ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-1071

Dichotomy for Pure Scoring Rules Under Manipulative Electoral Actions[1]

Edith Hemaspaandra[2] and **Henning Schnoor**[3]

Abstract. Scoring systems are an extremely important class of election systems. We study the complexity of manipulation, constructive control by deleting voters (CCDV), and bribery for scoring systems. For manipulation, we show that for all scoring rules with a constant number of different coefficients, manipulation is in P. And we conjecture that there is no dichotomy theorem.

On the other hand, we obtain dichotomy theorems for CCDV and bribery. More precisely, we show that both of these problems are easy for 1-approval, 2-approval, 1-veto, 2-veto, 3-veto, generalized 2-veto, and $(2, 1, ..., 1, 0)$, and hard in all other cases. These results are the "dual" of the dichotomy theorem for the constructive control by adding voters (CCAV) problem from [14], but do not at all follow from that result. In particular, proving hardness for CCDV is harder than for CCAV since we do not have control over what the controller can delete, and proving easiness for bribery tends to be harder than for control, since bribery can be viewed as control followed by manipulation.

INTRODUCTION

Elections are an important way to make decisions, both in human and electronic settings. Arguably the most important class of election systems are the scoring rules. A scoring rule is defined by, for each number m of candidates, a scoring vector $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. We typically want these vectors to be somehow similar. This is captured nicely by the notion of pure scoring rules from [4] where the length-$(m+1)$ vector is obtained by adding a coefficient in the length-m vector. Voters have complete tie-free preferences over the candidates, and a candidate ranked ith by a voter receives a score of α_i from that voter. The winners are the candidates with the highest score.

We are interested in determining, for all scoring rules at once, which of them give rise to easy computational problems and which of them lead to hard problems. Theorems of that form are known as dichotomy theorems, and have received a lot of attention in the literature, e.g., in computational social choice [13] and satisfiability problems ([19] and its many follow-ups). These results are in particular interesting due to Ladner's classic result [16] that if P≠NP, then there are problems in NP that are neither NP-complete nor solvable in polynomial time. For weighted scoring rules, in which each voter has a weight w and counts as w regular voters, there are dichotomy theorems for all standard manipulative actions: manipulation [13], bribery [9], and control [10]. The arguably more natural unweighted case is much harder to analyze (since in the unweighted case we can only get hardness when the number of candidates is unbounded, whereas in the weighted case hardness already occurs with a fixed number of candidates; since weighted dichotomy theorems typically look at a fixed number of candidates, the results for the unweighted cases do not at all follow from the results for the weighted cases). Despite the prevalence of scoring rules, there are only two dichotomy theorems for the unweighted case, namely for the possible winner problem [4, 3] and for the constructive control by adding voters (CCAV) problem [14].

In this paper, we look at bribery and manipulation for unweighted scoring rules, and, since bribery can be viewed as deleting voters followed by manipulation, we also look at the constructive control by deleting voters (CCDV) problem.

For manipulation, we show that for all scoring rules for which there is a constant upper bound on the number of different coefficients used for any number of candidates, the manipulation problem is in P. This subsumes all known polynomial-time results for unweighted manipulation for scoring rules. We conjecture that there is no dichotomy theorem for manipulation.

For bribery and CCDV, we obtain a dichotomy theorem for pure scoring rules. In particular, we show exactly when these problems are easy (in P) and that they are hard (NP-complete) in all other cases. Interestingly, our characterization is the "dual" of the CCAV characterization in the following sense: For every scoring rule f, the complexity of f-CCDV (and of f-bribery) is the same as for $dual\,(f)$-CCAV, where $dual\,(f)$ is obtained from f by multiplying each entry in a scoring vector by -1, and reversing the order of the vector.

These results are quite surprising: CCDV has less structure to encode hard problems into it than CCAV, but we still obtain the same complexity characterization (modulo duality). On the other hand, bribery can be seen as the combination of CCDV and manipulation, but the complexity is the same as for CCDV. However, in another sense bribery behaves very differently from CCDV: By slightly changing the definition of the bribery problem (to a variant where a subset of the voters is immune to bribery attempts), the complexity of the problem increases from polynomial time to NP-completeness for some generators, while the corresponding change never changes the complexity of CCDV. In that sense, the complexity of CCDV is much more robust than that of bribery.

The paper is structured as follows: In Section 1, we introduce relevant definitions, including the specific problems we study in this paper. In Section 2, we state our results on manipulation. Section 3 contains our dichotomy result for CCDV and bribery. Our individual results for CDDV and bribery can be found in Sections 4 and 5, respectively. We conclude with open questions in Section 6. Most of our proofs can only be found in the full version of this paper, [15].

[1] Supported in part by NSF grant CCF-1101452 and by COST Action IC1205. Work done in part while H. Schnoor visited the University of Rochester supported by an STSM grant of Cost Action IC1205.
[2] Rochester Institute of Technology, Rochester, NY, USA
[3] Kiel University, Kiel, Germany

1 PRELIMINARIES

An *election* consists of a non-empty, finite set of candidates and a finite set of voters. Each voter is identified with her vote, which is a linear order on the set of candidates. An *election system* or *voting rule* is a rule that, given an election, determines the set of candidates who are winners of the election according to this rule. A *scoring vector* for m candidates is a vector $(\alpha_1, \ldots, \alpha_m)$ of integer coefficients, where $\alpha_i \geq \alpha_{i+1}$ for all $1 \leq i < m$. Such a vector defines a voting rule for elections with m candidates by awarding, for each vote, α_i points to the candidate ranked in the i-th position of this vote, and defining the candidates with the most points to be the election winners. A *scoring rule* is an election system that for each number of candidates applies an appropriate scoring vector. Such a system can be described by a *generator*, which is a function f such that for each $m \in \mathbb{N}$, $f(m) = (\alpha_1^m, \ldots, \alpha_m^m)$ is a scoring vector for m candidates.

Well-known scoring rules are Borda (using $f(m) = (m-1, m-2, \ldots, 1, 0)$), k-approval (using $f(m) = (\underbrace{1, \ldots, 1}_{k \text{ many}}, 0, \ldots, 0)$) and k-veto (using $f(m) = (1, \ldots, 1, \underbrace{0, \ldots, 0}_{k \text{ many}})$) for natural numbers k. We usually identify a generator with the election system it defines.

To capture that elections for different numbers of candidates should use "similar" generators, we use the following notion [4]: A generator f as above is *pure*, if for all $m \geq 1$, the vector $f(m)$ can be obtained from the vector $f(m+1)$ by removing one coefficient from the sequence. [14] contains a study of different notions of purity for generators and the class of pure generators with rational numbers contains all generators studied in that paper.

We now define standard manipulative actions: Control [2], manipulation [1, 7], and bribery [9], for generators.

Definition 1.1. *For a generator f, the* constructive control problem for f by deleting voters*, f-CCDV, is the following problem: Given a set of voters V over a set of candidates C, a candidate $p \in C$ and a number k, is there a subset $V' \subseteq V$ with $\|V'\| \leq k$ such that p is a winner of the election if the votes in $V - V'$ are evaluated using f?*

A similar problem, the *constructive control problem for f by adding voters*, called f-CCAV, asks whether p can be made a winner by adding to V at most k voters from a given set of so-called unregistered voters. In the *manipulation problem for f*, we are given a set V of nonmanipulative voters and a set of manipulators, and we ask whether p can be made a winner by setting the votes of the manipulators, with no restriction on how these votes can be chosen. Finally, the *bribery problem* for f asks whether p can be made a winner by replacing up to k votes in V with the same number of arbitrary votes. The problems f-CCDV, f-bribery, and f-manipulation are easily seen to be in NP for all generators we study in this paper.

Two scoring vectors $(\alpha_1, \ldots, \alpha_m)$ and $(\beta_1, \ldots, \beta_m)$ are *equivalent* if they describe the same election system, i.e., if for any election, they lead to the same winner set. It is easy to see [14] that this is the case if and only if there are numbers $\gamma > 0$ and δ such that for each i, $\beta_i = \gamma\alpha_i + \delta$. We say that f_1 and f_2 are *ultimately equivalent* if $f_1(m)$ and $f_2(m)$ are equivalent for all but finitely many m. It is easy to see that in this case, CCDV, bribery, and manipulation have the same complexity for f_1 and f_2.

For algorithms, we need the function f to be efficiently computable. A generator f is polynomial-time uniform if $f(m)$ can be computed in polynomial time, given m in unary. (Given m in binary, polynomial time would not suffice to even write down a sequence of m numbers.) For the remainder of this paper, a generator is always a polynomial-time uniform pure generator with rational coefficients.

2 MANIPULATION

Our main result on manipulation is the following: Every generator f for which there is a fixed, finite upper bound on the number of coefficients that are used for any number of candidates has a polynomial-time solvable manipulation problem. We mention that this result also holds for generators that are not pure (but still are polynomial-time uniform). We note that the special cases where f generates k-approval or k-veto were shown in [21].

Theorem 2.1. *Let f be a generator such that there is a constant c such that for each number m of candidates, at most c different coefficients appear in the vector $f(m)$. Then f-manipulation can be solved in polynomial time.*

We give a proof sketch for a simple special case of the theorem, namely generators f of the form $f = (0, \ldots, 0, -\beta, -\alpha)$. With great care, this proof sketch generalizes to the general case.

Proof. (Sketch) Consider an election instance with a preferred candidate p, a set of candidates $C = \{c_1, \ldots, c_m, p\}$, a set V of nonmanipulative voters, and a set of k manipulators. From V, we compute a surplus $\mathit{surplus}(c) = \mathit{score}(c) - \mathit{score}(p)$ for each $c \in C$, this can be done easily as f is polynomial-time uniform. Clearly, we can assume that all manipulators will vote p first.

The obvious greedy approach of having a manipulator rank a candidate with the largest surplus last will not always work: If $\beta = 2$, $\alpha = 3$, the surplus of c_1 is 4, the surplus of c_2 and c_3 is 3, and we have two manipulators, the only successful manipulation is to have the manipulators vote $\cdots > c_1 > c_2$ and $\cdots > c_1 > c_3$ and so we cannot put c_1 last.

If there is a successful manipulation, then for all $i \in \{1, \ldots, m\}$, there are natural numbers x_i and y_i (the number of times c_i is ranked next to last / last by a manipulator) such that:

1. $x_i + y_i \leq k$,
2. $\sum_{1 \leq j \leq m} x_j = k$,
3. $\sum_{1 \leq j \leq m} y_j = k$, and
4. $\mathit{surplus}(c_i) - \beta x_i - \alpha y_i \leq 0$.

We define a Boolean predicate M such that $M(k, k, k, \mathit{surplus}(c_1), \ldots, \mathit{surplus}(c_m))$ is true if and only if there exists a successful manipulation. M is defined as follows. For $\ell \geq 0$, $M(k, k_\beta, k_\alpha, s_1, \ldots, s_\ell)$ is true if and only if for all i, $1 \leq i \leq \ell$, there exist natural numbers x_i and y_i such that

1. $x_i + y_i \leq k$,
2. $\sum_{1 \leq i \leq \ell} x_i = k_\beta$,
3. $\sum_{1 \leq i \leq \ell} y_i = k_\alpha$, and
4. $s_i - \beta x_i - \alpha y_i \leq 0$.

It is immediate that if there is a successful manipulation, then $M(k, k, k, \mathit{surplus}(c_1), \ldots, \mathit{surplus}(c_m))$ is true. It is not so easy to see that the converse holds. This is shown by induction on k. It is easy to come up with a simple ad-hoc proof for the simple case we are looking at here, but we will instead describe an approach that generalizes to the general case.

The inductive step uses the following argument. If $M(k+1, k_\beta, k_\alpha, s_1, \ldots, s_\ell)$ is true, let $X = \{c_i \mid x_i > 0\}$ and let $Y = \{c_i \mid y_i > 0\}$. Then the sequence (X, Y) can be shown to fulfill the "marriage condition," which then, by Hall's Theorem [12], implies that there is a "traversal," i.e., a sequence of distinct representatives of this sequence of sets which then gives us a vote for one of the manipulators. In this particular case, the traversal consists of two distinct candidates (c_i, c_j) such that $x_i > 0$ and $y_j > 0$. Let one

manipulator vote $\cdots > c_i > c_j$, subtract 1 from x_i and y_j, subtract β from *surplus*(c_i), and subtract α from from *surplus*(c_j). It follows from the induction hypothesis that the remaining k manipulators can vote to make p a winner.

Finally, we show by dynamic programming that M is computable in polynomial time for unary k, k_α, k_β (this suffices to solve the manipulation problem in polynomial time). This holds since:

1. $M(k, k_\beta, k_\alpha)$ is true if and only if $k_\beta = k_\alpha = 0$.
2. For $\ell \geq 1$, $M(k, k_\beta, k_\alpha, s_1, \ldots, s_\ell)$ if and only if there exist natural numbers x_ℓ and y_ℓ such that:
   (a) $x_\ell + y_\ell \leq k$,
   (b) $x_\ell \leq k_\beta$,
   (c) $y_\ell \leq k_\alpha$,
   (d) $s_\ell - \beta x_\ell - \alpha y_\ell \leq 0$, and
   (e) $M(k, k_\beta - x_\ell, k_\alpha - y_\ell, s_1, \ldots, s_{\ell-1})$.

□

Given Theorem 2.1 and the fact that manipulation for Borda is NP-complete [5, 8], it is natural to ask whether the manipulation problem is NP-complete for all generators not covered by Theorem 2.1. But this is extremely unlikely: Though our approach does not give polynomial-time algorithms when the number of coefficients is unbounded, it will give quasipolynomial algorithms when the coefficients are small enough and grow slowly enough.

It is also conceivable that additional cases will be in P. Though a general greedy approach seems unlikely (as manipulation for Borda is NP-complete), a greedy approach for specific cases is still possible.

We conjecture that there is no dichotomy theorem for manipulation for pure scoring rules, with different intermediate (between P and NP-complete) complexities showing up.

3 CCDV AND BRIBERY DICHOTOMY

We completely characterize the complexity of f-CCDV and f-bribery for every generator f. For each f, these two problems are polynomial-time equivalent, and are polynomial-time solvable or NP-complete. For the cases where f generates k-approval or k-veto, the complexity classification is already stated in [17]. For CCDV, the special case where f generates 1-approval was shown in [2]. For bribery, the special cases where f generates 1-approval or 1-veto were shown in [9]. We also note the existence of an unpublished manuscript that proves NP-hardness for bribery for generators of the form $(\alpha, \beta, 0, \ldots, 0)$, where α and β are coprimes with $\alpha > \beta \geq 1$ [6]. The remainder of this paper contains a proof sketch of our following main result on CCDV and bribery:

Theorem 3.1. *Let f be a pure, polynomial-time uniform generator. If f is ultimately equivalent to one of the following generators, then f-CCDV and f-bribery can be solved in polynomial time:*

1. $f_1 = (1, \ldots, 1, 0, 0, 0)$ *(3-veto),*
2. $f_2 = (1, 0, \ldots, 0)$ *(1-approval),*
3. $f_3 = (1, 1, 0, \ldots, 0)$ *(2-approval),*
4. *for some $\alpha \geq \beta \geq 0$, $f_4 = (0, \ldots, 0, -\beta, -\alpha)$ (this includes triviality, 1-veto, and 2-veto),*
5. $f_5 = (2, 1, \ldots, 1, 0)$.

Otherwise, f-CCDV and f-bribery are NP-complete.

This dichotomy theorem mirrors the one obtained for CCAV in [14] (modulo duality, see below). For the relationship of CCDV and CCAV, this implies that the partition of voters into those that can be affected by the controller and those who are immune to such attempts does not have any influence on the complexity of our decision problems for the class of generators we study.

In particular, our results imply that CCDV is "robust" in the following sense: The complexity of CCDV does not depend on whether we add a bit to each voter stating whether she can be deleted or not. This will be made formal below in our discussion of CCDV*.

The situation is different for bribery: Generalizing the bribery problem to allow marking some voters as "unbribable" increases the complexity to NP-complete for some generators. As an example of this phenomenon, we state the following result. (The version of bribery defined here may be of independent interest, but is only used here to highlight the differences between CCDV and bribery.)

Theorem 3.2. *The variation of the bribery problem for $(0, \ldots, 0, -1, -2)$ where each voter has a bit that states whether this voter can be bribed or not is NP-complete.*

4 CONTROL BY DELETING VOTERS

In this section we give an overview over our proof of the CCDV-part of Theorem 3.1. In Section 4.1, we show that our CCDV polynomial-time cases easily follow from a relationship to CCAV, whose complexity was studied in [14]. Sections 4.2 and 4.3 then contain our hardness results for CCDV.

4.1 Relationship between CCAV and CCDV and CCDV polynomial-time cases

CCDV and CCAV are closely related as follows: For a generator f, let $dual\,(f)$ be the generator obtained from f by multiplying each coefficient with -1 and reversing the order of the coefficients (to maintain monotonicity). Removing a vote in an f-election has the same effect as adding the same vote in a $dual\,(f)$-election. Using this observation, we show that f-CCDV reduces to $dual\,(f)$-CCAV for all generators f. In particular, if $dual\,(f)$-CCAV can be solved in polynomial time, then this is also true for f-CCDV. A similar relation holds for weighted k-approval and k-veto elections [10].

The other direction of this relationship does not follow so easily, as there is an important difference between CCDV and CCAV: In CCAV, the set of voters is partitioned into a set of *registered* voters and a set of *potential* voters, where the controller's actions can only influence the potential voters. This provides the problem with additional structure, as the controller cannot modify the registered voters. We often call these registered votes, which the controller cannot influence anymore, *setup votes*, as these allow us to set up the scenario of an NP-hard problem in hardness proofs.

The CCDV problem does not have a corresponding structure; here every vote may be (potentially) deleted by the controller. This makes it harder to construct the above-mentioned setup votes: Since we cannot simply "forbid" the controller to delete certain votes, hardness proofs for CCDV need to "setup" the relevant scenario with votes that are designed to be "unattractive" to delete for the controller.

To obtain CCDV hardness results, it is therefore natural to consider the following analog to the CCAV problem: CCDV* is a version of CCDV providing the additional structure that CCAV has: In CCDV*, the set of votes is partitioned into a set R of voters that cannot be deleted, and voters D that can be deleted. From the above discussion,

it follows that the complexities of CCAV and CCDV* are related with the following duality, which, together with the polynomial-time results obtained for CCAV in [14] immediately implies the polynomial-time CCDV cases of Theorem 3.1.

Proposition 4.1. *For every generator f, f-CCDV* and dual (f)-CCAV are polynomially equivalent.*

Proposition 4.1 is not completely trivial, since the reductions must convert election instances maintaining the relative points of the candidates. However, this is done using standard constructions.

From Theorem 3.1 and the results in [14], it follows that f-CCDV and f-CCDV* always have the same complexity. For bribery, the situation is quite different, see the above Theorem 3.2.

All polynomial-time cases for CCDV follow directly from the above relationship. This is not surprising, since CCDV in the above-discussed sense has less structure than CCAV, and thus easiness results for CCAV translate to (dual) CCDV. The interesting part of our dichotomy is the converse: If CCAV is NP-hard for some generator f, then CCDV is hard for $dual\,(f)$ as well.

A natural approach for the proof of the dichotomy theorem, suggested by Proposition 4.1, is to show that f-CCDV* always reduces to f-CCDV. While this does in fact follow, proving a generic reduction from f-CCDV* to f-CCDV for all generators f seems to be difficult, due to the additional structure provided by CCDV*.

Our proof of the CCDV part of Theorem 3.1 therefore uses a case distinction to obtain f-CCDV hardness for each remaining pure, polynomial-time uniform generator f.

We note that due to the relationship between CCDV and CCAV, all CCAV-hardness results in [14] easily follow from the results obtained in the current paper. However, our proofs make use of the results and proofs from [14].

4.2 CCDV hardness: "few coefficients"

We first consider generators with "few" different coefficients, i.e., generators of the form $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ for rationals $\alpha_1, \ldots, \alpha_6$. Using equivalence-preserving transformations, we can assume that all α_i are nonnegative integers. Note that a generator of this form is trivially polynomial-time uniform.

4.2.1 Reductions from CCDV*

A general reduction from f-CCDV* to f-CCDV does not seem feasible, as discussed above. However, there are cases where hardness of f-CCDV* leads to hardness of f-CCDV with a direct proof. The following two results (Theorems 4.2 and 4.3) are proven in this way.

Theorem 4.2. *Let $f = (\alpha, 0, \ldots, 0, -\beta)$ be a generator with $1 \leq \alpha < \beta$. Then f-CCDV is NP-complete.*

Proof. (Sketch) $dual\,(f)$-CCAV is NP-complete due to [14]. Proposition 4.1 implies that f-CCDV* is NP-complete as well. We show that f-CCDV* reduces to f-CCDV. Given an instance of f-CCDV*, we convert it into an equivalent instance of f-CCDV by replacing the undeletable votes R with votes that

1. result in the same relative points as the votes in R, and
2. are not deleted in a successful CCDV action.

We denote a vote $c_1 > c_2 > \cdots > c_{m-1} > c_m$ simply as $c_1 > c_m$ (the rest is irrelevant). We can use light preprocessing to convert the instance into one where no deletable vote has p in the first or last position; this allows us to compute the number $score(p)$ of points that p will have after the delete action. Similarly, we can assume that $score(p) = N_p\alpha$ for some $N_p \geq 2$.

Satisfying point 1 above boils down to adding, for an arbitrary candidate $c \neq p$, votes that let c gain α points against p, and which will not be deleted. This is done as follows: We add dummy candidates $d_1, \ldots, d_\ell$ (for a suitably chosen number ℓ) and a single vote $c > d_1$, letting c gain α points relative to p. To ensure that the vote $c > d_1$ cannot be removed, we add votes setting up the scores as follows:

- Each d_i for $1 \leq i \leq \ell - 1$ ties with p,
- the only way to make d_i lose points (relative to p) is to remove votes $d_i > d_{i+1}$, which then lets d_{i+1} gain points (relative to p).

Hence removing the vote $c > d_1$, which lets d_1 gain β points relative to p requires the controller to remove votes of the form $d_1 > d_2$, which each lets d_2 gain β points. This process continues for d_i with $i \geq 2$. Thus, removing $c > d_1$ triggers a "chain" of additional removals—more than the budget allows. The numbers of votes needed to setup grows exponentially in the number of steps. However, since the controller can only remove a polynomial number of votes, we only require logarithmically many steps, yielding a polynomial construction.

Constructing the actual set of votes that results in the above scores and satisfies the two points above is nontrivial, the construction is in fact the main technical difficulty in the proof. □

The following result is shown similarly, the difference is that instead of logarithmically many steps of an exponentially growing construction, here we apply a simpler linear process.

Theorem 4.3. *Let $f = (\alpha, 0, \ldots, 0, -\beta)$ be a generator with $\alpha > \beta \geq 1$. Then f-CCDV is NP-complete.*

4.2.2 Reductions by inspection of the CCAV reduction

Similarly to the preceding Section 4.2.1, the results in this section are proved by a reduction from f-CCDV* to f-CCDV. However, while the reductions above were "generic" (reducing from an arbitrary CCDV*-instance), we now start with instances of f-CCDV* produced by the hardness proof of f-CCDV*. Therefore, we do not need to construct "setup votes" that implement any possible given set of scores, but only need to achieve exactly the points used in the hardness proof of $dual\,(f)$-CCAV in [14].

In the following theorem, this is a significant advantage, as here the "setup votes" grant more points to the preferred candidate than to the remaining candidates. Therefore, it is easy to construct these votes in such a way that the controller has no incentive to delete them.

The proof of the theorem also illustrates the relationship between hardness results for CCAV [14] and the hardness results for CCDV and bribery we obtain in the current paper: The proof of Theorem 4.4 below uses the reduction of the corresponding result for CCAV in [14] as a starting point, but is technically more involved. We will later re-use parts of the following construction to obtain the corresponding hardness result for bribery as well, in the later Theorem 5.5.

Theorem 4.4. *Let $f = (\alpha_3, \ldots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ with $\alpha_3 > \alpha_4 > \alpha_6$. Then f-CCDV is NP-complete.*

Proof. We equivalently write f as $f = (0, \ldots, 0, -\gamma, -\beta, -\alpha)$ with $0 < \gamma < \alpha$. Then, $dual\,(f) = (\alpha, \beta, \gamma, 0, \ldots, 0)$. From [14], it follows that $dual\,(f)$-CCAV is NP-complete, and hence, due to

Proposition 4.1, it suffices to show that f-CCDV* reduces to f-CCDV. However, we do not give a general reduction between these two problems, but only one from the set of hard instances produced by the corresponding hardness proof from [14]. This makes our reduction simpler, since the "setup votes" from that reduction favor the candidate p and are therefore not attractive to delete.

Let an f-CCDV* instance with (un)deletable D (R), preferred candidate p, and budget ℓ be given. From the proof of Proposition 4.1 (in our full version [15]), we can assume that the instance is obtained from the hardness proof of $dual\,(f)$-CCAV as follows:

- the votes in D are exactly the votes available for addition in the CCAV instance, with the order of candidates reversed,
- the relative points gained by the candidates from the votes in $R \cup D$ are the same as the points of the candidates in the CCAV instance (before the addition of votes by the controller).

The hardness proof of $dual\,(f)$-CCAV uses a reduction from 3DM, which is the following problem: Given a multiset $M \subseteq X \times Y \times Z$ with X, Y and Z pairwise disjoint sets of equal size such that each $s \in X \cup Y \cup Z$ appears in exactly 3 tuples of M, decide whether there is a set $C \subseteq M$ with $\|C\| = \|X\|$ such that each $s \in X \cup Y \cup Z$ appears in some tuple of C (we also say that C *covers* s). From the problem definition, it follows that $\|M\| = 3\|X\|$. The condition that each s appears in exactly 3 tuples is not standard; we prove in the full version [15] of this paper that this version of 3DM remains NP-complete.

Let $M \subseteq X \times Y \times Z$ be an instance of 3DM. Following the notation used in [14], we set $n = \|M\|$ and $k = \|X\|$. Since $\|M\| = 3\|X\|$ in every 3DM instance, it follows that $n = 3k$. The hardness proof of $dual\,(f)$-CCAV, translated to the CCDV setting (i.e., we present the votes as in the CCDV instance—as reversals of votes from the CCAV instance), constructs the following situation:

- the candidate set is $\{p\} \cup X \cup Y \cup Z \cup \{S_i, S_i' \mid S_i \in M\}$,
- for each $S_i = (x, y, z) \in M$, the following votes are available for deletion (we only list the non-zero points part of each vote):
  - $\cdots > S_i > p > x$
  - $\cdots > S_i > p > y$
  - $\cdots > S_i' > p > z$
  - $\cdots > S_i' > p > S_i$
- the relative scores resulting from the registered voters of the CCAV instance (i.e., the undeletable voters of the CCDV* instance) are as follows:
  - $score_{final}(p) = \alpha + 2\gamma$,
  - $score_{final}(c) = (n + 2k)\beta + 2\gamma$ for each $c \in X \cup Y \cup Z$,
  - $score_{final}(S_i) = (n + 2k)\beta + \min(\alpha, 2\gamma)$,
  - $score_{final}(S_i') = (n + 2k)\beta + \alpha + \gamma$.

From the above votes introduced for the 3DM-elements, the candidates gain the following initial points—recall that each $c \in X \cup Y \cup Z$ appears in exactly 3 triples from M, and, since $\|M\| = n = 3k$, and there are 4 votes introduced for every element in M, there are exactly $12k$ deletable votes introduced above:

- $score_{3DM}(p) = -12k\beta$, since p gains $-\beta$ points in each of the $12k$ votes,
- $score_{3DM}(S_i) = -2\gamma - \alpha$, since S_i gains 0 points in all of the votes introduced for other elements $S_j \neq S_i \in M$, and gains $-\gamma$ points in 2 of the votes above, and $-\alpha$ points in one of the vote,
- $score_{3DM}(S_i') = -2\gamma$ analogously,
- $score_{3DM}(c) = -3\alpha$ for each $c \in X \cup Y \cup Z$, since each c appears in exactly 3 triples S_i from M.

For each candidate x, the undeletable votes from the CCDV* instance thus let her gain exactly $score_{final}(x) - score_{3DM}(x)$ points (modulo an offset added to the points of all candidates, since the CCAV reduction relies on an implementation lemma that only fixes the relative points of each candidate). The scores that the undeletable votes implement are therefore as follows (recall that $n = 3k$):

candidate x	$score_{final}(x) - score_{3DM}(x)$
p	$12k\beta + \alpha + 2\gamma$
$c \in X \cup Y \cup Z$	$5k\beta + 3\alpha + 2\gamma$
S_i	$5k\beta + \alpha + 2\gamma + \min(\alpha, 2\gamma)$
S_i'	$5k\beta + \alpha + 3\gamma$

Since we only need to implement the relative scores among the candidates, it is enough to consider the points the candidates have to gain/lose relative to p. These are as follows (clearly, p does not gain or lose any points relative to herself):

candidate x	points x needs to lose (result)
$c \in X \cup Y \cup Z$	$7k\beta - 2\alpha$
S_i	$7k\beta - \min(\alpha, 2\gamma)$
S_i'	$7k\beta - \gamma$

(For example, due to the first table, $c \in X \cup Y \cup Z$ must *gain* $7k\beta - 2\alpha$ fewer points than p, equivalently, c must *lose* $7k\beta - 2\alpha$ points relative to p, as indicated in the second table.)

To achieve this, we first introduce three dummy candidates d_1, d_2, and d_3 (by placing them in the 0-point segment of all present votes), and add sufficiently many votes voting all relevant candidates ahead of the dummy candidates. This lets the dummies lose α, β, or γ points relatively to the other candidates; we do this often enough to ensure that the dummy candidates cannot win the election. Using these dummies, we can easily let a relevant candidate $c \neq p$ lose $\delta \in \{\alpha, \beta, \gamma\}$ points, by using a vote placing c in the position worth $-\delta$ points, filling the other two positions out of the last three with dummy candidates, and voting the remaining candidates (including p) in the positions gaining 0 points. Such a vote will never be deleted by the controller, since it has p in the first position.

If $1 = \delta \in \{\alpha, \beta, \gamma\}$, then the required amount of points can easily be achieved by repeatedly losing $\delta = 1$ points as described above. Hence assume that, in particular, $\beta \geq 2$. For the candidate c, we proceed as follows:

- We add $\beta - 2$ many votes letting c lose α point each.
- After this, c, still needs to lose $(7k - \alpha)\beta$ points, which we can achieve by using $7k - \alpha$ many votes letting c lose β points as described above (we can assume that $7k \geq \alpha$).

For candidates S_i and S_i', we proceed analogously. It follows that, to make p a winner, the same number of vote removals is needed in the CCDV* instance as in the constructed f-CCDV instance. □

The following result uses a similar, but more involved argument.

Theorem 4.5. *Let $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3)$ with $\alpha_1 > \alpha_2 > \alpha_3$. Then f-CCDV is NP-complete.*

4.2.3 Direct Proofs

The remaining cases are shown with direct proofs (reducing from a variant of three dimensional matching, 3DM); they are in part similar to the hardness proofs of CCAV in [14].

Theorem 4.6. *f-CCDV is NP-complete in the following cases:*

1. $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ *with* $\alpha_1 > \alpha_3 > \alpha_5$,
2. $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3, \alpha_6)$ *with* $\alpha_1, \alpha_2 > \alpha_3 > \alpha_6$,
3. $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ *with* $\alpha_1 > \alpha_3 > \alpha_4$.

4.3 CCDV hardness: "many" coefficients

In Section 4.2, we have covered all generators of the form $f = (\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$. Generators *not* of this form must satisfy $\alpha_3^m > \alpha_{m-3}^m$ for some value m. We now prove that the CCDV problem is NP-hard for all generators satisfying this condition. The proof is similar to the corresponding result in [14].

An important observation is that when $\alpha_3^m > \alpha_{m-3}^m$ holds for some m, then purity of our generators implies that the condition also holds for any $m' \geq m$. Clearly, if $m \geq 6$ is a multiple of 3 and $\alpha_3^m > \alpha_{m-3}^m$, then one of $\alpha_3^m > \alpha_{\frac{2}{3}m}^m$ and $\alpha_{\frac{2}{3}m}^m > \alpha_{m-3}^m$ must hold. In the first case, the generator—for this number m of candidates—behaves in a similar way as 3-approval in the sense that there are three positions in the vote that let the candidates gain more points than a "large" number of positions later in the sequence of coefficients. In the latter case, the generator (for this number m) behaves similarly to 4-veto, as there are four "bad" positions in the votes.

Therefore, a generator satisfying $\alpha_3^m > \alpha_{m-3}^m$ behaves, for every $m' \geq m$, similarly to 3-approval or to 4-veto. However, the behavior can be different for different values for m. Therefore, our proof of NP-completeness for generators satisfying this condition uses an "adaptive" reduction from the problem 3DM, which, given a 3DM instance, constructs either a reduction exploiting the "3-approval likeness" or the "4-veto likeness" of the generator, depending on the size of the instance (which linearly corresponds to the number of candidates in the constructed election instance). This gives the following result:

Theorem 4.7. *Let f be a polynomial-time pure generator such that $\alpha_3^m > \alpha_{m-3}^m$ for some m. Then f-CCDV is NP-hard.*

5 BRIBERY

Bribery is closely related to both CCDV and manipulation: Bribing k voters can be viewed as deleting k voters followed by adding k manipulation voters. We thus can use our results obtained for CCDV and manipulation in Sections 2 and 4 as a starting point to obtain a complexity classification of the bribery problem.

However, it is not necessarily the case that an optimal bribery consists of an optimal deletion followed by an optimal manipulation (see Example 5.4), and so it is possible for bribery to be hard while the manipulation and deletion of voters problems are easy. However, we will show that for every pure scoring rule f, f-bribery is polynomial-time solvable if and only if f-CCDV is, though the proofs for bribery are more (and sometimes much more) involved.

We also obtain an interesting relationship between the complexities of manipulation and bribery: From Theorem 2.1, it follows that every "few coefficients" case leads to a polynomial-time solvable manipulation problem. From Theorem 3.1, we know that, for CCDV, *only* such cases can be solved in polynomial time (unless P = NP). Therefore, we obtain the following corollary:

Corollary 5.1. *Let f be a polynomial-time uniform pure generator. Then f-manipulation reduces to f-CCDV and f-manipulation reduces to f-bribery.*

5.1 Bribery polynomial-time cases

Our first bribery algorithm (for a generator equivalent to $(2, 1, \ldots, 1, 0)$) uses a reduction to network flow.

Theorem 5.2. *$(1, 0, \ldots, 0, -1)$-bribery is in P.*

Proof. (Sketch) There are three types of voters. Let V_1 be the set of voters that rank p last, let V_2 be the set of voters that rank p neither first nor last, and let V_3 be the set of voters that rank p first. In this case, we can assume without loss of generality that we bribe as many V_1 voters as possible, followed by as many V_2 voters as possible. We never have to bribe V_3 voters. All bribed voters will put p first, so we also know p's score after bribery.

The hardest case is the one where we bribe all V_1 voters and some V_2 voters. We view bribery as deletion followed by manipulation. Delete all V_1 voters. In V_2, deleting a voter with vote $a > \cdots > b$ corresponds to transferring a point from a to b. After deleting k voters, the deleted voters will be bribed to rank p first and to rank some other candidate last. After deleting V_1, for every candidate c, $score(c) = score_{V_2 \cup V_3}(c)$, i.e., the score of c in $V_2 \cup V_3$. For every V_2 voter $a > \cdots > b$ that is deleted, transfer one point from a to b. For every bribe $p > \cdots > d$, delete a point from d. There are exactly k bribes. After bribery, $score(p) = \|V_3\| + k$ and the score of every other candidate needs to be at most $score(p) = \|V_3\| + k$.

It is not too hard to see that this problem can be translated in min-cost network flow problem, in a similar, though somewhat more complicated, way as in the CCAV algorithm for the same generator in [14]. □

We now state our second bribery result.

Theorem 5.3. *Let $\alpha \geq \beta \geq 0$. Then bribery for $(0, \ldots, 0, -\beta, -\alpha)$ is solvable in polynomial time.*

Proof. (Sketch) As in the proof of Theorem 5.2, we partition V into V_1, V_2, and V_3. V_1 consists of all voters in V that rank p last, V_2 consists of all voters in V that rank p second-to-last, and V_3 consists of the remaining voters. In the proof of Theorem 5.2, it was important that we never had to bribe V_3 voters. This is not always the case here, as shown in the example below. That also means that this case is very different from CCDV, since in CCDV we never have to delete V_3 voters. It also shows that an optimal bribery is not always an optimal deletion followed by an optimal manipulation.

Example 5.4. *This example shows that we sometimes need to bribe V_3 voters. We will use the scoring rule $(0, \ldots, 0, -1, -3)$. Let $C = \{p, a, b, c, d, e, f\}$ and let V consist of the following votes:*

$\cdots > p > a, \quad \cdots > e > f, \quad \cdots > f > e,$
$\cdots > p > b, \quad \cdots > e > f, \quad \cdots > f > e,$
$\cdots > p > c.$

Then $score(p) = score(a) = score(b) = score(c) = -3$, $score(d) = 0$, and $score(e) = score(f) = -8$.

We can make p a winner by bribing one of the V_3 voters to vote $p > \cdots > d$. But it is easy to see that we can not make p a winner by bribing a V_2 voter, wlog, the voter voting $\cdots > p > a$, since in the bribed election, the score of p will be at most -2, and so both a and d must be in the last position of the bribed voter.

Though we may need to bribe V_3 voters, we can show that we never need to bribe more than a constant number of V_3 voters. This is crucial in obtaining a polynomial-time bound on a dynamic programming approach similar to, but more complicated than, the one in the proof for Theorem 2.1. □

Together with Lin's results on the complexity of bribery for k-veto and k-approval election systems in [17], the above results prove all polynomial-time bribery cases of our main result, Theorem 3.1. In the remainder of Section 5, we therefore discuss our NP-hardness results for bribery.

5.2 Bribery hardness approach

Our dichotomy results imply that each generator f for which CCDV is NP-hard also has an NP-hard bribery problem. Proving this via a generic reduction from CCDV to bribery does not seem to be easy: Even though bribery can be seen as CCDV followed by manipulation, solving a bribery instance is not the same as first finding an optimal deletion of votes and then performing an optimal manipulation. Therefore, hardness of f-bribery does not easily follow from hardness of f-CCDV. We briefly discuss a proof strategy to obtain a hardness proof for f-bribery from a hardness proof of f-CCDV.

A key difficulty in the construction of a bribery hardness proof is that the manipulation action of the controller allows her more freedom than her delete action: For the latter, the reduction controls the available votes, whereas the manipulation action can use arbitrary permutations of the candidates. However, we can always assume that the bribed voters will vote p in the first position, and will place any "dummy" candidates in the positions following p in their votes. This often allows us to compute the score of p after the bribery action.

To limit the controller's freedom in the manipulation votes, we proceed as follows: We identify a sufficiently long subsequence of the coefficients that differ by only a "small" amount. (Such a sequence exists by definition for the generators treated in Section 4.2, and can be found using a pigeon-hole argument for other generators.) We then set up the points such that the "relevant" candidates must be placed into this "low-variation" sequence of the vote. This is done using "blocking" candidates that must be placed in low-score positions, and dummy candidates occupying the high-score positions (except for the first position, in which the bribed voters always vote p). This ensures that moving a candidate inside the "low-variation" does not make a large difference, and allows the reduction to control the possible manipulation actions very tightly.

A second difficulty is that the controller is more powerful in bribery than in CCDV: In bribery, she can perform a manipulation action in addition to her delete action. Therefore, to make it "hard" for the controller to find an optimal bribery action, the scores in the election instance must be "worse" for p than in the CCDV setting. This leads to setup votes that are more attractive to delete than in the CCDV case. Similarly as in CCDV, the main technical issue is to define setup votes that both obtain the required scores and still will not be deleted by the controller, however due to the reasons above this is even more difficult for bribery as for CCDV.

The ideas outlined above allow us to prove the bribery hardness results of Theorem 3.1.

As an example for our approach to bribery hardness, we prove the following theorem, which is the "bribery version" of Theorem 4.4. We therefore apply our recipe to the proof of Theorem 4.4, which is the CCDV hardness result for the same generator.

Theorem 5.5. *Let $f = (\alpha_3, \dots, \alpha_3, \alpha_4, \alpha_5, \alpha_6)$ with $\alpha_3 > \alpha_4 > \alpha_6$. Then f-bribery is NP-complete.*

Proof. We follow the above recipe to obtain a hardness proof for f-bribery. Applying the recipe requires to make some changes to the construction of the CCDV hardness proof. As in the proof of Theorem 4.4, we write f as $f = (0, \dots, 0, -\gamma, -\beta, -\alpha)$. In particular, since we can assume that all bribed voters will vote p first, the score of p will not change from manipulation votes (but may, of course, change from the deleted votes).

The budget for the controller is, as in the CCAV proof from [14], $n + 2k$, where $n = 3k$, i.e., the budget is $5k$ (here, k is again the number $\|X\|$ from the given 3DM instance). We make the following changes to the setup in the proof of Theorem 4.4:

- We introduce three new distinct candidates b_α, b_β, and b_γ (i.e., even if $\beta = \gamma$, then b_β is still a different candidate from b_γ). These candidates will be placed in the three "relevant" positions of each manipulation vote in every successful bribery action.
- Recall that in the proof of the CCDV hardness result, i.e., Theorem 4.4, each element (x, y, z) leads to four 3DM votes. We make the following addition: For each 3DM vote introduced in this way, we introduce G additional votes obtained from the 3DM votes by swapping the candidates in the $-\gamma$ and $-\beta$-positions (for an appropriate value of G). More precisely: For each $(x, y, z) \in M$, we add the following votes:
  - a single vote $\cdots > S_i > p > x$
  - G many votes $\cdots > p > S_i > x$
  - a single vote $\cdots > S_i > p > y$
  - G many votes $\cdots > p > S_i > y$
  - a single vote $\cdots > S'_i > p > z$
  - G many votes $\cdots > p > S'_i > z$
  - a single vote $\cdots > S'_i > p > S_i$
  - G many votes $\cdots > p > S'_i > S_i$

  The purpose of these added votes (we will call them G-votes in the sequel) is to ensure that the candidates b_α, b_β, and b_γ gain points relatively to p, using votes that are less attractive than the actual 3DM votes from the construction of Theorem 4.4, such that the controller will delete the latter votes instead of the G-votes.

Note that the controller will clearly vote p in the first position of all manipulation votes. Therefore, the score of p will not change from the manipulation votes, but only from the CCDV-aspect of the bribery action. Hence, the score of p behaves in exactly the same way as in the CCDV proof. In the bribery instance we construct, the final scores (i.e., from the 3DM votes including the G-votes plus the setup votes that we will introduce below) will be as follows (recall that $n = 3k$ as above):

- $score_{final}(p) = \alpha + 2\gamma$,
- $score_{final}(c) = 5k\beta + 2\gamma$ for each $c \in X \cup Y \cup Z$,
- $score_{final}(S_i) = 5k\beta + \min(\alpha, 2\gamma)$,
- $score_{final}(S'_i) = 5k\beta + \alpha + \gamma$.
- $score_{final}(b_\heartsuit) = \alpha + 2\gamma + 5k(\beta + \heartsuit)$ for $\heartsuit \in \{\alpha, \beta, \gamma\}$.

The score of $b_\heartsuit$ is such that $b_\heartsuit$ ties with p if $5k$ 3DM votes are removed (letting p gain $5k\beta$ points), and $b_\heartsuit$ is placed in the $-\heartsuit$-position of each of the $5k$ manipulation votes. Since the score of $b_\heartsuit$ cannot be decreased by deleting 3DM votes, this ensures that the $b_\heartsuit$ candidates must in fact be placed in the relevant positions of each manipulation vote, and therefore, with regard to the remainder of the candidates, the construction works as in the CCDV case.

As in the proof of Theorem 4.4, we first compute the scores the candidates receive from the 3DM- and G-votes, for a candidate x, we call this value $score_{3DM}(x)$.

- $score_{3DM}(p) = -12k(\beta + G\gamma)$, since, for each tuple in M, p gains $-4\beta - 4G\gamma$ points, and $\|M\| = 3k$.
- $score_{3DM}(S_i) = -2\gamma - (G+1)\alpha - 2G\beta$,
- $score_{3DM}(S_i') = -2\gamma - 2G\beta$,
- $score_{3DM}(c) = -3\alpha(G+1)$ for each $c \in X \cup Y \cup Z$, since each c appears in exactly 3 tuples from M,
- $score_{3DM}(b_\heartsuit) = 0$ for each $\heartsuit \in \{\alpha, \beta, \gamma\}$.

For each candidate x, with $score_{setup}(x)$ we denote the points that x needs to receive from the setup votes in order to ensure that $score_{final}(x) = score_{3DM}(x) + score_{setup}(x)$, i.e., $score_{setup}(x) = score_{final}(x) - score_{3DM}(x)$. We get the following:

- $score_{setup}(p) = \alpha + 2\gamma + 12k(\beta + G\gamma) = (12kG + 2)\gamma + 12k\beta + \alpha$,
- $score_{setup}(S_i) = 5k\beta + \min(\alpha, 2\gamma) + 2\gamma + (G+1)\alpha + 2G\beta = \min(\alpha, 2\gamma) + 2\gamma + (5k + 2G)\beta + (G+1)\alpha$,
- $score_{setup}(S_i') = 5k\beta + \alpha + \gamma + 2\gamma + 2G\beta = 3\gamma + \beta(5k + 2G) + \alpha$,
- $score_{setup}(c) = 2\gamma + 5k\beta + 3\alpha(G+1)$ for each $c \in X \cup Y \cup Z$,
- $score_{setup}(b_\heartsuit) = 2\gamma + \alpha + 5k(\beta + \heartsuit)$.

Clearly, it again suffices to realize the relative scores among the candidates (and clearly, the absolute points of all candidates will be at most 0, since we wrote f as having no strictly positive coefficient). Hence, it suffices to construct setup votes that for each candidate x, let x gain $score_{setup}(p) - score_{setup}(x)$ points *less than the preferred candidate* p; this is the number of points that the x must lose against p from the setup votes. For each x, we get the following value (clearly for p itself, the value is 0):

- $score_{lose}(S_i) = (12kG + 2)\gamma + 12k\beta + \alpha - (\min(\alpha, 2\gamma) + 2\gamma + (5k + 2G)\beta + (G+1)\alpha) = -\min(\alpha, 2\gamma) + 12kG\gamma + \beta(7k - 2G) - G\alpha$, note that this value can be made arbitrarily large if k is chosen sufficiently large, since G is a constant chosen depending on α, β, γ, but independent of the instance and therefore of k.
- $score_{lose}(S_i') = (12kG + 2)\gamma + 12k\beta + \alpha - (3\gamma + \beta(5k + 2G) + \alpha) = (12kG + 2)\gamma + 12k\beta + \alpha - 3\gamma - \beta(5k + 2G) - \alpha = \gamma(12kG - 1) + \beta(7k - 2G)$, again this value grows arbitrarily large in k.
- $score_{lose}(c) = (12kG + 2)\gamma + 12k\beta + \alpha - (2\gamma + 5k\beta + 3\alpha(G + 1)) = (12kG + 2)\gamma + 12k\beta + \alpha - 2\gamma - 5k\beta - 3\alpha(G + 1) = 12kG\gamma + 7\beta + \alpha(-3G - 2)$, again the value grows arbitrarily large in k.
- $score_{lose}(b_\heartsuit) = (12kG + 2)\gamma + 12k\beta + \alpha - (2\gamma + \alpha + 5k(\beta + \heartsuit)) = (12kG + 2)\gamma + 12k\beta + \alpha - 2\gamma - \alpha - 5k(\beta + \heartsuit) = 12kG\gamma + (7k)\beta - 5k\heartsuit = k(12G\gamma + 7\beta - 5\heartsuit)$.
  Note that, for a suitable choice of G, this value also grows arbitrarily for increasing k.

Hence, all candidates must lose points against p, and the number of points they must lose grows arbitrarily in k. Therefore, the points can be implemented using setup votes letting a candidate x lose α, β, or γ points against all other relevant candidates. The controller will not delete these votes, since they have p in one of the top positions.

Note that, since b_α, b_β and b_γ must lose $5k(\alpha + \beta + \gamma)$ points against p even if p gains $5k\beta$ points, it follows that these candidates must take all relevant positions in the manipulation votes, and, if $\beta > \gamma$, then p can only win if p indeed gains $5k\beta$ points from the deletions, i.e., only if only non-G 3DM votes are deleted. Therefore, for the remainder of the proof, assume that $\gamma = \beta$.

It remains to show that the controller will in fact delete only the non-G-3DM votes. Therefore, assume that there is a successful bribery action in which at least one of the G-votes is also deleted. If for every G-vote v_G that is removed in the bribery action, the corresponding non-G vote v_{3DM} (obtained from the G-vote by swapping the $-\gamma$ and $-\beta$ positions) is not removed, then a bribery action deleting only non-G-3DM votes can be obtained by deleting v_{3DM} instead of v_G for every relevant vote (the effect for p is at least as good when deleting v_{3DM}). Therefore, we can without loss of generality assume that there is a successful bribery action in which there is a non-G 3DM vote v_{3DM} such that both v_{3DM} and the corresponding G-vote v_G are deleted. We show that in this case, p cannot win the election. To see this, first note that p gains at most $5k\beta$ points from the delete actions. Therefore, after the bribery action, p's points are at most (recall that we can assume $\beta = \gamma$)

$$score_{max}(p) \leq \alpha + 2\gamma + (5k - 1)\beta + \gamma = \alpha + (5k + 2)\beta.$$

We now make a case distinction depending on which type of G-vote is deleted.

- First assume that v_G has a candidate $c \in X \cup Y \cup Z$ in the last position. Then the same is true for v_{3DM}. Since both votes are deleted, c gains at least 2α points (and note that c cannot lose points from deleting other votes). Therefore, c has at least $5k\beta + 2\gamma + 2\alpha = (5k + 2)\beta + 2\alpha$ points, which is strictly more than $score_{max}(p)$.
- Now assume that v_G has a candidate S_i in the last position. Analogously to the above, S_i then gains at least 2α points, and hence ends up with at least (recall that $\beta = \gamma$) $5k\beta + \min(\alpha, 2\gamma) + 2\alpha \geq 5k\beta + 2\gamma + 2\alpha = (5k + 2)\beta + 2\alpha$ points, which again is strictly more than $score_{max}(p)$.

Therefore, in both cases p does not win the election and we have a contradiction. Therefore, if the bribery instance is positive, then there exists a successful bribe in which only non-G 3DM votes are deleted, as required. □

6 OPEN QUESTIONS

The main open question is to completely characterize the complexity of f-manipulation, for all generators f. As discussed at the end of Section 2, we conjecture that this will not be a dichotomy theorem. As a first step, we would like to prove that the cases listed in Theorem 2.1 are exactly the polynomial-time cases (under some reasonable complexity-theoretic assumptions) or to construct an explicit counterexample to this statement.

Other interesting avenues to pursue are going beyond NP-completeness, by looking at such issues as fixed-parameter tractability (see, e.g., [11]), approximability (see e.g., [10]), and experimental results (see, e.g., [20] and [18]).

ACKNOWLEDGEMENTS

We thank the referees for helpful comments and suggestions.

REFERENCES

[1] J. Bartholdi, III, C. Tovey, and M. Trick, 'The computational difficulty of manipulating an election', *Social Choice and Welfare*, **6**(3), 227–241, (1989).

[2] J. Bartholdi, III, C. Tovey, and M. Trick, 'How hard is it to control an election?', *Mathematical and Computer Modeling*, **16**(8/9), 27–40, (1992).

[3] D. Baumeister and J. Rothe, 'Taking the final step to a full dichotomy of the possible winner problem in pure scoring rules', *Information Processing Letters*, **112**(5), 186–190, (2012).
[4] N. Betzler and B. Dorn, 'Towards a dichotomy of finding possible winners in elections based on scoring rules', *J. of Computer and System Sciences*, **76**(8), 812–836, (2010).
[5] N. Betzler, R. Niedermeier, and G. Woeginger, 'Unweighted coalitional manipulation under the Borda rule is NP-hard', in *Proceedings of the 22st International Joint Conference on Artificial Intelligence*, pp. 55–60. AAAI Press, (July 2011).
[6] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and G. Woeginger, 'Some simple scoring voting rules that are hard to bribe'. Manuscript, 2012.
[7] V. Conitzer, T. Sandholm, and J. Lang, 'When are elections with few candidates hard to manipulate?', *J. of the ACM*, **54**(3), Article 14, (2007).
[8] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh, 'Complexity of and algorithms for Borda manipulation', in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 657–662. AAAI Press, (August 2011).
[9] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra, 'How hard is bribery in elections?', *J. of Artificial Intelligence Research*, **35**, 485–532, (2009).
[10] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra, 'Weighted electoral control', *J. Artif. Intell. Res. (JAIR)*, **52**, 507–542, (2015).
[11] P. Faliszewski and R. Niedermeier, 'Parameterization in computational social choice', in *Encyclopedia of Algorithms*, ed., M-Y. Kao, Springer, (2015).
[12] P. Hall, 'On representatives of subsets', *J. London Math. Soc.*, **10**(1), 26–30, (1935).
[13] E. Hemaspaandra and L. Hemaspaandra, 'Dichotomy for voting systems', *J. of Computer and System Sciences*, **73**(1), 73–83, (2007).
[14] E. Hemaspaandra, L. Hemaspaandra, and H. Schnoor, 'A control dichotomy for pure scoring rules', in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pp. 712–720. AAAI Press, (2014).
[15] Edith Hemaspaandra and Henning Schnoor, 'Complexity dichotomies for unweighted scoring rules', *CoRR*, **abs/1604.05264**, (2016).
[16] R. Ladner, 'On the structure of polynomial-time reducibility', *J. of the ACM*, **22**, 155–171, (1975).
[17] A. Lin, *Solving Hard Problems in Election Systems*, Ph.D. dissertation, Rochester Institute of Technology, Rochester, NY, 2012.
[18] J. Rothe and L. Schend, 'Control complexity in Bucklin, Fallback, and Plurality voting: An experimental approach', in *Proceedings of the 11th International Symposium on Experimental Algorithms*, pp. 356–368, (June 2012).
[19] T. J. Schaefer, 'The complexity of satisfiability problems', in *Proceedings 10th Symposium on Theory of Computing*, pp. 216–226. ACM Press, (1978).
[20] T. Walsh, 'Where are the hard manipulation problems?', *J. of Artificial Intelligence Research*, **42**, 1–29, (2011).
[21] M. Zuckerman, A. Procaccia, and J. Rosenschein, 'Algorithms for the coalitional manipulation problem', *Artificial Intelligence*, **173**(2), 392–412, (2009).