The Complexity of Online Bribery in Sequential Elections (Extended Abstract)

Edith Hemaspaandra*
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623, USA
eh@cs.rit.edu

Lane A. Hemaspaandra†
Department of Computer Science
University of Rochester
Rochester, NY 14627, USA
http://www.cs.rochester.edu/u/lane

Jörg Rothe‡
Institut für Informatik
Heinrich-Heine-Universität Düsseldorf
40225 Düsseldorf, Germany
rothe@hhu.de

Prior work on the complexity of bribery assumes that the bribery happens simultaneously, and that the briber has full knowledge of all voters' votes. But neither of those assumptions always holds. In many real-world settings, votes come in sequentially, and the briber may have a use-it-or-lose-it moment to decide whether to bribe/alter a given vote, and at the time of making that decision, the briber may not know what votes remaining voters are planning on casting.

In this paper, we introduce a model for, and initiate the study of, bribery in such an online, sequential setting. We show that even for election systems whose winner-determination problem is polynomial-time computable, an online, sequential setting may vastly increase the complexity of bribery, in fact jumping the problem up to completeness for high levels of the polynomial hierarchy or even PSPACE. On the other hand, we show that for some natural, important election systems, such a dramatic complexity increase does not occur, and we pinpoint the complexity of their bribery problems in the online, sequential setting.

1 Introduction

In computational social choice theory, the three most studied types of attacks on elections are bribery, control, and manipulation, and the models of those that are studied seek to model the analogous real-world actions.

Such studies are typically carried out for the model in which all the voters vote simultaneously. That sometimes is the case in the real world. But it also is sometimes the case that the voters vote in sequence—in what is sometimes called a roll-call election.

That type of setting has been relatively recently introduced and studied for control and manipulation—in particular, studies have been done of both control and manipulation in the so-called online, sequential setting [HHR14, HHR17a, HHR17b]. In the present paper, we study the complexity of, and algorithms for, the online, sequential case of bribery.

*Supported in part by NSF grant DUE-1819546 and a Renewed Research Stay grant from the Alexander von Humboldt Foundation. Work done in part while on sabbatical visits to ETH-Zürich and the University of Düsseldorf.

†Supported in part by a Renewed Research Stay grant from the Alexander von Humboldt Foundation. Work done in part while on sabbatical visits to ETH-Zürich and the University of Düsseldorf.

‡Supported in part by DFG grant RO 1202/14-2.

L.S. Moss (Ed.): TARK 2019
EPTCS 297, 2019, pp. 233–251, doi:10.4204/EPTCS.297.16

Briefly put, we are studying the case where the voting order (and the voter weights and cost of bribing each voter) is known ahead of time to the briber. But at the moment a voter seeks to vote, the voter's planned vote is revealed to the briber, who then has a use-it-or-lose-it chance to bribe the voter, by paying the voter's bribe-price (and doing so allows that vote to be changed to any vote the briber desires).

The problem we are studying is the complexity of that decision. In particular, how hard is it to decide whether under optimal play on the part of the briber there is an action for the briber towards the current voter such that under continued future optimal play by the briber (in the face of all future revelations of unknown information being pessimal), the briber can reach a certain goal (e.g., having one of his or her two favorite candidates win; or not having any of his or her three most hated candidates win).

What is an example of a situation that might be modeled by sequential bribery? One example would be the case of a new department chair meeting sequentially with each faculty member, on the morning of a noon faculty vote on some crucial issue, and at each meeting quickly finding the faculty member's preferences regarding the issue and assessing how much of a bribe would be needed to change those, and then deciding whether or not to commit that amount from the chair's (limited) discretionary raise pool in order to execute a bribe of that faculty member. (See the final three paragraphs of Section 3 of the technical-report version of this paper [HHR19] for a more detailed discussion of issues regarding the model, the varying forms the costs in bribery can take (from actual dollars to time or effort spent to risk accepted), and the fact that, despite the typical associations with the word "bribery," in many settings bribery is not modeling immoral or evil acts.)

The following list presents the section structure of our results, and mentions some of the novel proof approaches needed to obtain them.

1. Sections 4.1 and 4.2 establish our upper bounds—of PSPACE and the Π^p_{2k+1} level of the polynomial hierarchy—on online bribery (i.e., online, sequential bribery, but we will for the rest of the paper and especially in our problem names often omit the word "sequential" when the word "online" is present) in the general case and in the case of being restricted to at most k bribes.

   The Π^p_{2k+1} upper bounds are far less straightforward than upper bounds in the polynomial hierarchy typically are. Since bribes can occur on any voter (until one runs out of allowed bribes), and so a yes-no decision has to be made, even for the case of at most k bribes, there can be long strings of alternating existential and universal choices in the natural alternating Turing machine programs for the problems. And so there is the threat that one can prove merely a PSPACE upper bound.

   However, in Section 4.2.1, we prove a more general result about alternating Turing machines that, while perhaps having many alternations between existential and universal choices, make most of them in a "boring" way. Basically, we show that in the relevant setting one can pull much of the existential guessing upstream and make it external to the alternating Turing machine, and indeed one can do so in such a way that one transforms the problem into the disjunction of a polynomial number of uniformly generated questions about actions of alternating Turing machines each of which itself has at most $2k+1$ alternation blocks. From that, we establish the needed upper bound, both for the relevant abstract case of alternating Turing machines and for our online bribery problems.

2. Section 4.3 proves that there are election systems with simple winner problem such that each of the abovementioned upper-bounds is tight, i.e., that PSPACE-completeness holds or Π^p_{2k+1}-completeness holds.

   There is a substantial, novel challenge that the proof here has to overcome. Namely, to prove for example Π^p_{2k+1}-hardness, we generally need to reduce from quantified boolean formulas with particular quantifiers applying to particular variables. However, in online bribery, the briber is

allowed to choose where to do the bribing. This in effect corresponds to having a formula with clusters of quantified variables, yet such that we as we attempt to prove theorems related to these structures don't have control over which quantifiers are existential and which are universal. Rather, in effect what the online bribery setting will test is whether there exists an assignment (consistent with the number of bribes allowed—which limits the number of existential quantifiers one can set) of each quantifier to be either existential or universal, such that for that quantifier-assignment the formula evaluates as true. (This is not at all the same as quantifier exchange; in quantifier exchange, the exchanged quantifiers move around together with their associated variables.)

However, we handle this by showing how to construct a new formula that builds in protection against this setting. In particular, we note that one can take a quantified boolean formula and turn it into one such that, in this Wild West setting of quantifier assignment, the new formula can be made true by a legal (i.e., having at most as many $\exists$ quantifiers as the original formula) quantifier assignment exactly if the original formula is true.

3. In Section 5, we look at the complexity of online bribery for various natural systems. We show that for both Plurality and Approval, it holds that priced, weighted online bribery is NP-complete, whereas all other problem variants of online bribery are in P. This also shows that bribery can be harder than online bribery for natural systems. In addition, we provide complete dichotomy theorems that distinguish NP-hard from easy cases for all our online bribery problems for scoring rules and additionally we show that Veto elections, even with three candidates, have even higher lower bounds for weighted online bribery, namely $\mathrm{P}^{\mathrm{NP}[1]}$-hardness.

Due to space limits, many of our proofs and much additional discussion can be found only in the technical-report version of this paper [HHR19]. In particular, only three of our proofs are included in this conference version and at times this version may refer to proofs found in the technical report as if they were available to and known to the reader.

2 Related Work

Our paper's general area is computational social choice, in which studying the complexity of election and preference aggregation problems and manipulative attacks on them is a central theme. There are many excellent surveys and book chapters on computational social choice [BCE13, Rot16, BCE+16], and computational social choice and computational complexity have a long history of close, mutually beneficial interaction (see the survey [Hem18]).

The prior papers most related to our work are the papers that defined and studied the complexity of online control [HHR17a, HHR17b], of online manipulation [HHR14], and of (regular) bribery [FHH09]. Particularly important among those is online manipulation, as we will show connections/inheritance between online manipulation and our model. We also will show connections/inheritance between (regular) manipulation and our model. Regular manipulation was introduced by Bartholdi, Tovey, and Trick [BTT89] in the unweighted case and by Conitzer, Sandholm, and Lang [CSL07] in the weighted case.

The existing work most closely related to our work on the effect of formulas on limits on existential actions is the work on online voter control [HHR17b], though the issues tackled here are different and harder.

The work of Xia and Conitzer [XC10] (see also [DE10, Slo93, DP01]) that defines and explores the Stackelberg voting game is also about sequential voting, although unlike this paper their analysis is

game-theoretic and is about manipulation rather than bribery. Sequential (and related types of) voting have also been studied in an axiomatic way [Ten04] and using Markov decision processes [PP13], though neither of those works focuses on issues of bribery. Poole and Rosenthal [PR97] provide a history of roll-call voting.

Our approach to the briber's goal, which is assuming worst-case revelations of information, is inspired by the approach used in the area known as online algorithms [BE98].

Interesting work that is related—though somewhat distantly—in flavor to our study is the paper of Chevaleyre et al. [CLM$^+$12] on the addition of candidates. They also focus on the moment at which one has to make a key decision, in their case whether all of a group of potential additional candidates should be added.

3 Preliminaries

3.1 Basics

P is the class of decision problems in deterministic polynomial time. NP is the class of decision problems in nondeterministic polynomial time. For each $k \geq 0$, Σ_k^p is the class of decision problems in the kth Σ level of the polynomial hierarchy [MS72, Sto76], e.g., $\Sigma_0^p = \mathrm{P}$, $\Sigma_1^p = \mathrm{NP}$, and $\Sigma_2^p = \mathrm{NP}^{\mathrm{NP}}$ (i.e., the class of sets accepted by nondeterministic polynomial-time oracle Turing machines given unit-cost access to an NP oracle). For each $k \geq 0$, $\Pi_k^p = \{L \mid \overline{L} \in \Sigma_k^p\}$, e.g., $\Pi_0^p = \Sigma_0^p = \mathrm{P}$, $\Pi_1^p = \mathrm{coNP}$, and $\Pi_3^p = \mathrm{coNP}^{\mathrm{NP}^{\mathrm{NP}}}$. Further, $\Delta_2^p = \mathrm{P}^{\mathrm{NP}}$ is the class of decision problems solvable by a deterministic polynomial-time oracle Turing machine with access to an NP oracle, and $\mathrm{P}^{\mathrm{NP}[1]}$ is the same class restricted to one oracle query.

We say that $A \leq_m^p B$ (A polynomial-time many-one reduces to B) exactly if there is a polynomial-time computable function f such that $(\forall x)[x \in A \iff f(x) \in B]$.

Fact 3.1. *For each complexity class $\mathscr{C} \in \{\Sigma_0^p, \Sigma_1^p, \Pi_1^p, \Sigma_2^p, \Pi_2^p, \dots\}$, $\mathscr{C}$ is closed downwards under polynomial-time many-one reductions, i.e., $(B \in \mathscr{C} \wedge A \leq_m^p B) \implies A \in \mathscr{C}$.*

Each of the classes mentioned in Fact 3.1 is even closed downwards under what is known as polynomial-time disjunctive truth-table reducibility [LLS75]. Disjunctive truth-table reducibility can be defined as follows. We say that $A \leq_{dtt}^p B$ (A polynomial-time disjunctive truth-table reduces to B) exactly if there is a polynomial-time computable function f such that, for each x, it holds that (a) $f(x)$ outputs a list of 0 or more strings, and (b) $x \in A$ if and only if at least one string output by $f(x)$ is a member of B. (Polynomial-time many-one reductions are simply the special case of polynomial-time disjunctive truth-table reductions where the polynomial-time disjunctive truth-table reduction's output-list function is required to always contain exactly one element.)

Fact 3.2. *For each complexity class $\mathscr{C} \in \{\Sigma_0^p, \Sigma_1^p, \Pi_1^p, \Sigma_2^p, \Pi_2^p, \dots\}$, $\mathscr{C}$ is closed downwards under polynomial-time disjunctive truth-table reductions, i.e., $(B \in \mathscr{C} \wedge A \leq_{dtt}^p B) \implies A \in \mathscr{C}$.*

The above fact is obvious for P, and is easy to see and well known for NP and coNP (for example, the results follow immediately from the result of Selman [Sel82] that NP is closed downwards under so-called positive Turing reductions). The results for the NP and coNP cases relativize (as Selman's mentioned result's proof clearly relativizes), and that gives (namely, by relativizing the NP and coNP cases by complete sets for NP, $\mathrm{NP}^{\mathrm{NP}}$, etc.) the claims for the higher levels of the polynomial hierarchy (in fact, it gives something even stronger, since it gives downward closure under disjunctive truth-table reductions that themselves are relativized, but we won't need that stronger version in this paper).

Since all the many-one and disjunctive truth-table reductions discussed in the paper will be polynomial-

time ones, we henceforth will sometimes skip the words "polynomial-time" when speaking of a polynomial-time many-one or disjunctive truth-table reduction.

A set L is said to be (polynomial-time many-one) hard for a class $\mathscr{C}$ (for short, "L is $\mathscr{C}$-hard") exactly if $(\forall B \in \mathscr{C})[B \leq_m^p L]$. If in addition $L \in \mathscr{C}$, we say that L is polynomial-time many-one complete for $\mathscr{C}$, or simply that L is $\mathscr{C}$-complete.

An (unweighted) election system $\mathscr{E}$ takes as input a voter collection V and a candidate set C, such that each element of V contains a voter name and a preference order over the candidates in C; and for us in this paper preference orders are always total orders, except when we are speaking of approval voting where the preference orders are bit-vectors from $\{0,1\}^{\|C\|}$. (For the rest of the preliminaries, we'll always speak of total orders as the preference orders' type, with it being implicit that when later in the paper we speak of and prove results about approval voting, all such places will tacitly be viewed as speaking of bit-vectors.) The election system maps from that to a (possibly nonproper) subset of C, often called the winner set. We often will call each element of V a vote, though as is common sometimes we will use the term vote to refer just to the preference order. We often will use the variable names σ, σ_1, σ_2, $\ldots$, σ_i for total orders. We allow election systems to, on some inputs, have no winners.[1]

For a given (unweighted, simultaneous) election system, $\mathscr{E}$, the (unweighted) winner (aka the winner-determination) problem (in the unweighted case) is the set $\{\langle C,V,c\rangle \mid c \in C \wedge\ c \text{ is a winner of the election } (C,V) \text{ under election system } \mathscr{E}\}$.

For a given (unweighted [sic], simultaneous) election system, $\mathscr{E}$, the winner problem in the weighted [sic] case will be the set of all strings $\langle C,V,c\rangle$ such that C is a candidate set, V is a set of weighted (via binary nonnegative integers as weights) votes (each consisting of a voter name and a total order over C), $c \in C$, and in the unweighted election created from this by replacing each w-weighted vote in V with w unweighted copies of that same vote, c is a winner in that election under the (unweighted) election system $\mathscr{E}$. For an election system $\mathscr{E}$, it is clear that if the winner problem in the weighted case is in P, then so is the winner problem in the unweighted case. However, it is not hard to show that there are election systems $\mathscr{E}$ for which the converse fails. The above approach to defining the weighted winner problem is natural and appropriate for the election systems discussed in this paper. (The technical-report version of this paper [HHR19] has a detailed discussion of the strengths and weaknesses of using this approach to the weighted winner problem in other settings, see also the discussion and results in [FH16].)

3.2 Online Bribery in Sequential Elections

This paper is about the study of online bribery in sequential elections. In this setting, we are—this is the sequential part—assuming that the voters vote in a well-known order, sequentially, with each casting a ballot that expresses preferences over all the candidates. And we are assuming—this is the online part—that the attacker, called "the briber," as each new vote comes in has his or her one and only chance to bribe that voter, i.e., to alter that vote to any vote of the briber's choice.

Bribery has aspects of both the other standard types of electoral attacks: bribery is like manipulation in that one changes votes and it is like (voter) control in that one is deciding on a set of voters (in the case of bribery, which ones to bribe). Reflecting this, our model follows as closely as possible the relevant parts of the existing models that study manipulation and control in online settings [HHR14, HHR17b, HHR17a]. In particular, we will follow insofar as possible both the model of, and the notation of the model of, the

[1] Although in social choice this is often disallowed, as has been discussed previously, see, e.g., [FHH16, Footnote 3], artificially excluding the case of no winners is unnatural, and many papers in computational social choice allow this case. A typical real-world motivating example is that in Baseball Hall of Fame votes, having no inductees in a given year is a natural outcome that has at times occurred.

paper by Hemaspaandra, Hemaspaandra, and Rothe [HHR17b] that introduced the study of online voter control in sequential elections. In particular, we will follow the flavor of their model of control by deleting voters, except here the key decision is not whether to delete a given voter, but rather is whether a given voter should be bribed, i.e., whether the voter's vote should be erased and replaced with a vote supplied by the briber. That "replace[ment]" part is more similar to what happens in the study of online manipulation, which was modeled and studied by Hemaspaandra, Hemaspaandra, and Rothe [HHR14]. We will, as both those papers do, focus on a key moment—a moment of decision—and in particular on the complexity of deciding whether there exists an action the briber can take, at that moment, such that doing so will ensure, even under the most hostile of conditions regarding the information that has not yet been revealed, that the briber will be able to meet his or her goal.

If u is a voter and C is a candidate set, an *election snapshot for C and u* is specified by a triple $V = (V_{<u}, u, V_{>u})$, which loosely put is made up of all voters in the order they vote, each accompanied in models where there are prices and/or weights with their prices and/or weights (which in this paper are assumed to be nonnegative integers coded in binary).[2] In addition, for each voter voting before u (namely, the voters in $V_{<u}$), also included in this listing will be the vote they cast (or if they were bribed, what vote was cast for them by the briber) and whether they were bribed; and for u the listing will also include the vote u will cast unless bribed to cast a different vote. So $V_{>u}$ is simply a list, in the order they will vote, of the voters, if any, who come after u, each also including the voter's price and/or weight data if we are in a priced and/or weighted setting. Further, the vote for u and all the votes in $V_{<u}$ must be votes over the candidate set C (and in particular, in this paper votes are total orderings of the candidates, e.g., $a > b > c$).[3]

[2]Why do we feel it natural in most situations for the prices and weights be in binary rather than unary? A TARK referee, for example, asked whether it was not natural to assume that prices and weights would always be small, or if not, would always be multiples of some integer that when divided out would make the remaining numbers small.

Our answer is that both prices and weights in many settings tend to be large, and without any large, shared-by-all divisor. To see this clearly regarding weights, consider for example the number of shares of stock the various stockholders hold in some large corporation or the number of residents in each of the states of a country. Prices too are potentially as rich and varied as are individuals and objects, e.g., in some settings each person's bribe-price might be the exact fair market value of his or her house, or might be closely related to the number of visits a web site they own has had in the past year.

Pulling back, we note that requiring prices and weights to be in unary is often tremendously (and arguably inappropriately) *helping* the algorithms as to what their complexity is, since in effect one is "padding" the many inputs' sizes as much as exponentially. But if weighted votes are viewed as indivisible objects—and that is indeed how they are typically treated in the literature—the right approach indeed is to code the weights in binary, and not to give algorithm designers the potentially vastly lowered bar created by the padding effect of coding the weights in unary. Indeed, it is known in the study of (nonsequential) bribery that changing prices or weights to unary can shift problems' complexities from NP-hardness to being in deterministic polynomial time [FHH09, pp. 500–504].

Also, people typically do code natural numbers in binary, not unary.

[3]There is a slight overloading of notation above, in that we have not explicitly listed in the structure the location of the mentioned extra data. In fact, our actual definition is that the first and last components of the 3-tuple V are lists of tuples, and the middle component is a single tuple. Each of these contain the appropriate information, as mentioned above. For example, for priced, weighted bribery:

1. the elements of the list $V_{<u}$ will be 5-tuples $(v_i, p_i, w_i, \sigma_i, b_i)$ whose components respectively are the voter's name, the voter's price, the voter's weight, the voter's cast ballot (which is the voter's original preference order if the voter was not bribed and is whatever the voter was bribed into casting if the voter was bribed), and a bit specifying whether that vote resulted from being bribed,
2. the middle component of V will be a tuple that contains the first four of those five components, and
3. the elements of the list $V_{>u}$ will contain the first three of the above-mentioned five components.

Similarly, for example, for unpriced, unweighted bribery, the three tuple types would respectively have three components, two components, and one component.

As a remaining tidbit of notational overloading, in some places we will speak of u when we in fact mean the voter name that is

Let us, with the above in hand, define our notions of online bribery for sequential elections. Settings can independently allow or not allow prices and weights, and so we have four basic types of bribery in our online, sequential model, each having both constructive and destructive versions.

In the original paper on nonsequential bribery there were other types of bribery, e.g., microbribery, unary-coding, and succinct variants [FHH09]. Many other types have been studied since, e.g., nonuniform bribery [Fal08] and swap- (and its special case shift-) bribery [EFS09] (see also [EF10, BCF$^+$16]). However, for compactness and since they are very natural, this paper focuses completely on (standard) bribery in its eight typical versions (as to prices, weights, and constructive/destructive), except now in an online, sequential setting.

Our specification of these problems as languages is centered around what Hemaspaandra, Hemaspaandra, and Rothe [HHR14] called a *magnifying-glass moment*. This is a moment of decision as to a particular voter. To capture precisely what information the briber does and does not have at that moment, and to thus allow us to define our problems, we define a structure that we will call an *OBS*, which stands for *online bribery setting*. An OBS is defined as a 5-tuple (C, V, σ, d, k), where C is a set of candidates; $V = (V_{<u}, u, V_{u<})$ is an election snapshot for C and u as discussed earlier; σ is the preference order of the briber; $d \in C$ is a distinguished candidate; and k is a nonnegative integer (representing for unpriced cases the maximum number of voters that can be bribed, and for priced cases the maximum total cost, i.e., the sum of the prices of all the bribed voters).

Given an election system $\mathscr{E}$, we define the *online unpriced, unweighted bribery problem*, abbreviated by online-$\mathscr{E}$-Bribery, as the following decision problem. The input is an OBS. And the question is: Does there exist a legal (i.e., not violating whatever bribe limit holds) choice by the briber on whether to bribe u (recall that u is specified in the OBS, namely, via the middle component of V) and, if the choice is to bribe, of what vote to bribe u into casting, such that if the briber makes that choice then no matter what votes the remaining voters after u are (later) revealed to have, the briber's goal (the meeting of which itself depends on $\mathscr{E}$ and will be defined explicitly two paragraphs from now) can be reached by the current decision regarding u and by using the briber's future (legal-only, of course) decisions (if any), each being made using the briber's then-in-hand knowledge about what votes have been cast by then?

Note that this approach is about alternating quantifiers. It is asking whether there is a current choice by the briber such that for all potential revealed vote values for the next voter there exists a choice by the briber such that for all potential revealed vote values for the next-still voter there exists a choice by the briber such that… and so on… such that the resulting winner set under election system $\mathscr{E}$ meets the briber's goal. This is a bit more subtle than it might at first seem. The briber is acting very powerfully, since the briber is represented by existential quantifiers. But the briber is not all-powerful in this model. In particular, the briber can't see and act on future revelations of vote values; after all, those are handled by a universal quantifier that occurs downstream from an existential quantifier that commits the briber to a particular choice.

In the above we have not defined what "the briber's goal" is, so let us do that now. $W_{\mathscr{E}}(C, U)$ will denote the winner set, according to election system $\mathscr{E}$, of the (standard, nonsequential) election (C, U), where C is the candidate set and U is the set of votes. By *the briber's goal* we mean, in the constructive case, that if at the end of the above process U' is the set of votes (some may be the original ones and some may be the result of bribes), it holds that $W_{\mathscr{E}}(C, U') \cap \{c \mid c \geq_\sigma d\} \neq \emptyset$, i.e., the winner set includes some candidate (possibly itself being d) that the briber likes at least as much as the briber likes d. In the destructive case, the goal is to ensure that no candidate that the briber hates as much or more than the briber

the first component of the tuple that makes up the middle tuple of V. That is, we will use u both for a tuple that names u and gives some of its properties, and as a stand-in for the voter him- or herself. Which use we mean will always be clear from context.

hates d belongs to the winner set, i.e., the briber's goal is to ensure that $W_{\mathscr{E}}(C,U') \cap \{c \mid d \geq_\sigma c\} = \emptyset$.

Above, we have defined both online-$\mathscr{E}$-Bribery and online-$\mathscr{E}$-Destructive-Bribery. Those both are in the unpriced, unweighted setting. And so as per our definitions, the voters passed in as part of the problem statement do not come with or need price or weight information.

But our definitions note that for the priced and/or weighted settings, the OBS will carry the prices and/or weights. And so the same definition text that was used above defines all the other cases, except that one must keep in mind for the priced cases that when the "bribery limit" is mentioned one must instead speak of the "bribery budget," and in the weighted cases the winner set W is of course defined in terms of the weighted version of the given voting system (which must, for that to be meaningful, have a well-defined notion of what its weighted version is; Section 3.1 provides that notion for all systems in this paper). Thus, we also have tacitly defined the six problems online-$\mathscr{E}$-\$Bribery, online-$\mathscr{E}$-Destructive-\$Bribery, online-$\mathscr{E}$-Weighted-Bribery, online-$\mathscr{E}$-Destructive-Weighted-Bribery, online-$\mathscr{E}$-Weighted-\$Bribery, and online-$\mathscr{E}$-Destructive-Weighted-\$Bribery.

For an unpriced online bribery problem, we will postpend the problem name with a "[k]" to define the version where as part of the problem definition itself the bribery limit is—in contrast with the above unpriced problem—not part of the input but rather is fixed to be the value k. For example, online-$\mathscr{E}$-Bribery[k] denotes the unpriced, unweighted bribery problem where the number of voters who can be bribed is set not by the problem input but rather is limited to be at most k. Note that in each of the "[k]" variants, we tacitly are altering the definition of OBS from its standard 5-tuple, (C,V,σ,d,k), to instead the 4-tuple (C,V,σ,d); that is because for these cases, the k is fixed as part of the general problem itself, rather than being a variable part of the individual instances. For priced "[k]" variants, there will be both a limit (being a variable part of the input) on the total price of the bribes and a fixed as part of the general problem itself limit on the number of voters who can be bribed.

Of course, there are some immediate relationships that hold between these eight problems. One has to be slightly careful since there is a technical hitch here. We cannot for example simply claim that online-$\mathscr{E}$-Bribery[k] is a subcase of online-$\mathscr{E}$-\$Bribery[k]. If we had implemented the unpriced case by still including prices in the input but requiring them all to be 1, then it would be a subcase. But regarding both prices (weights), our definitions simply omit them completely from problems that are not about prices (weights). In spirit, it is a subcase, but formally it is not. Nonetheless, we can still reflect the relationship between these problems, namely, by stating how they are related via polynomial-time many-one reductions. (We could even make claims regarding more restrictive reduction types, but since this paper is concerned with complexity classes that are closed downwards under polynomial-time many-one reductions, there is no reason to do so.) The following proposition (and the connections that follow from it by the transitivity of polynomial-time many-one reductions) captures this.

Proposition 3.3. *1. For each $k > 0$ and for each election system $\mathscr{E}$,*

- online-$\mathscr{E}$-Bribery[k] $\leq_m^p$ online-$\mathscr{E}$-\$Bribery[k],
- online-$\mathscr{E}$-Bribery[k] $\leq_m^p$ online-$\mathscr{E}$-Weighted-Bribery[k],
- online-$\mathscr{E}$-\$Bribery[k] $\leq_m^p$ online-$\mathscr{E}$-Weighted-\$Bribery[k], *and*
- online-$\mathscr{E}$-Weighted-Bribery[k] $\leq_m^p$ online-$\mathscr{E}$-Weighted-\$Bribery[k].

2. The above item also holds for the case when all of its problems are changed to their destructive versions.

3. The above two items also hold for the case when all the "[k]"s are removed (e.g., we have online-$\mathscr{E}$-Bribery $\leq_m^p$ online-$\mathscr{E}$-\$Bribery*).*

As mentioned in the introduction, the technical-report version of this paper [HHR19] includes some additional discussion of the model, of the flexibility of bribery, and of the fact that bribery is not necessarily always modeling immoral/evil acts.

4 General Upper Bounds and Matching Lower Bounds

Even for election systems with simple winner problems, the best general upper bounds that we can prove for our problems reflect an extremely high level of complexity.

One might wonder whether that merely is a weakness in our upper-bound proofs. However, in each case, we provide a matching completeness result proving that these really are the hardest problems in the classes their upper bounds put them in.

However, in Section 5, we will see that for many specific natural, important systems, the complexity is tremendously lower than the upper bounds, despite the fact that the present section shows that there exist systems that meet the upper bounds.

4.1 The General Upper Bound, Without Limits on the Number of Bribes

This section covers upper bounds for the case when any bribe limit/bribery budget is passed in through the input—not hardwired into the problem itself.

Theorem 4.1. *1. For each election system $\mathscr{E}$ whose winner problem in the unweighted case is in polynomial time (or even in polynomial space), each of the problems* online-$\mathscr{E}$-Bribery*,* online-$\mathscr{E}$-Destructive-Bribery*,* online-$\mathscr{E}$-$Bribery*, and* online-$\mathscr{E}$-Destructive-$Bribery *is in* PSPACE*.*

2. For each election system $\mathscr{E}$ whose winner problem in the weighted case is in polynomial time (or even in polynomial space), each of the problems online-$\mathscr{E}$-Weighted-Bribery*,* online-$\mathscr{E}$-Destructive-Weighted-Bribery*,* online-$\mathscr{E}$-Weighted-$Bribery*, and* online-$\mathscr{E}$-Destructive-Weighted-$Bribery *is in* PSPACE*.*

The proof is made available through the technical-report version of this paper [HHR19].

4.2 The General Upper Bound, With Limits on the Number of Bribes

Turning to the case where in the problem the number of bribes has a fixed bound of k, these problems fall into the Π^p_{2k+1} level of the polynomial hierarchy. That is not immediately obvious. After all, even when one can bribe at most k times, one still for each of the current and future voters seems to need to explore the one-bit-per-voter decision of whether to bribe the voters (plus in those cases where one does decide to bribe, one potentially has to explore the exponential—in the number of candidates—possible votes to which the voter can be bribed). On its surface, for our problems, that would seem to say that the number of alternations between universal and existential moves that the natural polynomial-time alternating Turing program for our problem would have to make is about the number of voters—a bound that would not leave the problem in any fixed level of the polynomial hierarchy, but would merely seem to put the problem in PSPACE.

So these problems are cases where even obtaining the stated upper bound is interesting and requires a twist to prove. The twist is as follows. On the surface the exploration of these problems has an unbounded number of alternations between universal and existential states in the natural, brute-force alternating Turing machine program. But for all but k of the existential guesses on each accepting path, the guess is a boring one, namely, we guess that regarding that voter we don't bribe. We will show, by proving a

more general result about alternating Turing machines and restrictions on the structure of their maximal existential move segments along accepting paths, a Π^p_{2k+1} upper bound on our sets of interest. In some sense, in terms of being charged as to levels of the polynomial hierarchy, we will be showing that if for a certain collection of 0-or-1 existential decisions one on each accepting path chooses 0 all but a fixed number of times (although for the other times one may then make many more nondeterministic choices), one can manage to in effect not be charged at all for the guessing acts that guessed 0.

We know of only one result in the literature that is anything like this. That result, which also came up in the complexity of online attacks on elections, is a result of Hemaspaandra, Hemaspaandra, and Rothe [HHR17b], where in the context not of bribery but of voter control they showed that for each fixed $k > 0$ it holds that, for each polynomial-time alternating Turing machine M whose alternation blocks are each one bit long and that for at most k of the existential blocks guess a zero, the language accepted by M is in coNP.

In contrast, in the present paper's case we are in a far more complicated situation, since in bribery our existential blocks are burdened not just by 1-bit bribe-or-not decisions, but for the cases when we decide to try bribing, we need to existentially guess what bribe to do. And so we do not stay in coNP regardless of how large k is, as held in that earlier case. But we show that we can at least limit the growth to at most $2k+1$ alternating quantifiers—in particular, to the class Π^p_{2k+1}. And since we later provide problems of this sort that are complete for Π^p_{2k+1}, our $2k+1$ is optimal unless the polynomial hierarchy collapses.

We will approach this in two steps. First, as Section 4.2.1, we will prove the result about alternating Turing machines. And then, as Section 4.2.2, we will apply that to online bribery in the case of only globally fixed numbers of bribers being allowed.

4.2.1 A Result about Alternating Turing Machines

Briefly put, an alternating Turing machine [CKS81] (aka ATM) is a generalization of nondeterministic and conondeterministic computation. We will now briefly review the basics (see [CKS81] for a more complete treatment). An ATM can make both universal and existential choices. For a universal "node" of the machine's action to evaluate to true, all its child nodes (one each for each of its possible choices) must evaluate to true. For an existential "node" of the machine's action to evaluate to true, at least one of its child nodes (it has one child node for each of its possible choices) must evaluate to true. A leaf of the computation tree (a path, at its end) is said to evaluate to true if the path halted in an accepting state and is said to evaluate to false if the path halted in a nonaccepting state. (As our machines are time-bounded, all paths halt.) Without loss of generality, in this paper we assume that each universal or existential node has either two children (namely, does a universal or existential split over the choices 0 and 1; we will often call this a "1-bit move") or has exactly one child (it does a trivial/degenerate universal or existential choice of an element from the one-element set $\{0\}$; we will often call this a "0-bit move"). The latter case is in effect a deterministic move, except allowing degenerate $\forall$ steps of that sort will let us put a "separator" between otherwise contiguous $\exists$ computation segments. Of course, long existential guesses can be done in this model, for example by guessing a number of bits sequentially. An ATM accepts or rejects based on what its root node evaluates to (which is determined inductively in the way described above).

Definition 4.2. *Consider a path ρ in the tree of an ATM. The* weight *of that path is as follows. Consider all maximal segments of existential nodes along the path. (As mentioned above, we may without loss of generality, and do, assume that each nonleaf node is $\exists$ or $\forall$, although perhaps a degenerate such node in the way mentioned above). The weight of path ρ is its number of maximal existential segments such that the concatenation of the bits guessed in that segment is not the 1-bit string 0.*

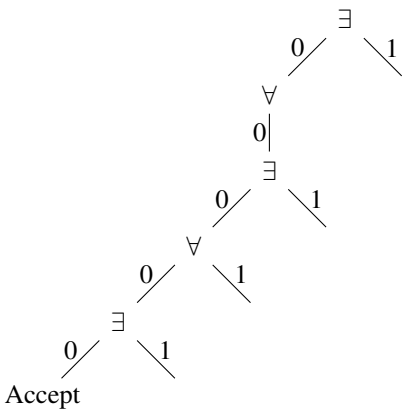

Figure 1: A weight 0 path in the tree of an ATM.

Figure 1 gives an illustration of this. In the figure, the illustrated path (the leftmost one at the left edge of the tree) has weight 0; it has three maximal existential segments, but each is of length one and makes the guess 0.

With this definition in hand, we can now state our key theorem showing that limited weight for ATMs simplifies the complexity of the languages accepted. The result is one where one may go back and forth between thinking it is obvious and thinking it is not obvious. In particular, note that even on accepting paths, which must be of weight at most k, it is completely possible that the number of alternations between existential and universal nodes may be far greater than k and may be far greater than $2k+1$, and indeed may grow unboundedly as the input's size increases. What the theorem below is saying is that despite that, machines with bounded weight on their accepting paths still accept only Π^p_{2k+1} sets.

Theorem 4.3. *Let $k \geq 0$ be fixed. Each polynomial-time ATM M such that on no input does M have an accepting path of weight strictly greater than k accepts a language in Π^p_{2k+1}.*

Proof. Let $k \geq 0$ be fixed. Let L be the language accepted by polynomial-time ATM M that has the property that each of its accepting paths has weight at most k. Our goal is to prove that $L \in \Pi^p_{2k+1}$.

We will do so by proving that there is a set $G \in \Pi^p_{2k+1}$ such that $L \leq^p_{dtt} G$, i.e., L polynomial-time disjunctively truth-table reduces to G. By Fact 3.2, it follows that $L \in \Pi^p_{2k+1}$.

Let q be a nondecreasing polynomial that upper-bounds the running time of M. Let $\langle \cdot, \cdot \rangle$ be a standard pairing function, i.e., a polynomial-time computable, polynomial-time invertible bijection between $\Sigma^* \times \Sigma^*$ and Σ^*. Recall that every step of our ATM M involves either a 0-bit existential move (which we'll think of basically as existentially choosing 0 from the choice palette set $\{0\}$) or a 1-bit existential move (which, recall, involves choosing one element from the choice palette set $\{0,1\}$ with the machine enforcing an "or" over the two children thus reached) or a 0-bit universal move (which we'll think of basically as universally choosing 0 from the choice palette set $\{0\}$) or a 1-bit universal move. And the 1-bit moves involve successor states hinged on whether the move-choice is a 0 or a 1. (As Turing machines are standardly defined, there can be (one or multiple) successor states to a given state, hinged on a (degenerate or nondegenerate) choice.)

G will be the set of all $\langle x, s \rangle$ such that all of the list of conditions that we will give below hold relative to x and s. The intuition here is that s is a bit-vector whose ith bit controls how the ith maximal existential segment is handled: If that bit is a 1, the segment moves forward unrestrained, but if that bit is a 0, then we expect and require (and cut off that part of the tree otherwise) the maximal existential segment to be a

single existential step (either guessing a bit from $\{0,1\}$ or the allowed but superfluous existential step of guessing a bit from the one-element set $\{0\}$) and we basically cut that step out of the tree by replacing it by a trivially universal step. Returning to our defining of G, the set G will be all $\langle x,s\rangle$ such that all the following claims hold.

1. $x \in \Sigma^*$ and $s \in \{0,1\}^{q(|x|)}$.
2. The number of "1"s in the bit-string s is at most k.
3. M accepts when we simulate it on input x but with the following changes in the machine's action.

   As one simulates M on a given path, consider the first existential node (if any) that one encounters.

   If the first bit of s is a 1, then for that node we will directly simulate it, and on all paths that follow from this one, on all the following existential nodes (if any) that are in an unbroken segment of existential nodes from this one, we will similarly directly simulate them. On the other hand, if the first bit of s is a 0, then (a) if it is the case that if the current node makes the choice 0 then the node that follows it is an existential node, then the current path halts and rejects (because something that s is specifying as being a maximal existential segment consisting of a single 0 clearly is not); and (b) if (a) does not hold (and so the node that follows if we make the choice 0 is either universal or a leaf), then do not take an existential action at the current node but rather implement it as a degenerate universal step (namely, a "$\forall$" guess over one option, namely, 0, matching as to next state and so on whatever the existential node would have done on the choice of 0).

   If the path we are simulating didn't already end or get cut off during the above-described handling of its first, if any, maximal existential segment, then continue on until we hit the start of its second existential segment. We handle that exactly as described above, except now our actions are controlled not by the first bit of s but by the second.

   And similarly for the third maximal existential segment, the fourth, and so on.

   All other aspects of this simulation are unchanged from M's own native behavior.

Note that $G \in \Pi^p_{2k+1}$. Why? Even in the worst of cases for us, the computation of M starts with a $\forall$ block and then has k $\exists$ blocks each separated by a $\forall$ block; and then we finish with a $\forall$ block. But then our ATM as it does that simulation starts with $\forall$ and has $2k$ alternations of quantifier type, and thus has $2k+1$ alternation blocks with $\forall$ as the leading one. And so by Chandra, Kozen, and Stockmeyer's [CKS81] characterization of the languages accepted by ATMs with that leading quantifier and that number of alternations, this set is in Π^p_{2k+1}.

Finally, we argue that $L \leq^p_{dtt} G$. In particular, note that $L \leq^p_{dtt} G$ via the reduction that on input x generates every length $q(|x|)$ bit-string having less than or equal to k occurrences of the bit 1, and as its $\leq^p_{dtt}$ output list outputs each of those paired with x. This list is easily generated and is polynomial in size. In particular, the number of pairs in the list is clearly at most $\sum_{0\leq j\leq k}\binom{q(|x|)}{j} \leq \sum_{0\leq j\leq k}(q(|x|))^j \leq (k+1)(q(|x|))^k$. That completes the proof. ❑

In the above, we focused on maximal existential guess sequences, and limiting the number of those, on accepting paths, whose bit-sequence-guessed was other than the string 0. So we barred from accepting paths any maximal existential guess sequences that contain a 1 and any that have two or more bits. We mention in passing that we could have framed things more generally in various ways. For example, we could have made each maximal existential guess be of a fixed polynomial length and could have defined our notion of a "boring" guess sequence not as the string "0" but as a string of 0's of exactly that length. The Π^p_{2k+1} upper bound holds also in that setting, via only slight modifications to the proof.

4.2.2 The Upper-Bound Results Obtained by Applying the Previous Section's Result about Alternating Turing Machines

Theorem 4.4. *1. For each $k \in \{0,1,2,\dots\}$, and for each election system $\mathscr{E}$ whose winner problem in the unweighted case is in polynomial time,*[4] *each of the problems* online-$\mathscr{E}$-Bribery$[k]$, online-$\mathscr{E}$-Destructive-Bribery$[k]$, online-$\mathscr{E}$-\$Bribery$[k]$, *and* online-$\mathscr{E}$-Destructive-\$Bribery$[k]$ *is in* Π^p_{2k+1}.

2. For each $k \in \{0,1,2,\dots\}$, and for each election system $\mathscr{E}$ whose winner problem in the weighted case is in polynomial time,[5] *each of the problems* online-$\mathscr{E}$-Weighted-Bribery$[k]$, online-$\mathscr{E}$-Destructive-Weighted-Bribery$[k]$, online-$\mathscr{E}$-Weighted-\$Bribery$[k]$, *and* online-$\mathscr{E}$-Destructive-Weighted-\$Bribery$[k]$ *is in* Π^p_{2k+1}.

Proof. Let $k > 0$ be fixed. Let us start by arguing that online-$\mathscr{E}$-Weighted-\$Bribery$[k] \in \Pi^p_{2k+1}$.

As noted (for the case without the bound of k) in Section 3.2, what is really going on here is about alternating quantifiers. Consider a given input to the problem online-$\mathscr{E}$-Weighted-\$Bribery$[k]$. Let the voter under consideration (i.e., u) in the focus moment of that problem just for this paragraph be referred to as u_1, and let the ones coming after it be called, in the order they occur, u_2, u_3, …, u_ℓ. What the membership problem is in essence asking is whether there *exists* an allowable (within both the price budget and the global limit of k allowed bribes) choice as to whether to bribe u_1 (and if the decision is to bribe, then whether there *exists* a vote to which to bribe u_1) such that, *for each* vote that u_2 may then be revealed to have, there *exists* an allowable (within both the price budget and the global limit of k allowed bribes) choice as to whether to bribe u_2 (and if the decision is to bribe, then whether there *exists* a vote to which to bribe u_2) such that, *for each* vote that u_3 may then be revealed to have, …, such that there *exists* an allowable (within both the price budget and the global limit of k allowed bribes) choice as to whether to bribe u_ℓ (and if the decision is to bribe, then whether there *exists* a vote to which to bribe u_ℓ) such that $W_{\mathscr{E}}(C,U') \cap \{c \mid c \geq_\sigma d\} \neq \emptyset$ (recall that that inequality says that the winner set includes some candidate that the briber likes at least as much as the briber likes d; U' is here representing the vote set after all the voting/bribing, as per Section 3.2's definitions).

Note that for at most k of the choice blocks associated with $u_1, u_2, \dots$ can we make the choice to bribe. (In fact, if we have already done bribing of one or more voters in $V_{<u}$, then our remaining number of allowed bribes will be less than k.) Keeping that in mind, imagine implementing the above paragraph's alternating-quantifier-based algorithm on a polynomial-time ATM. In our model, every step is either a universal or an existential one, and let us program up all deterministic computations that are part of the above via degenerate universal steps. (We do that rather than using degenerate existential steps since those degenerate existential steps would interact fatally with our definition of maximal existential sequence; we really need those places where one guesses that one will not bribe to be captured as a maximal existential sequence of length one with guess bit 0; this comment is quietly using the fact that when making a 1-bit choice as to whether to bribe we associate the choice 1 with "yes bribe this voter" and 0 with "do not bribe this voter.") In light of that and the fact that we know that the weighted winner problem of election system $\mathscr{E}$ is in P, the limit of k ensures that no accepting path will have weight greater than k. And so by Theorem 4.3, we have that online-$\mathscr{E}$-Weighted-\$Bribery$[k] \in \Pi^p_{2k+1}$.

[4] Unlike Theorem 4.1, we cannot allow the winner problem here to be in PSPACE and argue that the rest of the theorem holds unchanged. However, we can allow the winner problem here to even be in NP $\cap$ coNP, and then the rest of the theorem holds unchanged. The key point to notice to see that that holds is—as follows immediately from the fact that $\mathrm{NP}^{\mathrm{NP}\cap\mathrm{coNP}} = \mathrm{NP}$ [Sch83]—that for each $k \geq 0$, NP $\cap$ coNP is Π^p_{2k+1}-low, i.e., that $\left(\Pi^p_{2k+1}\right)^{\mathrm{NP}\cap\mathrm{coNP}} = \Pi^p_{2k+1}$.

[5] As in the case of the immediately preceding footnote, the rest of the theorem remains unchanged even if we relax the "polynomial time" to instead be "NP $\cap$ coNP."

By the exact same argument, except changing the test at the end to $W_{\mathscr{E}}(C,U') \cap \{c \mid d \geq_\sigma c\} = \emptyset$, we have that online-$\mathscr{E}$-Destructive-Weighted-$Bribery$[k] \in \Pi^p_{2k+1}$.

From these two results, it follows by Proposition 3.3 that online-$\mathscr{E}$-Weighted-Bribery$[k] \in \Pi^p_{2k+1}$ and online-$\mathscr{E}$-Destructive-Weighted-Bribery$[k] \in \Pi^p_{2k+1}$.

That completes the proof of part 2 of the theorem. Now, we cannot simply invoke Proposition 3.3 to claim that part 1 holds. The reason is that part 1's hypothesis about the winner problem merely puts the unweighted winner problem in P, but the proof we just gave of part 2 used the fact that for that part we could assume that the weighted winner problem is in P.

However, the entire construction of this proof works perfectly well in the unweighted case, namely, we are only given that the unweighted winner problem is in P, but the four problems we are studying are the four unweighted problems of part 1 of the theorem statement. So we have that each of the problems online-$\mathscr{E}$-Bribery$[k]$, online-$\mathscr{E}$-Destructive-Bribery$[k]$, online-$\mathscr{E}$-$Bribery$[k]$, and online-$\mathscr{E}$-Destructive-$Bribery$[k]$ is in Π^p_{2k+1}. That completes the proof of the theorem. ❑

4.3 Matching Lower Bounds

For each of the PSPACE and Π^p_{2k+1} upper bounds established so far in this section, we can in fact establish a matching lower bound. We show that by, for each, proving that there is an election system, with a polynomial-time winner problem, such that the given problem is polynomial-time many-one hard for the relevant class (and so, in light of the upper-bound results, is polynomial-time many-one complete for the relevant class).

Theorem 4.5. *1. For each problem I from this list of problems:* online-$\mathscr{E}$-Bribery*,* online-$\mathscr{E}$-Destructive-Bribery*,* online-$\mathscr{E}$-$Bribery*,* online-$\mathscr{E}$-Destructive-$Bribery*,* online-$\mathscr{E}$-Weighted-Bribery*,* online-$\mathscr{E}$-Destructive-Weighted-Bribery*,* online-$\mathscr{E}$-Weighted-$Bribery*, and* online-$\mathscr{E}$-Destructive-Weighted-$Bribery*, there exists an (unweighted) election system $\mathscr{E}$, whose winner problem in both the unweighted case and the weighted case is in polynomial time, such that I is* PSPACE*-complete.*

2. For each $k \in \{0,1,2,\ldots\}$, and for each problem I from this list of problems: online-$\mathscr{E}$-Bribery$[k]$*,* online-$\mathscr{E}$-Destructive-Bribery$[k]$*,* online-$\mathscr{E}$-$Bribery$[k]$*,* online-$\mathscr{E}$-Destructive-$Bribery$[k]$*,* online-$\mathscr{E}$-Weighted-Bribery$[k]$*,* online-$\mathscr{E}$-Destructive-Weighted-Bribery$[k]$*,* online-$\mathscr{E}$-Weighted-$Bribery$[k]$*, and* online-$\mathscr{E}$-Destructive-Weighted-$Bribery$[k]$*, there exists an (unweighted) election system $\mathscr{E}$, whose winner problem in both the unweighted case and the weighted case is in polynomial time, such that I is Π^p_{2k+1}-complete.*

This is the most involved and interesting proof in the paper, but it also is quite long. So due to space the proof—and how to extend the proof to ensure that all the constructed election systems are simultaneously (candidate-)neutral and (voter-)anonymous—is made available through the technical-report version of this paper [HHR19].

5 Online Bribery for Specific Systems

In this section, we look at the complexity of online bribery for various natural systems. For both Plurality and Approval, we show that priced, weighted online bribery is NP-complete but that the election system's other three online bribery variants are in P. This also shows that bribery can be harder than online bribery for natural systems. In addition, we provide complete dichotomy theorems that distinguish NP-hard from easy cases for all our online bribery problems for scoring rules and additionally we show that Veto

elections, even with three candidates, have even higher lower bounds for weighted online bribery, namely $P^{NP[1]}$-hardness.

The following theorem is useful for proving lower bounds for online bribery for specific systems.

Theorem 5.1. *1. ("Regular") manipulation reduces to corresponding online bribery. (So,* $\mathscr{E}$-UCM *reduces to* online-$\mathscr{E}$-Bribery*,* $\mathscr{E}$-DUCM *reduces to* online-$\mathscr{E}$-Destructive-Bribery*,* $\mathscr{E}$-WCM *reduces to* online-$\mathscr{E}$-Weighted-Bribery*, and* $\mathscr{E}$-DWCM *reduces to* online-$\mathscr{E}$-Destructive-Weighted-Bribery.*)*

2. *Constructive manipulation in the unique winner model reduces to corresponding online destructive bribery. (So,* $\mathscr{E}$-UCM *in the unique winner model reduces to* online-$\mathscr{E}$-Destructive-Bribery *and* $\mathscr{E}$-WCM *in the unique winner model reduces to* online-$\mathscr{E}$-Destructive-Weighted-Bribery.*)*

3. *Online manipulation reduces to corresponding online priced bribery. (So,* online-$\mathscr{E}$-UCM *reduces to* online-$\mathscr{E}$-$Bribery*,* online-$\mathscr{E}$-DUCM *reduces to* online-$\mathscr{E}$-Destructive-$Bribery*,* online-$\mathscr{E}$-WCM *reduces to* online-$\mathscr{E}$-Weighted-$Bribery*, and* online-$\mathscr{E}$-DWCM *reduces to* online-$\mathscr{E}$-Destructive-Weighted-$Bribery.*)*

The proof is made available through the technical-report version of this paper [HHR19].

It is interesting to note that, assuming $P \neq NP$, bribery does not reduce to corresponding online bribery, not even for natural systems. For example, Approval-Bribery is NP-complete [FHH09, Theorem 4.2], but we will show below in Theorem 5.7 that online-Approval-Bribery (and even online-Approval-Weighted-Bribery and online-Approval-$Bribery) are in P.

We end this section with a simple observation about unpriced, unweighted online bribery.

Observation 5.2. *For unpriced, unweighted online bribery, it is always optimal to bribe the last k voters (we don't even have to handle u in a special way). This implies that unpriced, unweighted online bribery is certainly reducible to unweighted online manipulation, and so we inherit those upper bounds.*

5.1 Plurality

In this section, we completely classify the complexity of all our versions of online bribery for the most important natural system, Plurality. In this system, each candidate scores a point when it is ranked first in a vote and the candidates with the most points are the winners. We show that these problems are NP-complete if we have both prices and weights, and in P in all other cases.

In the constructive case we call all members of $\{c \mid c \geq_\sigma d\}$ and in the destructive case we call all members of $\{c \mid c >_\sigma d\}$ *desired* candidates, where σ is the briber's ideal ranking and d the designated candidate. The following observation is crucial in our upper bound proofs: For online-Plurality-Weighted-$Bribery and online-Plurality-Destructive-Weighted-$Bribery, there is a successful bribery if and only if there is a successful bribery where all bribed voters from u onward vote for the same highest-scoring desired candidate and all nonbribed voters after u vote for the same highest-scoring undesired candidate. If u is bribed, we do not count u's original vote to compute the highest score. If u is not bribed, then we count u's vote.

Theorem 5.3. online-Plurality-Bribery*,* online-Plurality-Destructive-Bribery*,* online-Plurality-Weighted-Bribery*,* online-Plurality-Destructive-Weighted-Bribery*,* online-Plurality-$Bribery*, and* online-Plurality-Destructive-$Bribery *are in* P.

Theorem 5.4. online-Plurality-Weighted-$Bribery *and* online-Plurality-Destructive-Weighted-$Bribery *are* NP*-complete, even when restricted to two candidates.*

The proofs of the above two theorems are made available through the technical-report version of this paper [HHR19].

5.2 Beyond Plurality

A scoring rule for m candidates is a vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ of integers $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m \geq 0$. This defines an election system on m candidates where each candidate earns α_i points for each vote that ranks it in the ith position and the candidates with the most points are the winners.

Theorem 5.5. *For each scoring vector $\alpha = (\alpha_1, \ldots, \alpha_m)$,*

1. online-α-Weighted-$Bribery *and* online-$\alpha$-Destructive-Weighted-$Bribery *are in* P *if* $\alpha_1 = \alpha_m$ *and* NP*-hard otherwise;*
2. online-α-Weighted-Bribery *and* online-α-Destructive-Weighted-Bribery *are in* P *if* $\alpha_2 = \alpha_m$ *and* NP*-hard otherwise; and*
3. online-α-Bribery, online-α-Destructive-Bribery, online-α-$Bribery, *and* online-$\alpha$-Destructive-$Bribery *are in* P.

The proof is made available through the technical-report version of this paper [HHR19].

In Veto, each candidate scores a point when it is not ranked last in a vote and the candidates with the most points are the winners. Now let's look at 3-candidate-Veto.

Theorem 5.6. *1.* online-3-candidate-Veto-Bribery, online-3-candidate-Veto-Destructive-Bribery, online-3-candidate-Veto-$Bribery, *and* online-3-candidate-Veto-Destructive-$Bribery *are in* P.

2. online-3-candidate-Veto-Weighted-Bribery *and* online-3-candidate-Veto-Destructive-Weighted-Bribery *are* $\mathrm{P}^{\mathrm{NP}[1]}$*-complete.*
3. online-3-candidate-Veto-Weighted-$Bribery *and* online-3-candidate-Veto-Destructive-Weighted-$Bribery *are* $\mathrm{P}^{\mathrm{NP}[1]}$*-hard and in* Δ_2^p *(and we conjecture that they are* Δ_2^p*-complete).*

Proof. We look at different cases for the placement of the designated candidate in the preference order $a >_\sigma b >_\sigma c$. (Note that the polynomial-time cases also follow from Theorem 5.5.)

1. If $d = c$ in the constructive case or $d = a$ in the destructive case, the problem is trivial.
2. If $d = a$ in the constructive case, we need to ensure that a is a winner, and if $d = b$ in the destructive case, we need to ensure that a is the unique winner. In both these cases, the nonbribed voters veto a, no matter what. This means that the location of the bribed voters doesn't matter (though their prices and weights will). In the priced case, bribe the cheapest voters (as many as possible within the budget) and bribe to minimize the maximum score of b and c. So that's in P. The priced and weighted (and so also the weighted) case is in NP: Guess a set of voters to bribe, check that they are within the budget, guess votes for the bribed voters, and have all nonbribed voters veto a. NP-hardness for the weighted case follows from the NP-hardness for weighted manipulation plus the proof of Theorem 5.1.
3. If $d = b$ in the constructive case, the goal is to not have c win uniquely, and if $d = c$ in the destructive case, the goal is to have c not win. In this case, all bribed voters veto c (no matter what). The same argument as above shows that priced bribery is in P.

   The unpriced, weighted cases are coNP-complete. To show that the complement is in NP, pick the k heaviest voters to bribe. Then check if you can partition the remaining voters to veto a or b in such a way that c wins uniquely in the constructive case or that c wins in the destructive case. Note that it is always best for the briber to bribe the k heaviest voters: Swapping the weights of a lighter voter to be bribed with a heavier voter not to be bribed will never make things worse for the briber. To show hardness, note that the complement is basically (the standard NP-complete problem) Partition.

To show the upper bound for the priced, weighted case, note that we need to check that there exists a set of voters that can be bribed within the budget such that if all bribed voters veto c, then for all votes for the nonbribed voters, c is not the unique winner (in the constructive case) or not a winner (in the destructive case). This is clearly in Σ_2^p. With some care, we can show that it is in fact in Δ_2^p. First use an NP oracle to determine the largest possible total weight (within the budget) of bribed voters. Then determine whether we should bribe u, by using the oracle again to determine the largest possible total weight (within the budget) of bribed voters, assuming we bribe u. If that weight is the same as the previous weight, bribe u. Otherwise, do not bribe u. Repeating this will give us a set of voters to bribe of maximum weight. All these voters will veto c. It remains to check that for all votes for the nonbribed voters, c is not the unique winner (in the constructive case) or not a winner (in the destructive case). This takes one more query to an NP oracle.

Putting all cases together, the priced case (and so also the unpriced, unweighted case) is in P. The weighted case is $\mathrm{P}^{\mathrm{NP}[1]}$-complete, since it can be written as the union of a NP-complete set and a coNP-complete set that are P-separable. (Two sets A and B are P-separable [GS88] if there exists a set X computable in polynomial time such that $A \subseteq X \subseteq \overline{B}$.)

The priced, weighted case inherits the $\mathrm{P}^{\mathrm{NP}[1]}$-hardness from the weighted case (in fact, it already inherited this from online manipulation, using Theorem 5.1), and it is in Δ_2^p. ❑

We end this section by looking at approval voting. In approval voting, each candidate scores a point when it is approved in a vote and the candidates with the most points are the winners. Though Approval-Bribery is NP-complete [FHH09, Theorem 4.2], we show that online-Approval-Bribery (and even online-Approval-Weighted-Bribery and online-Approval-$Bribery) are in P. This implies that even for natural systems, bribery can be harder than online bribery (assuming $\mathrm{P} \neq \mathrm{NP}$).

Theorem 5.7. *1.* online-Approval-Bribery*,* online-Approval-Destructive-Bribery*,* online-Approval-$Bribery*,* online-Approval-Destructive-$Bribery*,* online-Approval-Weighted-Bribery*, and* online-Approval-Destructive-Weighted-Bribery *are each in* P.

2. online-Approval-Weighted-$Bribery *and* online-Approval-Destructive-Weighted-$Bribery *are each* NP*-complete.*

The proof is made available through the technical-report version of this paper [HHR19].

Acknowledgments

We are grateful to the anonymous TARK 2019 referees for helpful comments.

References

[BCE13] F. Brandt, V. Conitzer & U. Endriss (2013): *Computational Social Choice*. In G. Weiss, editor: *Multiagent Systems*, 2nd edition, MIT Press, pp. 213–284.

[BCE+16] F. Brandt, V. Conitzer, U. Endriss, J. Lang & A. Procaccia, editors (2016): *Handbook of Computational Social Choice*. Cambridge University Press, doi:10.1017/CBO9781107446984.

[BCF+16] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein & R. Niedermeier (2016): *Prices Matter for the Parameterized Complexity of Shift Bribery*. Information and Computation 251, pp. 140–164, doi:10.1016/j.ic.2016.08.003.

[BE98] A. Borodin & R. El-Yaniv (1998): *Online Computation and Competitive Analysis*. Cambridge

University Press.

[BTT89] J. Bartholdi, III, C. Tovey & M. Trick (1989): *The Computational Difficulty of Manipulating an Election*. Social Choice and Welfare 6(3), pp. 227–241, doi:10.1007/BF00295861.

[CKS81] A. Chandra, D. Kozen & L. Stockmeyer (1981): *Alternation*. Journal of ACM 26(1), pp. 114–133, doi:10.1145/322234.322243.

[CLM+12] Y. Chevaleyre, J. Lang, N. Maudet, J. Monnot & L. Xia (2012): *New Candidates Welcome! Possible Winners with Respect to the Addition of New Candidates*. Mathematical Social Sciences 64(1), pp. 74–88, doi:10.1016/j.mathsocsci.2011.12.003.

[CSL07] V. Conitzer, T. Sandholm & J. Lang (2007): *When Are Elections with Few Candidates Hard to Manipulate?* Journal of the ACM 54(3), p. Article 14, doi:10.1145/1236457.1236461.

[DE10] Y. Desmedt & E. Elkind (2010): *Equilibria of Plurality Voting with Abstentions*. In: Proceedings of the 11th ACM Conference on Electronic Commerce, ACM Press, pp. 347–356, doi:10.1145/1807342.1807398.

[DP01] E. Dekel & M. Piccione (2001): *Sequential Voting Procedures in Symmetric Binary Elections*. Journal of Political Economy 108(1), pp. 34–55, doi:10.1086/262110.

[EF10] E. Elkind & P. Faliszewski (2010): *Approximation Algorithms for Campaign Management*. In: Proceedings of the 6th International Workshop On Internet And Network Economics, pp. 473–482, doi:10.1145/268999.269002.

[EFS09] E. Elkind, P. Faliszewski & A. Slinko (2009): *Swap Bribery*. In: Proceedings of the 2nd International Symposium on Algorithmic Game Theory, Springer-Verlag Lecture Notes in Computer Science #5814, pp. 299–310, doi:10.1016/j.artint.2008.11.005.

[Fal08] P. Faliszewski (2008): *Nonuniform Bribery*. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, International Foundation for Autonomous Agents and Multiagent Systems, pp. 1569–1572.

[FH16] Z. Fitzsimmons & E. Hemaspaandra (2016): *High-Multiplicity Election Problems*. Technical Report arXiv:1611.08927 [cs.GT], Computing Research Repository, arXiv.org/corr/. Revised, April 2019.

[FHH09] P. Faliszewski, E. Hemaspaandra & L. Hemaspaandra (2009): *How Hard Is Bribery in Elections?* Journal of Artificial Intelligence Research 35, pp. 485–532, doi:10.1613/jair.2676.

[FHH16] Z. Fitzsimmons, E. Hemaspaandra & L. Hemaspaandra (2016): *Manipulation Complexity of Same-System Runoff Elections*. Annals of Mathematics and Artificial Intelligence 77(3–4), pp. 159–189, doi:10.1016/j.artint.2008.11.005.

[GS88] J. Grollmann & A. Selman (1988): *Complexity Measures for Public-Key Cryptosystems*. SIAM Journal on Computing 17(2), pp. 309–335, doi:10.1137/0217018.

[Hem18] L. Hemaspaandra (2018): *Computational Social Choice and Computational Complexity: BFFs?* In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI Press, pp. 7971–7977.

[HHR14] E. Hemaspaandra, L. Hemaspaandra & J. Rothe (2014): *The Complexity of Online Manipulation of Sequential Elections*. Journal of Computer and System Sciences 80(4), pp. 697–710, doi:10.1016/j.jcss.2013.10.001.

[HHR17a] E. Hemaspaandra, L. Hemaspaandra & J. Rothe (2017): *The Complexity of Controlling Candidate-Sequential Elections*. Theoretical Computer Science 678, pp. 14–21, doi:10.1016/j.tcs.2017.03.037.

[HHR17b] E. Hemaspaandra, L. Hemaspaandra & J. Rothe (2017): *The Complexity of Online Voter Control in Sequential Elections*. Autonomous Agents and Multi-Agent Systems 31(5), pp. 1055–1076, doi:10.1007/s10458-016-9349-1.

[HHR19] E. Hemaspaandra, L. Hemaspaandra & J. Rothe (2019): *The Complexity of Online Bribery in Sequential Elections*. Technical Report arXiv:1906.08308 [cs.GT], Computing Research Repository, arXiv.org/corr/.

[LLS75] R. Ladner, N. Lynch & A. Selman (1975): *A Comparison of Polynomial Time Reducibilities*. Theoretical

Computer Science 1(2), pp. 103–124, doi:10.1016/0304-3975(75)90016-X.

[MS72] A. Meyer & L. Stockmeyer (1972): *The Equivalence Problem for Regular Expressions with Squaring Requires Exponential Space*. In: *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, IEEE Press, pp. 125–129, doi:10.1109/SWAT.1972.29.

[PP13] D. Parkes & A. Procaccia (2013): *Dynamic Social Choice with Evolving Preferences*. In: *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, AAAI Press, pp. 767–773.

[PR97] K. Poole & H. Rosenthal (1997): *Congress: A Political-Economic History of Roll-Call Voting*. Oxford University Press.

[Rot16] J. Rothe, editor (2016): *Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Springer, doi:10.1007/978-3-662-47904-9.

[Sch83] U. Schöning (1983): *A Low and a High Hierarchy within NP*. *Journal of Computer and System Sciences* 27(1), pp. 14–28, doi:10.1016/0022-0000(83)90027-2.

[Sel82] A. Selman (1982): *Reductions on NP and P-Selective Sets*. *Theoretical Computer Science* 19(3), pp. 287–304, doi:10.1016/0304-3975(82)90039-1.

[Slo93] B. Sloth (1993): *The Theory of Voting and Equilibria in Noncooperative Games*. *Games and Economic Behavior* 5(1), pp. 152–169, doi:10.1006/game.1993.1008.

[Sto76] L. Stockmeyer (1976): *The Polynomial-Time Hierarchy*. *Theoretical Computer Science* 3(1), pp. 1–22, doi:10.1016/0304-3975(76)90061-X.

[Ten04] M. Tennenholtz (2004): *Transitive Voting*. In: *Proceedings of the 5th ACM Conference on Electronic Commerce*, ACM Press, pp. 230–231, doi:10.1145/988772.988808.

[XC10] L. Xia & V. Conitzer (2010): *Stackelberg Voting Games: Computational Aspects and Paradoxes*. In: *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI Press, pp. 697–702.