 \exists y_3 \ \ldots \ \forall x_{\ell(|x|)} \exists y_{\ell(|x|)} \left(B(x, x_1 \ldots x_{\ell(|x|)}, 01y_3 \ldots y_{\ell(|x|)}) \wedge \textstyle\sum_{i=3}^{\ell(|x|)} y_i \leq k-1\right) \vee \\
&\qquad \forall x_3 \ (\ldots \ \forall x_{\ell(|x|)} \exists y_{\ell(|x|)} \left(B(x, x_1 \ldots x_{\ell(|x|)}, 001y_4 \ldots y_{\ell(|x|)}) \wedge \textstyle\sum_{i=4}^{\ell(|x|)} y_i \leq k-1\right) \vee \\
&\qquad\quad \vdots \\
&\qquad\qquad \forall x_{\ell(|x|)} \quad \left(B(x, x_1 \ldots x_{\ell(|x|)}, 0^{\ell(|x|)-1}1) \wedge \textstyle\sum_{i=\ell(|x|)+1}^{\ell(|x|)} y_i \leq k-1\right) \vee \\
&\qquad\qquad\qquad B(x, x_1 \ldots x_{\ell(|x|)}, 0^{\ell(|x|)}) \ldots))).
\end{aligned}$$

(Of course, $\sum_{i=\ell(|x|)+1}^{\ell(|x|)} y_i \leq k-1$ is true, since $k > 0$ in the present case, and the sum is an empty sum and so by convention evaluates to 0.) The long expression above is not quite in

[6] Recall that each path of a polynomial-time alternating Turing machine has as its individual (leaf) value either Accept or Reject, and the overall action of the Turing machine is determined by the thought-experiment of applying the existential and universal node actions of the machine to those leaf values, resulting in an Accept or Reject at the root that determines the machine's acceptance or rejection on the given input.

the right form to apply the inductive hypothesis. In order to be able to do so, define language C such that $\langle x, x_1 \ldots x_r \rangle \in C$ if and only if $r \leq \ell(|x|)$ and

$$\forall x_{r+1} \exists y_{r+1} \ldots \forall x_{\ell(|x|)} \exists y_{\ell(|x|)} \left(B(x, x_1 \ldots x_{\ell(|x|)}, 0^{r-1} 1 y_{r+1} \ldots y_{\ell(|x|)}) \wedge \sum_{i=r+1}^{\ell(|x|)} y_i \leq k-1 \right).$$

Clearly C can be accepted by a polynomial-time alternating Turing machine that satisfies the property that on each accepting computation path the number of existential guesses on which the bit is guessed as 1 is at most $k-1$. By the inductive hypothesis, C is in coNP. Since $x \in A$ if and only if

$$\forall x_1(\langle x, x_1 \rangle \in C \vee \forall x_2(\langle x, x_1 x_2 \rangle \in C \vee \forall x_3(\langle x, x_1 x_2 x_3 \rangle \in C \vee \\ \ldots \forall x_{\ell(|x|)}(\langle x, x_1 x_2 x_3 \ldots x_{\ell(|x|)} \rangle \in C) \vee B(x, x_1 \ldots x_{\ell(|x|)}, 0^{\ell(|x|)}) \ldots))),$$

it follows that A is in coNP. (Why is it in coNP? Note that its complement is in NP due to having a polynomial-length witnesses. Let N be a fixed NP Turing machine accepting $\overline{C}$. Our witness scheme for membership in $\overline{A}$ is: Guess an $x_1, \ldots, x_{\ell(|x|)}$ such that $B(x, x_1 \ldots x_{\ell(|x|)}, 0^{\ell(|x|)})$ holds and also guess for each of $\langle x, x_1 \rangle$, $\langle x, x_1 x_2 \rangle$, ..., $\langle x, x_1 x_2 \ldots x_{\ell(|x|)} \rangle$ an accepting path of N on that input.) □

Theorem 4 *For each $k \geq 0$, the following hold:*

1. (a) *For each election system $\mathscr{E}$ with a polynomial-time winner problem, online-$\mathscr{E}$-*CCDV$[k]$ *is in* coNP. (b) *There exists an election system $\mathscr{E}$ with a polynomial-time winner problem such that online-$\mathscr{E}$-*CCDV$[k]$ *is* coNP*-complete, even when limited to two candidates.*
2. (a) *For each election system $\mathscr{E}$ with a polynomial-time winner problem, online-$\mathscr{E}$-*CCAV$[k]$ *is in* coNP. (b) *There exists an election system $\mathscr{E}$ with a polynomial-time winner problem such that online-$\mathscr{E}$-*CCAV$[k]$ *is* coNP*-complete, even when limited to two candidates.*

Proof Sketch Parts 1(a) and 2(a) follow immediately from Theorem 3.

Now consider part 1(b). Even for $k = 0$ (and in effect so for all k, as those have within them $k = 0$ as subcases we can map to) we claim that there is an election system $\mathscr{E}$ with a polynomial-time winner problem such that online-$\mathscr{E}$-CCDV$[k]$ is easily shown to be coNP-hard, namely by a $\leq_m^p$-reduction from the coNP-complete tautology problem. The mapping and $\mathscr{E}$ are inspired by the proof of Theorem 2: We use the lexicographically least candidate name to be a proposed tautology and we use the voters as tests of various assignments to it (if the assignment satisfies, everyone wins). So the problem can force the chair's top choice (candidate a, see the proof of Theorem 2) to win exactly if the formula is a tautology. As in the statement and proof of Theorem 2, this reduction maps to outputs having only two candidates.

The proof sketch for part 2(b) (online-$\mathscr{E}$-CCAV$[k]$) is similar to that of part 1(b). The first (and current) voter in our reduction is unregistered (but with $k = 0$ she obviously cannot be added), and the remaining voters are testing assignments to a proposed tautology and we have only two candidates, just as in the above proof sketch for online-$\mathscr{E}$-CCDV$[k]$. □

4.2 Control by partition of voters

Theorem 5 *There exist election systems $\mathscr{E}$ and $\mathscr{E}'$, whose winner problems can be solved in polynomial time, such that online-$\mathscr{E}$-*CCPV *and online-$\mathscr{E}'$-*DCPV *are* PSPACE*-complete, even when limited to two candidates.*

Proof This proof is similar in flavor to the proof of Theorem 2, but since we now handle control by partition of voters, there are some decisive differences.

The election system $\mathscr{E}$ is now defined as follows.

Case 1: There is a candidate named RoundOne, and no voter is named Marker. In this case, everyone loses.

Case 2: There is a candidate named RoundOne and a voter named Marker. In this case, interpret—in our fixed, natural encoding—the lexicographically least candidate not named RoundOne as a boolean formula, Φ, whose variable names must be the strings $x_1, x_2, \ldots, x_{2\ell}$ for some ℓ, and $x_{2\ell}$ must actually appear in Φ (the others do not have to, but no variable other than $x_1, x_2, \ldots, x_{2\ell}$ can appear in Φ). If this candidate is not of the required syntactic form, exactly RoundOne wins. If the candidate set does not consist of exactly RoundOne and the above candidate, then exactly RoundOne wins. If the voter list consists of exactly $2\ell + 1$ voters such that one voter is named Marker, one voter is named v_1^{yes} or v_1^{no}, one voter is named v_2, one voter is named v_3^{yes} or v_3^{no}, ..., one voter is named $v_{2\ell-1}^{\text{yes}}$ or $v_{2\ell-1}^{\text{no}}$, and one voter is named $v_{2\ell}$, where all subscripts are given in binary, then assign the 2ℓ variables of Φ as follows. (If the voter list is not exactly that then exactly RoundOne wins.) For each odd i, $1 \leq i \leq 2\ell - 1$, set x_i to *true* if there is a voter named v_i^{yes} and to *false* if there is a voter named v_i^{no}. For each even i, $2 \leq i \leq 2\ell$, set x_i to *true* if the voter named v_i has the property that in her preference order RoundOne is the top choice, and otherwise set x_i to *false*. If this assignment makes Φ *true*, then the candidate not named RoundOne is the only winner, otherwise (exactly) RoundOne wins.

Case 3: There is no candidate named RoundOne. In this case, everyone wins.

This ends the specification of $\mathscr{E}$. Clearly, $\mathscr{E}$ has a polynomial-time winner problem, since it is just evaluating a fully specified and assigned boolean formula, and doing various syntactic checks.

Our online control by partition of voters problems are all in PSPACE by Theorem 1. To prove PSPACE-hardness, we again $\leq_m^p$-reduce from the PSPACE-complete problem QBF$'$ defined in the proof of Theorem 2. Let $F(x_1, \ldots, x_{2\ell})$ be a given QBF$'$ instance, where $x_{2\ell}$ actually occurs in F. (If our input is syntactically incorrect, then map it to a fixed nonmember of our target problem.) Our candidate set will be $C = \{\text{RoundOne}, a\}$, where a will in her name encode F (without loss of generality, that will not form the string "RoundOne"), a will be our distinguished candidate, our current voter will be $u = \tilde{v}_0$, the chair's preference order will be $a >_\sigma$ RoundOne, and there will be $3\ell + 1$ voters who vote in order $\tilde{v}_0, \tilde{v}_1, \ldots, \tilde{v}_{3\ell}$, where $\tilde{v}_0$ is named Marker, and the remaining voters are named as follows:

voter	$\tilde{v}_1$	$\tilde{v}_2$	$\tilde{v}_3$	$\tilde{v}_4$	$\tilde{v}_5$	$\tilde{v}_6$	$\cdots$	$\tilde{v}_{3\ell-2}$	$\tilde{v}_{3\ell-1}$	$\tilde{v}_{3\ell}$
name	v_1^{yes}	v_1^{no}	v_2	v_3^{yes}	v_3^{no}	v_4	$\cdots$	$v_{2\ell-1}^{\text{yes}}$	$v_{2\ell-1}^{\text{no}}$	$v_{2\ell}$

This ends our statement of the reduction. Why does it work?

If $F \in$ QBF$'$, then

$$(\exists b_1)\,(\forall b_2)\,\cdots(\exists b_{2\ell-1})\,(\forall b_{2\ell})$$
$$[F(x_1 := b_1, x_2 := b_2, \ldots, x_{2\ell} := b_{2\ell}) \text{ evaluates to } \textit{true}], \tag{1}$$

where the $b_i \in \{0, 1\}$ are truth assignments. So the partition that puts Marker and all voters v_i, i even, on one side, say into V_{left}, and for each $v_i^{\text{yes}}/v_i^{\text{no}}$ pair, i odd, follows (1) by putting v_i^{yes} into V_{left} and v_i^{no} into V_{right} if $b_i = 1$, and v_i^{no} into V_{left} and v_i^{yes} into V_{right} if $b_i = 0$ (and crucially note that the preference orders of the v_i, i even, we will have seen in the future can (in the future) effect the future partition choices), will by Case 2 have one first-round election (namely, (C, V_{left})) in which a is the only winner. And in the other first-round election, (C, V_{right}), by Case 1 everyone, including RoundOne, loses. Thus, only a proceeds to the second-round runoff election, where by Case 3 everyone wins, i.e., our distinguished candidate a wins.

In the other direction, suppose F is syntactically correct, and it is possible by some partition of voters to force "a or better" (so a) to be a winner. Since RoundOne is in both first-round elections (so Case 3 cannot occur), the only way candidate a can be guaranteed to even survive at least one first-round election is if we can guarantee that Case 2 is satisfied. But that means that $F \in \text{QBF}'$.

Since our reduction can be computed in polynomial time, this shows that online-$\mathscr{E}$-CCPV is PSPACE-hard.

To show that online-$\mathscr{E}'$-DCPV is PSPACE-hard, we modify the election system $\mathscr{E}$ defined above as follows, yielding our modified system $\mathscr{E}'$: Most crucially, Case 2 of the election system description changes to now making everyone lose if Φ evaluates to *true* under the specified assignment, and if Φ evaluates to *false* (or there is any syntactic problem regarding who is in the voter list) then everyone wins. Case 3 changes to now having everyone lose, and Case 1 stays the same. The $\leq_m^p$-reduction from QBF$'$ remains the same, except that the chair's preference order will now be reversed to RoundOne $>_\sigma a$, and with these changes the reduction can be shown to work correctly by arguments analogous to those in the constructive case. ◻

The above proof establishes that there are election systems, with polynomial-time winner problems, for which constructive and destructive online control by partition of voters are PSPACE-complete even when limited to two candidates. Can we make do with one candidate and still have PSPACE-hardness? The following result shows that if we could, then PSPACE would equal NP $\cap$ coNP.[7]

Theorem 6 1. *For each election system $\mathscr{E}$ with a polynomial-time winner problem, the problems online-$\mathscr{E}$-*CCPV *and online-$\mathscr{E}$-*DCPV *when limited to one candidate are in* NP.
2. *There exist election systems $\mathscr{E}$ and $\mathscr{E}'$ with polynomial-time winner problems such that the problems online-$\mathscr{E}$-*CCPV *and online-$\mathscr{E}'$-*DCPV*, even when restricted to one candidate, are* NP*-complete.*

Proof We give the proof for the destructive case. For the first part, with one candidate, c, every voter has the same preference as her full vote: c. So there is no sequentially revealed information, as in our model we know the voter names (and their order but here that does not matter) as part of our input. So we just in NP can guess every partition of the voters from u, the current voter, onward, and see if one of those meets the chair's destructive goal, "c does not win."

For the second part, membership in NP follows from the first part. As to NP-hardness, let us $\leq_m^p$-reduce from SAT. The election system, $\mathscr{E}'$, is defined as follows:

Case 1: If there are two or more candidates, everyone wins.
Case 2: If there is one candidate and that candidate's name gives a syntactically correct boolean formula φ that has, say, k variables, and there are exactly k voters, and if we set the ith variable of φ to *true* exactly if 1 is the lowest order bit of the voter whose name ranks ith in lexicographic order among the voters' names, then φ is satisfied either by

[7] Are elections with just one candidate ever interesting in the real world? We feel they sometimes are. For example, a popular referendum—or for that matter a vote in a legislature on a bill—is essentially an up-or-down vote on one "candidate." So is a vote on whether to recall an elected official, or to impeach a judge, or to ratify a person who has been nominated for a sports hall of fame.

that assignment or by the bitwise complemented twin of that assignment, then everyone loses.

Case 3: In all other cases (including syntactical problems), everyone wins.

The reduction SAT$\leq_m^p$ online-$\mathscr{E}'$-DCPV is defined as follows. Given a boolean formula $F(x_1, \ldots, x_k)$, where without loss of generality all variables actually appear in F, we construct an online-$\mathscr{E}'$-DCPV instance with candidate set $C = \{c\}$, where c encodes F, the voters are named (in binary) $1, 2, \ldots, 2k$ and they vote in this order, $u = 1$ is the current voter, the distinguished candidate is c, and the chair's preference order σ is c. We claim that c can be made not a winner if and only if F is satisfiable. We will now show that this claim holds.

As to the "if" direction of the claim, if F is satisfiable then we can determine a satisfying assignment by the partition choices we make among each voter pair $(2i - 1, 2i)$, $1 \leq i \leq k$, by choosing exactly one per pair for the right-hand side of the partition, such that the left-hand side of the partition has the bitwise complement of that same satisfying assignment. So, by the definition of $\mathscr{E}'$, c will not be a winner in either first-round subelection, and so will not even be in the final runoff election, which will have zero candidates, and so c will not be a winner.

As to the "only if" direction of the claim, if c loses, by the election rule that proves that (Case 2 in the definition of $\mathscr{E}'$), F is satisfiable.

The constructive case can be shown analogously.

Corollary 1 *The following three statements are equivalent:*

1. PSPACE $=$ NP $\cap$ coNP.
2. *There exists an election system $\mathscr{E}$ with a polynomial-time winner*
3. *problem such that online-$\mathscr{E}$-*DCPV *is* PSPACE*-hard when restricted to one candidate.*
4. *There exists an election system $\mathscr{E}$ with a polynomial-time winner problem such that online-$\mathscr{E}$-*CCPV *is* PSPACE*-hard when restricted to one candidate.*

Proof To show equivalence of the first two statements, suppose PSPACE $=$ NP $\cap$ coNP. So PSPACE $=$ NP. The second statement now follows from the second part of Theorem 6. Conversely, by the second part's hypothesis and the first part of Theorem 6, we have PSPACE $\subseteq$ NP, which (since PSPACE $=$ coPSPACE) is equivalent to PSPACE $=$ NP $\cap$ coNP. The equivalence of the first and the third statements is proven analogously. □

The analogues of the destructive cases of both parts of Theorem 6 also hold when "online" is removed, i.e., for the problem $\mathscr{E}$-DCPV. In contrast, the *constructive non-online* analogue of Theorem 6's first part can be strengthened to a P upper bound. (Why can we get a P result here but not in Theorem 6? The proof of the following result does not apply if some voters are already committed to sides of the partition—it is assuming (and truly using the fact) that we have full control of where *all* voters go. But in the online setting, the current voter u can be a voter who does *not* come first and so some voters may already be assigned to sides of the partition. And why do we get P for constructive but not destructive? The effect the following proof uses is specific to the constructive case.)

Theorem 7 *For each election system $\mathscr{E}$ with a polynomial-time winner problem, $\mathscr{E}$-*CCPV*, when restricted to one candidate, is in* P.

Proof For the one candidate to win, she certainly must win the runoff, in which all voters vote. Also, if she does win when all voters vote, then she can easily be made to survive the first round, using the partition structure $(V, \emptyset)$. It follows from these two observations that constructive (non-online) control by partition of voters is possible if and only if the one candidate wins in the election with voter list V. □

5 Online control for plurality

We have seen in the previous section that online control can be very hard, namely PSPACE-complete, even for voting systems whose winners can be determined in polynomial time. In this section, we study online control for plurality voting.

In this very simple yet popular voting system, every voter gives one point to her most preferred candidate, and all candidates with the most points win. For plurality it is known that non-online control by adding and by deleting voters can be done in polynomial time, both in the constructive and in the destructive cases. For the constructive cases, this is true because the two relevant unique-winner-model results of [5], as noted in [17], also hold in the nonunique-winner model. For the destructive cases, we have checked and here state as true that those unique-winner-model results of [23] are easily seen to also hold in the nonunique-winner model.

We now show that the corresponding types of online control are also easy.

Theorem 8 *The problems online*-plurality-CCDV*, online*-plurality-CCAV*, online*-plurality-DCDV*, and online*-plurality-DCAV *are in* P.

Proof Let us first address the case of online-plurality-CCDV. We now present the polynomial-time algorithm for this case.

The input to the algorithm will be (C, u, V, σ, d), a given basic OVCS, augmented by the additional information of online control by deleting voters: a deletion upper bound k, for each voter v before u a flag saying if v was deleted and the vote cast by v (if not deleted), where at most k voters can be marked as deleted, and a vote to cast for u (if u is not to be deleted).

If d is the chair's bottom choice in σ, we are done, since the input then is trivially in online-plurality-CCDV (unless it is syntactically illegal); this is so since d being the chair's bottom choice in σ means that any nonempty winner set is acceptable, but in plurality the winner set is indeed nonempty.

If exactly k voters have been marked as already deleted, we can do no more deletions, so u and all later voters go in, and we assume (as this is the most challenging case) that all later voters vote for one particular candidate in $\Lambda_d = \{c \in C \mid c <_\sigma d\}$ that among the candidates in Λ_d has the most first place votes after u is put in, and so we can easily answer the online control question.

If fewer than k voters have been selected already for deletion, then delete u if and only if u's top choice is a highest scoring (with respect to the voters before u) candidate in $\{c \in C \mid c <_\sigma d\}$. Then assume that all later voters vote for one particular candidate in $\Lambda_d = \{c \in C \mid c <_\sigma d\}$ that among the candidates in Λ_d has the most first place votes after u is put in. And assume we delete as many of those as the deletion amount left (after u) allows. It is easy to see whether this results in "d or better" being a winner (in which case our algorithm answers "yes") or not (in which case our algorithm answers "no").

This concludes our clearly polynomial-time algorithm for online-plurality-CCDV. (One might comment that it would suffice, especially to just handle the decision version, to follow the very simple "operational" approach mentioned on page 5 of Sect. 2. However, we have given a more dynamic description of the process both as we want to make clear how the chair can decide what action to take at each point and as the description above is also helping establish the correctness of the actions taken.)

For online-plurality-CCAV, let (C, u, V, σ, d) be a given basic OVCS, augmented by the additional information of online control by adding voters: an addition upper bound k, for each voter the information of whether she is registered or not, and for each unregistered voter before u the information of whether she has been added or not, the vote of each registered or added voter before u, and u's potential vote. Again, the question is trivial if d is the chair's bottom choice in σ. Otherwise, we can see what u's vote is and if k has yet been reached. If k has not been reached yet, we add u if and only if u's top choice belongs to $\{c \in C \mid c \geq_\sigma d\}$.[8] And in the worst case all future voters vote for the same member of $\{c \in C \mid c <_\sigma d\}$, which will be one that after u votes has the most first-place votes among those.

The two destructive cases can be handled analogously. The main differences are, in both cases, that the question now is trivial to decide if d is the chair's *top* choice in σ; in the deleting-voters case, that u is to be deleted (provided the deletion limit k has not been reached yet) if and only if u's top choice is a highest scoring (with respect to the voters before u) candidate in $\{c \in C \mid c \leq_\sigma d\}$; and in the adding-voters case, that u is to be added (provided the addition limit k has not been reached yet) if and only if u's top choice belongs to $\{c \in C \mid c >_\sigma d\}$. And, in both cases, we again assume that all future votes will belong to some particular member of $\{c \in C \mid c \leq_\sigma d\}$ that after u votes has the most first-place votes among those candidates. □

Non-online control by partition of voters, in the model we feel is most natural and have adopted in this paper (called "ties promote"), is NP-complete in both the constructive and destructive cases ([23] showed this in the unique-winner model, and we have checked and here state that NP-completeness also holds for the nonunique-winner model analogues). In contrast, the corresponding types of online control are both coNP-hard. This implies that these problems cannot be in NP, unless NP = coNP, which is considered to be highly unlikely. It remains open whether or not they are in coNP; we conjecture that they are not, although admittedly that conjecture is driven by our inability so far to find any way of capturing this problem with a single universal quantification.

Theorem 9 *online*-plurality-CCPV *and online*-plurality-DCPV *are* coNP-*hard.*

Proof We prove this by a reduction from the complement of the following NP-complete problem, Hitting Set: Given a set $B = \{b_1, \ldots, b_m\}$, a nonempty collection $\mathscr{S} = \{S_1, \ldots, S_n\}$ of subsets of B, and a positive integer $k \leq m$, does $\mathscr{S}$ have a hitting set of size at most k, i.e., does there exist a set $B' \subseteq B$ such that $\|B'\| \leq k$ and for all $S_i \in \mathscr{S}$, $S_i \cap B' \neq \emptyset$.

We turn an instance $(B, \mathscr{S}, k)$ of Hitting Set into the following instance of online partition of voters. The set of candidates is $\{c, w, b_1, \ldots, b_m\} \cup A$, where $A = \{a_i \mid 1 \leq i \leq 4mnk+1\}$. The current voter is u. The votes before u that are on the left side of the partition are exactly the same as the votes before u that are on the right side of the partition. Both sides of the partition consists of the following votes.

- $4nk$ votes $c > w > \cdots$, where $\cdots$ denotes that the remaining candidates follow in some arbitrary order.

[8] Sure enough, u's top choice could be one of those candidates that end up having only few votes, so adding u could be a wasted addition that will block some future good addition in some vote sequences, but in the worst case all future voters put first a candidate disliked by the chair; so our action is fine within the quantifier structure of the problem.

- $4nk$ votes $w > c > \cdots$.
- For every i, $1 \leq i \leq n$, $2k$ votes $S_i > c > \cdots$, where S_i denotes the candidates in S_i in some arbitrary order.
- For every j, $1 \leq j \leq m$, as many votes $b_j > B - \{b_j\} > c > w > \cdots$ as needed to make the score of b_j equal to $4nk - 1$ in this subelection.
- For every i, $1 \leq i \leq 4mnk$, one vote $a_i > c > \cdots$ and one vote $a_i > w > \cdots$.

Voter u votes $a_{4mnk+1} > w > \cdots$. And there are k voters after u. The chair's top choice is c and the chair's bottom choice is w, and the distinguished candidate is c in the constructive case (i.e., for online-plurality-CCPV) and w in the destructive case (i.e., for online-plurality-DCPV).

A simple but crucial observation is that no candidate $a \in A$ will ever make it to the final round, since her score in the first round in either subelection will be at most $2 + k$, which is less than c's score in that subelection. (Let us explain where that $2 + k$ bound comes from. Each $a \in A - \{a_{4mnk+1}\}$ gets 2 points from the fifth bullet item above, and gets at most k points from the voters after u, and—keeping in mind that the S_i are subsets of B_i and so do not involve a—gets no other points; so such an a will have a first-round score of at most $2 + k$. a_{4mnk+1} is slightly different; it will not gain the two points from the fifth bullet point as it is excluded from that, but will get one point from voter u, and so its first-round score is at most $1 + k$, which of course is less than $2 + k$.) If both c and w participate in the final round, c gains $8mnk$ points, w gains $8mnk + 1$ points, and no other candidate gains points from the voters specified above whose top choice was in A.

We will show that $\mathscr{S}$ does not have a hitting set of size at most k if and only if c can always be made a winner in the constructed election, and we will show that $\mathscr{S}$ does not have a hitting set of size k if and only if w can always be made to not be a winner in the constructed election. This proves the theorem.

First suppose that $\mathscr{S}$ has a hitting set of size at most k. Let B' be a hitting set of size k. B' exists, since $k \leq m$. Let the k voters after u vote such that the top choice of the ith voter is the ith candidate in B'. Then, no matter how we partition the voters, the set of candidates that participate in the final round is $\{c, w\} \cup B'$. The scores in the final round are as follows: (a) $score(c) = 8nk + 8mnk$, (b) $score(w) = 8nk + 8mnk + 1$, and (c) $\sum_{b \in B'} score(b) = 8mnk - 2m + k$. It follows that c is not a winner and that w is a winner.

For the converse, suppose that $\mathscr{S}$ does not have a hitting set of size at most k. Partition by putting u and all voters after u in the same first-round election. Then the set of candidates in the final round is $\{c, w\} \cup B'$, where $B' \subseteq B$ and $\|B'\| \leq k$. Since B' is not a hitting set, in the final round c gains at least $4k$ points from voters voting $S_i > c > \cdots$ such that $S_i \cap B' = \emptyset$. Thus in the final election the following hold: (a) $score(c) \geq 8nk + 8mnk + 4k$, (b) $score(w) \leq 8nk + 8mnk + 1 + k$, and (c) $\sum_{b \in B'} score(b) \leq 8mnk - 2m + k$. It follows that c is the unique winner of this election. □

6 Conclusions and open questions

Inspired by the maxi-min approach of online algorithms, we studied online voter control in sequential voting. We showed that for suitably constructed election systems with polynomial-time winner problems, the resulting voter-control problems can be extremely hard, namely PSPACE-complete, even for just two candidates. We additionally obtain coNP-completeness for the deleting/adding-voter cases, even for just two candidates, when there is a bounded deletion/addition limit. For plurality, things are easier still: Online control by deleting or adding voters is in polynomial time for plurality, just as in the non-online case.

Attractive future directions include the study of natural election systems in addition to plurality, or even the study of whole classes of natural election systems. Can one obtain PSPACE-completeness results for existing systems, for example? Another interesting direction would be to investigate online control through a typical-case analysis of heuristic approaches (such as, for example, [22,34] do rigorously in a winner-problem setting, see also [38]).

Acknowledgements Parts of this paper appeared in preliminary form in ECAI-2012 [26]. We are very grateful to the referees and editor for their comments and suggestions. The authors' work was supported in part by Grants ARC-DP110101792, DFG-RO-1202/15-1, and NSF-CCF-{0915792,1101452,1101479}, and by COST Action IC1205, Friedrich Wilhelm Bessel Research Awards from the Alexander von Humboldt Foundation, the NRW-MIWF project "Online Partizipation," and the SFF grant "Cooperative Normsetting" from HHU Düsseldorf. Work done in part while E. and L. Hemaspaandra were visiting Heinrich-Heine-Universität Düsseldorf, and while J. Rothe was visiting the University of Rochester.

References

1. Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge: Cambridge University Press.
2. Baumeister, D., Erdélyi, G., Erdélyi, O., & Rothe, J. (2013). Computational aspects of manipulation and control in judgment aggregation. *Proceedings of the 3rd international conference on algorithmic decision theory* (Vol. 8176, pp. 71–85). Lecture Notes in Computer Science, Berlin: Springer.
3. Buhrman, H., & Hitchcock, J. (2008, June). 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, CA.
4. Baumeister, D., & Rothe, J. (2016). Preference aggregation by voting. In J. Rothe (Ed.), *Economics and computation: An introduction to algorithmic game theory, computational social choice, and fair division* (pp. 197–325). Berlin: Springer.
5. Bartholdi III, J., Tovey, C., & Trick, M. (1992). How hard is it to control an election? *Mathematical and Computer Modeling*, *16*(8/9), 27–40.
6. Cai, J., Chakaravarthy, V., Hemaspaandra, L., & Ogihara, M. (2005). Competing provers yield improved Karp–Lipton collapse results. *Information and Computation*, *198*(1), 1–23.
7. Chandra, A., Kozen, D., & Stockmeyer, L. (1981). Alternation. *Journal of the ACM*, *26*(1), 114–133.
8. Desmedt, Y., & Elkind, E. (2010, June). Equilibria of plurality voting with abstentions. In *Proceedings of the 11th ACM conference on electronic commerce* (pp. 347–356). ACM Press, New York.
9. Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001, March). Rank aggregation methods for the web. In *Proceedings of the 10th international world wide web conference* (pp. 613–622). ACM Press, New York.
10. Dekel, E., & Piccione, M. (2001). Sequential voting procedures in symmetric binary elections. *Journal of Political Economy*, *108*(1), 34–55.
11. Erdélyi, G., Fellows, M., Rothe, J., & Schend, L. (2015). Control complexity in Bucklin and fallback voting: A theoretical analysis. *Journal of Computer and System Sciences*, *81*(4), 632–660.
12. Erdélyi, G., Nowak, M., & Rothe, J. (2009). Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly*, *55*(4), 425–443.
13. Ephrati, E., & Rosenschein, J. (1997). A heuristic technique for multi-agent planning. *Annals of Mathematics and Artificial Intelligence*, *20*(1–4), 13–67.
14. Fraenkel, A., Garey, M., Johnson, D., Schaefer, T., & Yesha, Y. (1978, October). The complexity of checkers on an NxN board—Preliminary report. In *Proceedings of the 19th IEEE symposium on foundations of computer science* (pp. 55–64). IEEE Computer Society, Los Alamitos, CA.
15. Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2010). Using complexity to protect elections. *Communications of the ACM*, *53*(11), 74–82.
16. Fitzsimmons, Z., Hemaspaandra, E., & Hemaspaandra, L. (2013, August). Control in the presence of manipulators: Cooperative and competitive cases. In *Proceedings of the 23rd international joint conference on artificial intelligence* (pp. 113–119). AAAI Press, Menlo Park, CA.

17. Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. (2014). The complexity of manipulative attacks in nearly single-peaked electorates. *Artificial Intelligence*, *207*, 69–99.
18. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research*, *35*, 275–341.
19. Faliszewski, P., Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). A richer understanding of the complexity of election systems. In Ravi, S., & Shukla, S. (Eds.), *Fundamental problems in computing: Essays in honor of Professor Daniel J. Rosenkrantz* (pp. 375–406). Berlin: Springer.
20. Faliszewski, P., & Rothe, J. (2016). Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, & A. Procaccia (Eds.), *Handbook of computational social choice* (pp. 146–168). Cambridge: Cambridge University Press.
21. Ghosh, S., Mundhe, M., Hernandez, K., & Sen, S. (1999). Voting for movies: The anatomy of recommender systems. In *Proceedings of the 3rd annual conference on autonomous agents* (pp. 434–435). ACM Press, New York.
22. Homan, C., & Hemaspaandra, L. (2009). Guarantees for the success frequency of an algorithm for finding Dodgson-election winners. *Journal of Heuristics*, *15*(4), 403–423.
23. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2007). Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, *171*(5–6), 255–285.
24. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2009). Hybrid elections broaden complexity-theoretic resistance to control. *Mathematical Logic Quarterly*, *55*(4), 397–424.
25. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2012, August). Controlling candidate-sequential elections. In *Proceedings of the 20th European conference on artificial intelligence* (pp. 905–906). IOS Press, Amsterdam.
26. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2012, August). Online voter control in sequential elections. In *Proceedings of the 20th European conference on artificial intelligence* (pp. 396–401). IOS Press, Amsterdam.
27. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2013, January). The complexity of online manipulation of sequential elections. In *Proceedings of the 14th conference on theoretical aspects of rationality and knowledge* (pp. 111–120).
28. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2014). The complexity of online manipulation of sequential elections. *Journal of Computer and System Sciences*, *80*(4), 697–710.
29. Hemaspaandra, E., Hemaspaandra, L., & Rothe, J. (2013). The complexity of manipulative actions in single-peaked societies. In J. Rothe (Ed.), *Economics and computation: An introduction to algorithmic game theory, computational social choice, and fair division* (pp. 327–360). Berlin: Springer.
30. Hopcroft, J., & Ullman, J. (1979). *Introduction to automata theory, languages, and computation*. Reading, MA: Addison-Wesley.
31. Hemaspaandra, L., & Williams, R. (2012). An atypical survey of typical-case heuristic algorithms. *SIGACT News*, *43*(4), 71–89.
32. Lichtenstein, D., & Sipser, M. (1980). GO is polynomial-space hard. *Journal of the ACM*, *27*(2), 393–401.
33. Mattei, N., Goldsmith, J., Klapper, A., & Mundhenk, M. (2015). On the complexity of bribery and manipulation in tournaments with uncertain information. *Journal of Applied Logic*, *13*(4, Part 2), 557–581.
34. McCabe-Dansted, J., Pritchard, G., & Slinko, A. (2008). Approximability of Dodgson's rule. *Social Choice and Welfare*, *31*(2), 311–330.
35. Oren, J., & Lucier, B. (2014, July). Online (budgeted) social choice. In *Proceedings of the 28th AAAI conference on artificial intelligence* (pp. 1456–1462). AAAI Press, Menlo Park, CA.
36. Papadimitriou, C. (1994). *Computational complexity*. Reading, MA: Addison-Wesley.
37. Parkes, D.,& Procaccia, A. (2013, July). Dynamic social choice with evolving preferences. In *Proceedings of the 27th AAAI conference on artificial intelligence* (pp. 767–773). AAAI Press, Menlo Park, CA.
38. Rothe, J., & Schend, L. (2013). Challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey. *Annals of Mathematics and Artificial Intelligence*, *68*(1–3), 161–193.
39. Rossi, F., Venable, K., & Walsh, T. (2011). *A short introduction to preferences: Between artificial intelligence and social choice*., Synthesis Lectures on Artificial Intelligence and Machine Learning San Rafael: Morgan & Claypool Publishers.
40. Simon, H. (1969). *The sciences of the artificial* (3rd ed., 1996). Cambridge: MIT Press.
41. Sloth, B. (1993). The theory of voting and equilibria in noncooperative games. *Games and Economic Behavior*, *5*(1), 152–169.
42. Storer, J. (1983). On the complexity of chess. *Journal of Computer and System Sciences*, *27*(1), 77–100.

43. Tennenholtz, M. (2004, July). Transitive voting. In *Proceedings of the 5th ACM conference on electronic commerce* (pp. 230–231). ACM Press, New York.
44. Xia, L., & Conitzer, V. (2010, July). Stackelberg voting games: Computational aspects and paradoxes. In *Proceedings of the 24th AAAI conference on artificial intelligence* (pp. 697–702). AAAI Press, Menlo Park, CA.
45. Yang, Y., & Dimitrov, D. (2016, June). How hard is it to control a group? Technical Report, arXiv:1606.03366 [cs.GT], Computing Research Repository.