A Control Dichotomy for Pure Scoring Rules[1]

Edith Hemaspaandra
Dept. of Comp. Science
RIT
Rochester, NY 14623

Lane A. Hemaspaandra
Dept. of Comp. Science
University of Rochester
Rochester, NY 14627

Henning Schnoor
Institut für Informatik
Kiel University
24098 Kiel, Germany

Abstract

Scoring systems are an extremely important class of election systems. A length-m (so-called) scoring vector applies only to m-candidate elections. To handle general elections, one must use a family of vectors, one per length. The most elegant approach to making sure such families are "family-like" is the recently introduced notion of (polynomial-time uniform) pure scoring rules [BD10], where each scoring vector is obtained from its precursor by adding one new coefficient. We obtain the first dichotomy theorem for pure scoring rules for a control problem. In particular, for constructive control by adding voters (CCAV), we show that CCAV is solvable in polynomial time for k-approval with $k \leq 3$, k-veto with $k \leq 2$, every pure scoring rule in which only the two top-rated candidates gain nonzero scores, and a particular rule that is a "hybrid" of 1-approval and 1-veto. For all other pure scoring rules, CCAV is NP-complete. We also investigate the descriptive richness of different models for defining pure scoring rules, proving how more rule-generation time gives more rules, proving that rationals give more rules than do the natural numbers, and proving that some restrictions previously thought to be "w.l.o.g." in fact do lose generality.

1 Introduction

Elections give rise to a plethora of interesting questions in the social and political sciences, and have been extensively studied from a computer-science point of view in the last two decades. We study the *control problem*, in which the chair of an election (ab)uses her power to try to affect the election outcome. In this paper we focus on *constructive control by adding voters* (CCAV), i.e., where the chair tries to make her favorite candidate win by adding voters. Constructive control by adding voters is an extremely important control type, since it occurs in (political) practice very often. A standard example is the "Get-out-the-Vote" efforts of political parties, aimed at (supposed) supporters of those parties. However, we should also mention that, as has been pointed out for example in books by Riker [Rik86] and Taylor [Tay05], in modern politics issues of candidate introduction or removal (CCAC/CCDC) have become highly important; the case of Ralph Nader in two recent American elections vividly supports this point.

The computational complexity of CCAV and other forms of control was first studied by Bartholdi, Tovey, and Trick [BTT92], for plurality and (so-called) Condorcet elections. In this paper, we study the complexity of CCAV for *pure scoring rules*, an attractive class introduced by Betzler and Dorn [BD10] that contains many important voting systems. A scoring rule for an election with m candidates is defined by m coefficients $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. Each voter ranks the m candidates from her most favorite to her least favorite; a candidate gains α_i points from being in position i on that voter's ballot.

Well-known examples of families of scoring rules include the following. Borda Count for m candidates uses coefficients $m-1, m-2, \ldots, 1, 0$. k-approval uses coefficients

[1]Supported in part by NSF grants CCF-{0915792,1101452,1101479} and by COST Action IC1205. Work done in part while H. Schnoor visited RIT supported by an STSM grant of Cost Action IC1205. This paper will also be presented at the Twenty-Eighth Conference on Artificial Intelligence (AAAI-2014) [HHS14b].

$\underbrace{1,\dots,1}_{k},\underbrace{0,\dots,0}_{m-k}$; k-veto uses $\underbrace{1,\dots,1}_{m-k},\underbrace{0,\dots,0}_{k}$. Dowdall voting, used for Nauru's parliament, uses $1, \frac{1}{2}, \frac{1}{3}, \dots, \frac{1}{m}$.

The construction of the scoring vector for a specific number of candidates usually follows a natural pattern, as in the above examples. This leads to the definition of a "pure scoring rule." We discuss the notion of "purity" in detail in this paper. (Basically, it means that at each length we insert one entry into the previous length's vector; all our above examples are pure.)

There is a rich literature on computational aspects of scoring rules, e.g., dichotomy theorems on weighted manipulation [HH07], the possible winner problem [BD10, BR12], and bribery [FHH09], as well as results about specific voting systems [BNW11, DKNW11, FHH13].

In this paper, we provide the first complete investigation of the complexity of the unweighted CCAV problem for pure scoring rules. We prove a dichotomy theorem that gives a complete complexity-theoretic classification of that control problem for pure scoring rules. Our result is as follows.

It turns out that there are only 4 types of pure scoring rules for which CCAV is solvable in polynomial time:

1. k-approval for $k \leq 3$,
2. k-veto for $k \leq 2$,
3. every pure scoring rule in which only the two top-rated candidates receive a nonzero score,
4. a particular rule which is a "hybrid" of 1-approval and 1-veto: each voter awards her favorite candidate 1 point, and her least favorite candidate -1 point.

For every pure scoring rule that is not one of the above election systems, CCAV is NP-complete. The last rule mentioned above is particularly interesting for two reasons: First, it was the only one for which the complexity of the possible winner problem was left open in Betzler and Dorn [BD10]. Second, it is the only one for which polynomial-time solvability depends on the actual coefficients, and not only on the $<$-order of the values: While this election system is equal to the rule generated by coefficients $2, 1, \dots, 1, 0$, the rule generated by using coefficients $3, 1, \dots, 1, 0$ leads to an NP-complete CCAV-problem.

A key point in the proof of any dichotomy theorem is to cover all relevant cases. We thus base our dichotomy result on a study of the descriptional richness of definitions of scoring protocols. We examine how variations of key parameters as the abovementioned purity requirement, the complexity allowed to compute the coefficients, and the universe from which the coefficients may be chosen affect the set of election systems that can be represented. Interestingly, we discover that assumptions made previously in the literature, which were believed to be without loss of generality, in fact restrict the rules that can be expressed. Taking these results into account, we introduce a flexible purity condition that is more robust with respect to the abovementioned variations. We show that the new notion strictly generalizes pure scoring rules with integer scores and coincides with pure scoring rules with rational coefficients.

The paper is structured as follows. In Section 2, we give necessary preliminaries about the class of election systems that we study. In Section 3, we study the descriptive richness of definitions of positional scoring rules. Section 4 contains our main complexity result. Some proofs and discussions that are not included here due to space limitations can be found in this paper's technical report version [HHS14a].

2 Preliminaries

As is standard, given an m-component (so-called "scoring") vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ such that $\alpha_1 \geq \cdots \geq \alpha_m$, we define a so-called (m-candidate) "scoring system" election based on this by, for each voter (the voters vote by strict, linear orders over the candidates), giving α_i points to the voter's ith-most-favorite candidate. Whichever candidate(s) get the highest number of points, summed over all voters, are the winner(s). We refer to the α_is as "coefficients."

We will use the following very pure definition of pure scoring rules, which removes some of the additional assumptions that have been used in earlier work. Indeed, earlier work asserted that (at least some of) these assumptions were not restrictions; we will look at that issue anew below.

An election system $\mathcal{E}$ is a $\mathcal{T}$-GSR (generalized scoring rule) if there is a function f that on each input 0^m, $m \geq 1$, outputs an m-component scoring vector $\alpha^m = (\alpha_1^m, \ldots, \alpha_m^m)$ such that $\alpha_1^m \geq \ldots \geq \alpha_m^m$ and each α_j^i belongs to $\mathcal{T}$, and for each m, the winner set under $\mathcal{E}$ is exactly the winner set given by the scoring system using the scoring vector α^m. The notation 0^m denotes a string of m "0"s. Using this as the argument ensures that the generator's computation time is measured as a function of m, since its input length is exactly m. Since the votes are of size at least m each, this provides the natural, fair approach to framing FP-uniformity (as we will do below). We call f a generator for $\mathcal{E}$. While one can consider election systems based on an f that is not computable, in practice we want there to be an efficient algorithm computing f. This is expressed with different *uniformity* conditions. If $\mathcal{E}$ is a $\mathcal{T}$-GSR via some generator f that can be computed in a complexity class $\mathcal{F}$, then $\mathcal{E}$ is an $\mathcal{F}$-uniform-$\mathcal{T}$-GSR and f is an $\mathcal{F}$-uniform $\mathcal{T}$-generator. The values of $\mathcal{T}$ we will be interested in are the naturals $\mathbb{N}$, the nonnegative rationals $\mathbb{Q}_{\geq 0}$, the rationals $\mathbb{Q}$, and the integers $\mathbb{Z}$. Our most important value of $\mathcal{F}$ will be the polynomial-time functions, FP. When we do not state "nonuniform" or some specific uniformity, we always mean FP-uniform. When we put a name in boldface, it indicates all the elections that can be generated by a generator of the named sort, e.g., **FP-uniform-$\mathbb{Z}$-GSR** is the class of all polynomial-time-uniform generalized scoring rules with integer coefficients, a class first defined and discussed by Hemaspaandra and Hemaspaandra [HH07].

A $\mathcal{T}$-GSR (of whatever uniformity) is a $\mathcal{T}$-PSR (pure scoring rule) if it has a generator (of the same uniformity) f satisfying the following "purity" constraint: For each $m \geq 2$, there is a component of α^m that when deleted leaves exactly the vector α^{m-1}.

Throughout the paper we use the following observation, which notes that two scoring vectors, after being "normalized," differ if and only if they are capturing distinct election systems: An m-position scoring vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ (over $\mathbb{N}$) is normalized if $\alpha_m = 0$ and the greatest common divisor of its nonzero α_is is 1. Normalizing a given scoring vector (over $\mathbb{N}$ or $\mathbb{Z}$) is easily achievable in polynomial time: subtract α_m from each coefficient and then divide each coefficient by the gcd (computed using Euclid's gcd algorithm) of the nonzero thus-altered coefficients. The normalization of a scoring vector over $\mathbb{Q}$ (resp., $\mathbb{Q}_{\geq 0}$) is done by multiplying through by the lcm of the denominators of the nonzero coefficients, and then viewing that as a vector over $\mathbb{Z}$ (resp., $\mathbb{N}$) and normalizing it as above.

Proposition 2.1. *Let $m \geq 1$ and let α and α' be m-position scoring vectors over $\mathcal{T} \subseteq \mathbb{Q}$. Then α and α' have the same winner sets on each m-candidate election if and only if α and α' both have the same normalized version.*

The if direction basically follows from Observation 2.2 of Hemaspaandra and Hemaspaandra [HH07], as noted by Betzler and Dorn [BD10]. The only if follows by giving a construction that for any two unequal normalized scoring vectors constructs a vote set on which their winner sets differ. The construction works by "aligning" the vectors by multiplying each so that their first coefficients are equal, and then using a padding construction to ensure that only two candidates are crucial and that the winner sets can be distinguished by appropriately exploiting the first position at which the aligned vectors differ.

Generators f_1 and f_2 are *equivalent* if they generate the same election system. Due to Proposition 2.1, this is the case if and only if, for each length m, the *normalized* scoring vectors generated by f_1 and f_2 for m candidates are identical.

3 Descriptive Richness and PSRs

We now examine how amount of time used to generate PSRs, and the universe the PSR's coefficients are drawn from, affect the family of election rules that can be obtained. We also look at whether such a seemingly innocuous and standard assumption as having the last coefficient always being zero in fact loses generality; we'll see that it does lose generality, but in a way that can in part be papered over.

Proofs or proof sketches for each of this section's results, and additional discussion, can be found in our technical report version.

3.1 Generation Time Gives Descriptional Richness

Does more generation time give a richer class of pure scoring rules? A very tight time hierarchy can be achieved by a legal form of cheating. In particular, consider any (nice) time class that can for some $m \geq 3$ generate a scoring vector over $\mathbb{N}$ of the form $(\alpha_1, \underbrace{1, \ldots, 1}_{m-2}, 0)$ such that α_1 is so big that some other time class cannot generate this scoring vector (for example, because it simply doesn't have enough time to write down enough bits to get a number as large as α_1). Our vector is normalized, and by Proposition 2.1 this already is enough to allow us to argue that the two time classes differ in their winner sets. (This proof works for GSRs. And it works for PSRs if the "nice"ness of the class allows it to obey purity while employing the above approach—hardly an onerous requirement.) However, that claim simply uses the fact that more time can write more bits. A truly fair and far more interesting separation would show that one can with more time obtain more pure scoring rules in a way that does *not* depend on using coefficient lengths that simply cannot be produced by the weaker time class. In the dream case, all coefficients in fact would simply be 0 or 1, so there are no long coefficients in play at all. We in fact have achieved such a hierarchy theorem. Full time constructibility is a standard notion that most natural time functions satisfy and, as is standard, when we speak of a time function $T(m)$ by convention that is shorthand for $\max(m+1, \lceil T(m) \rceil)$ [HU79]. FDTIME$[g(\cdot)]$ denotes the functions computable in the given amount of deterministic time.

Theorem 3.1. *If $T_2(m)$ is a fully time-constructible function and $\limsup_{n\to\infty} \frac{T_1(m)\log T_1(m)}{T_2(m)} = 0$, then there is an election rule in* **FDTIME$[T_2(m)]$*-uniform-*$\{0,1\}$-PSR** *that is not in* **FDTIME$[T_1(m)]$*-uniform-*$\mathbb{Q}$-GSR**.

The log factor here is not surprising; this is the standard overhead it takes for a 2-tape Turing machine to simulate a multitape TM. What is surprising is that no additional factor of m is needed. Why might we expect such a factor? (We caution that the rest of this paragraph is intended mostly for those having familiarity with the diagonalization techniques used to prove hierarchy theorems.) In the context of a diagonalization construction (which is the basic technique used in the proof of Theorem 3.1), one might expect to (all counting against the overall time T_2 limit) at vector-length m have to "recreate" all the shorter vectors used in earlier diagonalizations to ensure that the length-m and length-$(m-1)$ vectors are related in a "pure" way. We however sidestep the need for that overhead by a "purity"-inducing trick: For each odd m our scoring vector will be of the form $1^{\lfloor m/2 \rfloor}0^{\lfloor m/2 \rfloor+1}$, and at each even length, we purely extend that to whichever of $1^{\lfloor m/2 \rfloor+1}0^{\lfloor m/2 \rfloor+1}$ or $1^{\lfloor m/2 \rfloor}0^{\lfloor m/2 \rfloor+2}$ diagonalizes against the T_1-time machine that is being diagonalized against (if this is an m when we have time to so diagonalize). Briefly, we lurch back to fixed, safe way-stations at every second length, and this removes the need to recompute our own history.

3.2 Coefficient Richness Gives Descriptional Richness

The richness and structure of the coefficient set for PSRs affects how broad a class of election rules can be captured, as shown by the following result. (Of course, trivially $\mathbb{N}$-**PSR** $\subseteq$ $\mathbb{Q}_{\geq 0}$-**PSR** $\cap$ $\mathbb{Z}$-**PSR** $\subseteq$ $\mathbb{Q}_{\geq 0}$-**PSR** $\cup$ $\mathbb{Z}$-**PSR** $\subseteq$ $\mathbb{Q}$-**PSR.**)

Theorem 3.2. $\mathbb{Q}_{\geq 0}$-**PSR** $\not\subseteq$ ***nonuniform*-$\mathbb{Z}$-PSR***,* $\mathbb{Z}$-**PSR** $\not\subseteq$ ***nonuniform*-$\mathbb{Q}_{\geq 0}$-PSR***, and* $\mathbb{Q}$-**PSR** $\not\subseteq$ ***nonuniform*-$\mathbb{Q}_{\geq 0}$-PSR** $\cup$ ***nonuniform*-$\mathbb{Z}$-PSR***.*

So for example, in pure scoring rules, (polynomial-time uniformly) using positive rationals cannot be simulated by naturals or integers, even nonuniformly. And in pure scoring rules, (polynomial-time uniformly) using integers cannot be simulated by naturals or positive rationals, even nonuniformly. One might think these claims are impossible, and that by normalizing one can go back and forth, but it is precisely the purity requirement that is making that sort of manipulation impossible—there is a price to purity, and it is showing itself here. (The final part of the theorem does not follow automatically from the first two parts plus the trivial observation before the theorem; just the weaker variant of that part in which $\cup$ is replaced with $\cap$ follows from those.) Note that enlarging the universe does not necessarily lead to a larger class of election systems: For example, requiring that coefficients are odd natural numbers gives the same set of election systems as merely requiring them to be natural numbers.

We mention a more flexible and highly attractive notion of purity that erases the differences just discussed. $\mathcal{T}$-FPSRs (flexible pure scoring rules), of whatever uniformity or nonuniformity, will be defined exactly the same way as $\mathcal{T}$-PSRs were defined, except the purity condition is changed to: For each $m \geq 2$, there is a component of α^m whose removal gives a scoring vector equivalent to α^{m-1}. Due to Proposition 2.1, this means that the relevant scoring vectors have the same normalization. We call such generators *flexible.* For *this* notion we have for the nonuniform case and the FP-uniform case (and most other nice cases), the following equality.

Theorem 3.3. $\mathbb{N}$-**FPSR** $=$ $\mathbb{Q}$-**FPSR** $=$ $\mathbb{Q}_{\geq 0}$-**FPSR** $=$ $\mathbb{Z}$-**FPSR** $=$ $\mathbb{Q}$-**PSR***.*

3.3 Having Smallest Coefficient of Zero Loses Generality—Slightly

Betzler and Dorn [BD10] in their definition of scoring rules require that at each m, we have $\alpha_m^m = 0$ (let us call this condition norm-0), and comment that this is not a restriction. We note that there are PSRs that cannot be generated by any pure scoring rule meeting that constraint. What is at issue here is a bit subtle: At each fixed length, the restriction is innocuous. But in the context of families that are bound by the purity constraint, the restriction loses generality. On the other hand, we will also note that each pure scoring rule has a generator that is "close" to meeting that constraint—it meets it at all but finitely many lengths.

Betzler and Dorn [BD10] also have a "gcd is 1" condition (although the phrasing is not crystal clear as to whether the gcd constraint applies to the union of all nonzero coefficients that occur over all lengths, or whether it must in fact apply separately at each length; the latter (which we call norm-gcd) would block the vector $(2, 0)$ but the former would not if the next step were, for example, $(3, 2, 0)$). However, the gcd issues also can be made to go away "almost everywhere." In particular, we have established the following.

Theorem 3.4. *There is an* FP*-uniform* PSR *that is not generated even by any nonuniform* $\mathrm{PSR}_{\textit{norm-0}}$ *generator. On the other hand, every* FP*-uniform (respectively, nonuniform)* PSR *is generated by a* FP*-uniform (respectively, nonuniform)* PSR *generator that for all but a finite number of m has the property that the last coefficient, α^m_m, is zero and the gcd of the nonzero coefficients in the length-m vector is one.*

Does the second sentence of the theorem imply that each PSR has all its vectors the same at each length (except for a finite number of exceptional lengths) as the vectors of some PSR that satisfies norm-0 and norm-gcd? The answer is actually "no." The somewhat subtle issue at play is that PSRs can generate vectors that no generator satisfying norm-0 and norm-gcd can ever generate, such as the family $(3, 2, \ldots, 2, 0)$. So one should not from our above theorem claim that it follows from the main dichotomy *theorem* of Betzler and Dorn [BD10], as completed by Baumeister and Rothe [BR12], that we can read off the complexity (of the possible winner problem) even in our slightly more flexible case. However, our above theorem does—in light of the actual *proof case decomposition* used in those papers (which is based on issues such as whether one has an unbounded number of positions that differ and so on) and some additional argumentation to connect to that and in particular to note that Betzler and Dorn (and Baumeister and Rothe) are in effect quietly covering well even those cases that do not satisfy gcd constraints—connect so well to their work that each of our cases is settled by their proofs.

4 A Control Dichotomy for PSRs

We study the following problem for an election system $\mathcal{E}$: When R is a set of registered voters, is there some subset of the unregistered voters U of size at most k that we can add to the election to ensure that p is the winner?

Definition 4.1. *Let $\mathcal{E}$ be an election system. The* constructive control problem for $\mathcal{E}$ by adding voters*, $\mathcal{E}$-CCAV, is the following problem: Given two multisets sets of votes R and U, a candidate p and a number k, is there a set $A \subseteq U$ with $\|A\| \leq k$ such that p is a winner of the election if the votes in the multiset $R \cup A$ are evaluated using the system $\mathcal{E}$?*

We often use a generator, f, as a shorthand for the election system (scoring rule family) it generates, e.g., we write f-CCAV. For a generator f, we use $\alpha^{f,m} = (\alpha^{f,m}_1, \ldots, \alpha^{f,m}_m)$ to denote the scoring vector generated by f for m candidates and its individual coefficients. To simplify presentation, we only consider FP-uniform generators. However our results continue to hold as long as we can solve the following question in polynomial time: Given m, i, and j in *unary*, does $\alpha^m_i > \alpha^m_j$ hold, where $f(m) = (\alpha^m_1, \ldots, \alpha^m_m)$?

Our main result is a complexity dichotomy for f-CCAV when f is an FP-uniform pure $\mathbb{Q}$-generator (or, equivalently due to Theorem 3.3, a FP-uniform flexible $\mathbb{N}$-generator). Recall that equivalent generators result in the same election system, hence, due to Proposition 2.1, Theorem 4.2 implies polynomial-time results for all generators with the same normalization as one below. We state our main result.

Theorem 4.2. *Let f be a* FP*-uniform pure $\mathbb{Q}$-generator. Then f-CCAV is solvable in polynomial time if f is equivalent to one of the following generators:*

- $f_1 = (1, 1, 1, 0, \ldots, 0)$ *(this generates* 3*-approval),*

- *$f_2 = (1, \dots, 1, 0)$ or $f_3 = (1, \dots, 1, 0, 0)$ (1/2-veto),*
- *for some $\alpha \geq \beta$, $f_4 = (\alpha, \beta, 0, \dots, 0)$,*
- *$f_5 = (2, 1, \dots, 1, 0)$.*

In all other cases, f-CCAV is NP-complete.

Note that 1- and 2-approval are covered by the generator f_4. The remainder of the paper contains the proof of Theorem 4.2: Section 4.1 contains the algorithms for all polynomial-time solvable cases, Section 4.2 contains our hardness results, and the proof of Theorem 4.2 follows in Section 4.3.

4.1 Polynomial-Time Results

The following result is proven by Lin [Lin12].

Theorem 4.3. *$\mathcal{E}$-CCAV is solvable in polynomial time if $\mathcal{E}$ is k-approval with $k \leq 3$ or k-veto with $k \leq 2$.*

Due to Proposition 2.1, Theorem 4.3 implies that CCAV remains polynomial-time solvable for "scaled" versions of k-approval with $k \leq 3$ or k-veto with $k \leq 2$, i.e., generators of the form $(\alpha, \beta, \gamma, \delta, \dots, \delta)$ with $\beta, \gamma \in \{\alpha, \delta\}$ or $(\alpha, \dots, \alpha, \beta, \gamma)$ with $\beta \in \{\alpha, \gamma\}$. We now look at a generalization of 2-approval: Voters approve of 2 candidates, and can distinguish between their first and second choice. CCAV for this generalization remains efficiently solvable. In contrast, Theorem 4.11 shows that our control problem for the corresponding generalization of 3-approval is NP-hard.

Theorem 4.4. *Let $\alpha \geq \beta$ be fixed. Then f-CCAV is polynomial-time solvable for $f = (\alpha, \beta, 0, \dots, 0)$.*

Proof. Due to Theorem 4.3, assume $\alpha > \beta$. Let ℓ be the smallest natural number such that $\ell(\alpha - \beta) \geq \beta$. Let an instance with candidates C, favorite candidate p, registered voters R, and potential voters U be given; let k be the number of voters that can be added. Assume $\ell \leq k$, otherwise brute-force. Let V_1 (V_2) be the set of voters in U that put p in the first (second) spot. Clearly, we add voters only from $V_1 \cup V_2$. Assume w.l.o.g. that two voters who vote identically in the first two positions also rank the remaining candidates identically. In particular, two voters in V_1 (V_2) are different if and only if they vote different candidates in the second (first) place. We use the following facts (whose proofs can be found in our technical report version).

Fact 1. *For $i \in \{1, 2\}$, given a set $S \subseteq V_1 \cup V_2$, it can be checked in polynomial time whether S can be extended, by adding at most $k - \|S\|$ voters from V_i to make p win.*

Fact 2. *To make p win, it is never better to add ℓ voters from V_2 than adding ℓ pairwise different voters from V_1.*

Due to Fact 2, we do not have to consider solutions that use ℓ or more voters from V_2 and leave ℓ or more distinct voters from V_1 unused. So if a solution exists, we can find one using fewer than ℓ voters from V_2, or leaving fewer than ℓ pairwise different voters from V_1 unused. For both cases we will test whether there is a corresponding solution.

We start with the first case. Since ℓ is constant, we can test every subset $S \subseteq V_2$ with $\|S\| < \ell$. For each of these S, we use Fact 1 to check in polynomial time whether S can be extended to a solution by adding voters only from V_1.

For the second case, we determine in polynomial time whether there is a solution that does not leave ℓ pairwise different voters from V_1 unused as follows: We encode the choice of the unused voters from V_1 as a function $u\colon C \to \{0, \dots, \|V_1\|\}$ that states, for each candidate c, the number of unused V_1-voters placing c second. Since we look for solutions satisfying the second case, we only consider functions u for which $u(c) > 0$ for at most $\ell - 1$ candidates.

Thus we can brute-force over all of these possibilities as follows:

1. In the outer loop, we test every subset $S \subseteq C$ with $\|S\| \leq \ell - 1$. Since ℓ is a constant, there are only polynomially many of these sets.

2. For each such S, we test all functions u as above for which $u(c) > 0$ holds iff $c \in S$. Such a function can be regarded as a function $u\colon S \to \{1, \dots, \|V_1\|\}$. There are $\|V_1\|^{\|S\|} \leq \|V_1\|^{\ell-1}$ many of these functions. Since $\|V_1\|$ is bound by the input size and ℓ is a constant, this number is polynomial in the input size.

So we can polynomially go through all possibilities of potentially unused V_1-voters, which is the same as going through all possible sets S' of *used* V_1-voters. For each of these sets S', we again use Fact 1 to check in polynomial time whether S' can be extended to a solution. This completes the proof. $\square$

Our final polynomial-time case is the generator $(2, 1, \dots, 1, 0)$. Here every voter "approves" one candidate and "vetoes" another. This case is interesting for two reasons. First, it is the only case where the algorithm depends on the coefficients itself, as opposed to their $>$-order. Namely, for all $\alpha > \beta > 0$ with $\alpha \neq 2\beta$, f-CCAV with $f = (\alpha, \beta, \dots, \beta, 0)$ is NP-complete (Theorem 4.12.2). Second, this case was the only one left open in Betzler and Dorn's [BD10] possible winner dichotomy; the question was eventually settled by Baumeister and Rothe [BR12], who proved NP-completeness.

Theorem 4.5. *f-CCAV is solvable in polynomial time for the generator $f = (2, 1, \dots, 1, 0)$.*

Proof. Let C, R, and U be the set of candidates, registered voters, and unregistered voters, p the preferred candidate, and k the number of voters we can add. We add no voter voting p last, and it is never better to add a voter voting p second than to add one voting p first. So we first add all voters from U that place p in the first position. If there are more than k of these voters, we choose the ones to add with the obvious greedy strategy that always picks, among all available votes of the form $p > \cdots > c$, the one where c currently has the highest score. After this preprocessing, all relevant voters in U vote $c_1 > \cdots > c_2$ with $p \notin \{c_1, c_2\}$. To simplify presentation, we use Proposition 2.1 and consider f as the generator $(1, 0, \dots, 0, -1)$. Then the score of p is determined by the votes in R.

We reduce the problem to min-cost (network) flow, which can be solved in polynomial time. Let $S = \sum_{c \in C - \{p\}} \mathit{score}(c)$. We use the following nodes and edges:

- For each $c \in C - \{p\}$, there is a node c, additionally, there are source and target nodes s and t.

- There is an edge from candidate c_1 to candidate c_2 with cost 1 and with capacity equal to the number of voters in U voting $c_2 > \cdots > c_1$.

- For each candidate-node c, there is an edge from s to c with cost 0 and capacity $\mathit{score}(c)$ and an edge from c to t with cost 0 and capacity $\mathit{score}(p)$.

Now p can be made winner with at most k additional voters if and only if there is a flow from s to t with value S and cost at most k: Clearly, network flows with cost at most k correspond to subsets of U with size at most k, and using an edge (c_1, c_2) r times corresponds to adding r voters voting $c_2 > \cdots > c_1$, since this vote transfers one point from c_1 to c_2. The capacity of the outgoing edges of s ensure that each candidate initially gets the correct number of points (since S points must be distributed), the edges to t ensure that in the end, no candidate may have more points than p. □

The above results cover all polynomial-time cases of Theorem 4.2. We now turn to the NP-complete cases.

4.2 Hardness Results

We use the standard NP-complete problem 3DM (3-dimensional matching).

Definition 4.6. *3DM is defined as follows:*

Input	*Pairwise disjoint sets X, Y, and Z with $\|X\| = \|Y\| = \|Z\|$, and a set $M \subseteq X \times Y \times Z$.*
Question	*Is there a set $C \subseteq M$ with $\|C\| = \|X\|$ that covers X, Y, and Z?*

We say that C covers X (resp., Y, Z) if every element from X (resp., Y, Z) appears in a tuple of C. Since X, Y, and Z are pairwise disjoint, in this case every element from X (Y, Z) appears in the first (second, third) component of a tuple from C. Since $\|X\| = \|Y\| = \|Z\|$, a set $C \subseteq M$ with $\|C\| = \|X\|$ covers X (Y, Z) if and only if no two tuples from C agree in the first (second, third) component. A set C covering X, Y, and Z is called a *cover*.

Constructing Elections

In our hardness proofs, we often need to set up the registered voters to ensure specific scores for the candidates. The following lemma (whose proof can be found in our technical report version) shows that, if there is a "dummy" candidate to whom any surplus points can be "shifted," we can obtain every set of relative scores that can be expressed as a polynomial-size linear combination of the coefficients in the scoring vector.

Lemma 4.7. *Given a scoring vector $(\alpha_1, \ldots, \alpha_m)$, and for each $c \in \{1, \ldots, m-1\}$, numbers $a_1^c, \ldots, a_m^c$ in signed unary, and a number k in unary, we can compute, in polynomial time, votes such that the scores of the candidates when evaluating these votes according to the scoring vector $(\alpha_1, \ldots, \alpha_m)$ are as follows: There is some o such that for each $c \in \{1, \ldots, m-1\}$, $score(c) = o + \sum_{i=1}^{m} a_i^c \alpha_i$, and $score(c) > score(m) + k\alpha_1$.*

The value o in Lemma 4.7 is the common offset for all relevant scores. The actual value of o is irrelevant, since the winner of the election is determined by the relative scores. The value k is given so that the computed votes ensure that the dummy candidate m cannot win the election with the addition of at most k voters.

"Many" Different Coefficients

We now show that the CCAV-problem is NP-complete for generators using "many" different coefficients. Consider any generator f using (at least) 7 different coefficients for some length m. Then with $\alpha^{f,m} = (\alpha_1^{f,m}, \alpha_2^{f,m}, \alpha_3^{f,m}, \alpha_4^{f,m}, \ldots, \alpha_{m-2}^{f,m}, \alpha_{m-1}^{f,m}, \alpha_m^{f,m})$ we know that $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$. This condition in fact suffices for the CCAV problem to be NP-hard; the result applies to, e.g., Borda, 3-veto, and 4-approval (the latter two use just two different coefficients, but satisfy $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$ for $m \geq 7$).

For 4-approval or 3-veto, NP-hardness can be proven by positioning the elements of M from a 3DM-instance, along with p, in the 4 top positions of an unregistered 4-approval vote or (without p) in the last 3 positions of an unregistered 3-veto vote. In our cases, we can always "simulate" one of these systems: If $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$, then being ranked in one of the first 4 positions is strictly better than being ranked in one of the last 3 positions. Roughly speaking, if "many" intermediate coefficients are larger than the last 3, then the last 3 are the "exception," and we can use them to "simulate" 3-veto. On the other hand, if "many" intermediate coefficients are smaller than the first 4, then the first 4 are the "exception" and we "simulate" 4-approval.[2] NP-hardness for both 3-veto and 4-approval is proved by Lin [Lin12]; however we use a direct reduction from 3DM in our generalization.

We start with the "simulation" of 3-veto. The statement of the following result is a bit unusual. It indeed gives a reduction for generators meeting the condition $\alpha_{3k+1}^{f,m} > \alpha_{m-2}^{f,m}$ for all m. But beyond that the function g gives what we call a "partial" reduction from 3DM to f-CCAV for fs that meet the condition for some values of m. In the proof, the size of the 3DM instance is artificially enlarged to ensure that this "partial reduction" meets an analogue counterpart in such a way that for every generator f that satisfies $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$ for *some* m, we know that for *each* large enough m, one of the two reductions can be applied. (Our technical report version provides additional discussion of this.)

Theorem 4.8. *Let f be an* FP*-uniform $\mathbb{Q}$-generator. Then there exists an* FP*-computable function g such that*

- *g takes as input an instance I_{3DM} of 3DM and produces an instance I_{CCAV} of f-CCAV with $m = 6k$ candidates, where $k = \|X\| = \|Y\| = \|Z\|$.*
- *If $\alpha_{3k+1}^{f,m} > \alpha_{m-2}^{f,m}$, then: I_{3DM} is a positive instance of 3DM iff I_{CCAV} is a positive instance of f-CCAV.*

Proof. We write α_i for $\alpha_i^{f,m}$. W.l.o.g., let $X = \{s_1, \ldots, s_k\}$, $Y = \{s_{k+1}, \ldots, s_{2k}\}$, and $Z = \{s_{2k+1}, \ldots, s_{3k}\}$. We use the following candidates:

- Each $s_i \in \{s_1, \ldots, s_{3k}\}$ is a candidate.
- p is the preferred candidate.
- There are dummy candidates $d_1, \ldots, d_{m-3k-1}$. We assume there are at least 3 dummy candidates, i.e., $k \geq 2$.

We now use Lemma 4.7 to construct the set R of registered voters such that the scores of the candidates are as follows. (In the following, we "normalize" the scores of all candidates using the score of p as a base. So we pretend that the number o from the application of Lemma 4.7 is zero in order to simplify the presentation, clearly the absolute points of all candidates must be positive and are shifted by the actual number o from the lemma.)

- $\mathit{score}(p) = 0$.
- $\mathit{score}(s_i) = k\alpha_1 - (k-1)\alpha_{1+i} - \alpha_{m-r(i)}$, where $r(i) = 2, 1, 0$ depending on whether $s_i \in X, Y, Z$, respectively. (So $\alpha_{m-r(i)}$ is exactly the amount of points that s_i gains from a vote that "vetoes" s_i, see below)
- $\mathit{score}(d_i) < -k\alpha_1$ for $i \in \{1, \ldots, d_{m-3k-1}\}$.

[2] For generators where both cases apply such as $f = (2, 2, 2, 2, 1, 1, 1, \ldots, 1, 0, 0, 0)$, either reduction works.

Let $M \subseteq X \times Y \times Z$ be the set from I_{3DM}. For each $(x, y, z) = (s_h, s_i, s_j) \in M$ (so clearly $h < i < j$), we add an available voter to U voting as follows:

$p > s_1 > \cdots > s_{h-1} > d_1 > s_{h+1} > \cdots > s_{i-1} >$
$d_2 > s_{i+1} > \cdots > s_{j-1} > d_3 > s_{j+1} > \cdots > s_{3k} >$
$d_4 > \cdots > d_{m-3k-1} > x > y > z.$

We say that such a vote *vetoes* the candidates x, y, and z, and identify elements of M and the corresponding votes.

We show that the reduction is correct. First assume that the instance of 3DM is positive, and let $C \subseteq M$ be the cover with $\|C\| = k$. We add the voters corresponding to the elements of C in the obvious way and show that p indeed wins the resulting election.

To see this, it suffices to show that p has at least as many points as each candidate s_i, since by construction, the dummy candidates cannot win the election with adding at most k votes. So let $i \in \{1, \dots, 3k\}$. The final score for p and s_i are as follows:

- p gains α_1 points in each of the k additional votes, so p ends up with exactly $k\alpha_1$ points.
- s_i gains $(k-1)\alpha_{1+i}$ points from the $(k-1)$ votes corresponding to elements $(x, y, z) \in C$ with $s_i \notin \{x, y, z\}$, and $\alpha_{m-r(i)}$ points from the single vote vetoing s_i. So s_i ends up with a final score of $k\alpha_1 - (k-1)\alpha_{1+i} - \alpha_{m-r(i)} + (k-1)\alpha_{1+i} + \alpha_{m-r(i)} = k\alpha_1$ as well.

For the converse, let $C \subseteq M$ be a set of at most k votes whose addition lets p win. If this is not a cover, then there is some s_i that is vetoed in none of the added votes. We now compare the points of p and s_i.

- p gains α_1 points in each of the $\|C\|$ additional votes, so p ends up with exactly $\|C\|\alpha_1$ points.
- Since s_i is not vetoed in any new vote, s_i gains α_{1+i} points in each added vote and thus ends up with $k\alpha_1 - (k-1)\alpha_{1+i} - \alpha_{m-r(i)} + \|C\|\alpha_{1+i}$ points.

Since $\|C\| \leq k$, $\alpha_1 \geq \alpha_{i+1}$ and $\alpha_{i+1} \geq \alpha_{3k+1} > \alpha_{m-2} \geq \alpha_{m-r(i)}$, it follows that $k\alpha_1 - (k-1)\alpha_{1+i} - \alpha_{m-r(i)} + \|C\|\alpha_{1+i} - \|C\|\alpha_1$
$= \underbrace{(k - \|C\|)(\alpha_1 - \alpha_{1+i})}_{\geq 0} + \underbrace{\alpha_{1+i} - \alpha_{m-r(i)}}_{>0} > 0$. So s_i beats p if C is not a cover; since by assumption adding C makes p win, C must be a cover. □

In a similar way, we can prove an analogous result for all scoring rules that "can implement" 4-approval in the sense that being voted in one of the first 4 positions is strictly better than being voted in most "later" positions. The proof of the following result is very similar to the proof of Theorem 4.8, except that an additional argument is needed to ensure that the favorite candidate cannot be made a winner with less than k additional voters. The proof can be found in our technical report version.

Theorem 4.9. *Theorem 4.8 also holds when the condition $\alpha_{k+1}^{f,m} > \alpha_{m-2}^{f,m}$ is replaced with $\alpha_4^{f,m} > \alpha_{m-3k+1}^{f,m}$.*

As mentioned above, we now put the two reductions above together to obtain the NP-hardness result of this section, i.e., to prove that f-CCAV is NP-complete as soon as there is a number m where the coefficients of f satisfy $\alpha_4^{f,m} > \alpha_{n-2}^{f,m}$. If this condition is true, then we know that one of the inequalities $\alpha_4^{f,m} \geq \alpha_5^{f,m} \geq \cdots \geq \alpha_{m-3}^{f,m} \geq \alpha_{m-2}^{f,m}$ is in fact strict. Depending on the position of this strict inequality, we choose which reduction to apply: If the strict inequality appears "close" to the first candidate, then the first "few" positions are strictly better than "most," and the system can "simulate" k-approval for some $k \geq 4$. On the other hand, if the strict inequality appears "close" to the last candidate, then the last "few" positions are worse than "most," and we can similarly "simulate" k-veto for some $k \geq 3$. The proof of the following theorem can be found in our technical report version.

Theorem 4.10. *f-CCAV is NP-complete for every* FP*-uniform pure $\mathbb{Q}$-generator f with $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$ for some m.*

"Few" Different Coefficients

We now study pure generators f *not* covered by Theorem 4.10, i.e., where $\alpha_4^{f,m} \leq \alpha_{m-2}^{f,m}$ for all m. Then for $m \geq 6$, $\alpha^{f,m}$ is of the form $(\alpha_1^{f,m}, \alpha_2^{f,m}, \alpha_3^{f,m}, \alpha_4^{f,m}, \dots, \alpha_4^{f,m}, \alpha_5^{f,m}, \alpha_6^{f,m})$. The reductions above cannot work in this case, since there are no 3 positions "worse than most" and no 4 positions "better than most."

Due to Theorem 3.3, we can regard f equivalently as flexible $\mathbb{N}$-generators or as pure $\mathbb{Q}$-generators. For the latter representation, purity requires that all coefficients from $\alpha^{f,m}$ also appear in $\alpha^{f,m+1}$. So the above numbers $\alpha_1, \dots, \alpha_6$ do not depend on m. We can use a fixed affine transformation for these finitely many coefficients and, using Proposition 2.1, rewrite all coefficients as natural numbers.

Our next hardness result concerns a generalization of 3-approval. Recall from Theorem 4.3 that CCAV for 3-approval itself, i.e., the generator $(\alpha, \alpha, \alpha, 0, \dots, 0)$, is solvable in polynomial time. In Theorem 4.4, we proved a generalization of 2-approval to still give a polynomial-time solvable CCAV-problem. We now show that the analogous generalization of 3-approval leads to NP-completeness.

Theorem 4.11. *Let $\alpha \geq \beta \geq \gamma > 0$ and $\alpha \neq \gamma$. Let f be the generator giving $(\alpha, \beta, \gamma, 0, \dots, 0)$. Then f-CCAV is NP-complete.*

Proof. Let M be the set from an instance of 3DM with $\|M\| = n$, and let $k = \|X\| = \|Y\| = \|Z\|$ (recall X, Y, and Z must be pairwise disjoint). We use the candidates $X \cup Y \cup Z \cup \{p\} \cup \{S_i, S_i' \mid S_i \in M\}$ and a dummy candidate d to be able to apply Lemma 4.7. We use the lemma to set up the registered votes such that the resulting relative scores are as follows: $\mathit{score}(p) = \alpha + 2\gamma$, $\mathit{score}(c) = (n + 2k)\beta + 2\gamma$ for all $c \in X \cup Y \cup Z$, $\mathit{score}(S_i) = (n + 2k)\beta + \min(\alpha, 2\gamma)$, and $\mathit{score}(S_i') = (n + 2k)\beta + \alpha + \gamma$ for each $S_i \in M$. Further, $\mathit{score}(d) < -(n + 2k)\alpha_1$. For each $S_i = (x, y, z)$, we introduce four unregistered voters voting as follows:

$x > p > S_i > \dots$.

$y > p > S_i > \dots$.

$z > p > S_i' > \dots$.

$S_i > p > S_i' > \dots$.

We show that p can be made a winner of the election by adding at most $n + 2k$ voters if and only if the 3DM-instance is positive, i.e., there is a set $C \subseteq M$ with $\|C\| = k$ and for $S_i \neq S_j \in C$, S_i and S_j differ in all three components.

First assume that there is such a cover. In this case, p can be made a winner of the election by adding the following voters: For each $S_i = (x, y, z) \in C$, we add the votes $x > p > S_i$, $y > p > S_i$, and $z > p > S_i'$. For each $S_i = (x, y, z) \notin I$, we add the vote $S_i > p > S_i'$. Note that this adds exactly $3k + (n - k) = n + 2k$ votes. Adding these votes results in the following scores:

- p gains β points in each added vote, so p gains $(n + 2k)\beta$ points and p's final score is $\alpha + 2\gamma + (n + 2k)\beta$,

- each candidate in $X \cup Y \cup Z$ gains α points, leading to a final score of $\alpha + \underbrace{(n+2k)\beta + 2\gamma}_{\text{previous score}}$ as well,
- for each $S_i \in C$, we have that $\mathit{score}(S_i) = \underbrace{(n+2k)\beta + \min(\alpha, 2\gamma)}_{\text{previous score}} + 2\gamma \leq (n+2k)\beta + \alpha + 2\gamma$, which again is the score of p.
- for each $S_i \notin C$, we have $\mathit{score}(S_i) = (n+2k)\beta + \min(\alpha, 2\gamma) + \alpha \leq (n+2k)\beta + 2\gamma + \alpha$, equal to the score of p.
- for each S_i' (independent of whether $S_i' \in C$ or $S_i' \notin C$), we have $\mathit{score}(S_i') = \underbrace{(n+2k)\beta + \alpha + \gamma}_{\text{previous score}} + \gamma = (n+2k)\beta + \alpha + 2\gamma$, again this is the score of p.

Thus all candidates tie and so in particular, p is a winner of the election.

For the converse, assume that p can be made a winner by adding at most $n + 2k$ voters. Since each S_i' initially beats p, at least one vote is added. Thus there is a candidate $c \in X \cup Y \cup Z$ with $\mathit{score}(c) \geq (n+2k)\beta + 2\gamma + \alpha$, or some S_i' with $\mathit{score}(S_i') \geq (n+2k)\beta + \alpha + 2\gamma$. In both cases, we need to add at least $n+2k$ voters to ensure that p has at least $\alpha + 2\gamma + (n+2k)\beta$ points as well.

Since $n + 2k$ votes are added, and each of these votes gives points to p and 2 other candidates, there are $2n + 4k$ positions awarding points in the added votes that are filled with (not necessarily different) candidates other than p. Each of the $3k$ candidates from $X \cup Y \cup Z$ can only gain α points without beating p in the election, so each of these can fill at most one of these $2n + 4k$ positions. So at least $2n + k$ positions must be filled by (again, not necessarily different) candidates from $\{S_i, S_i' \mid 1 \leq i \leq n\}$. Each S_i' can appear at most once in the third position without beating p. Since there are n candidates of the form S_i', it follows that there must be $n + k$ occurrences of candidates S_i in the first three positions of the added votes. Since no S_i can gain $\alpha + \gamma$ points without beating p,[3] each S_i can either appear in a vote $S_i > p > S_i'$, or in up to two votes of the form $c > p > S_i$ with $c \in X \cup Y$. ($S_i = (x, y, z)$ cannot appear in three of these, since then one of x and y would gain too many points.) So the only way to fill $n + k$ positions with candidates of the form S_i is having $2k$ occurrences of S_i in the third place, and $n - k$ occurrences of S_i in the first place. In order to fill all positions, each S_i' has to appear once in the final position, and due to the above, $n - k$ of these occurrences are in a vote of the form $S_i > p > S_i'$. Thus there are k votes of the form $z > p > S_i'$. It follows that there are $3k$ votes added that vote a candidate from $X \cup Y \cup Z$ in the first position, and $n - k$ voters are added that vote some S_i first. Since no S_i may appear both in first and in last position, and each S_i' may appear only once, and each x_i, y_i, and z_i may gain only α points, it follows that the added votes correspond to a cover. □

We also have proved the following cases NP-complete; the proofs can be found in our technical report version.

Theorem 4.12. *The problem f-CCAV is NP-complete if f is one of the following pure generators:*

1. $f = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \ldots, \alpha_4, \alpha_5, 0)$ with $\alpha_2 > \alpha_4 > 0$.

[3] To see this, we compute the difference between the score of S_i after gaining $\alpha + \gamma$ points and that of p after gaining $(n+2k)\beta$ points. This value is $(n+2k)\beta + \min(\alpha, 2\gamma) + \alpha + \gamma - \alpha - \gamma - (n+2k)\beta = \min(\alpha, 2\gamma) - \gamma$. Since $\alpha > \gamma$ and $\gamma > 0$, this value is strictly positive, so S_i indeed beats p if S_i gains $\alpha + \gamma$ points.

2. $f = (\alpha_1, \alpha_2, \ldots, \alpha_2, 0)$ *with* $\alpha_1 \notin \{\alpha_2, 2\alpha_2\}$, $\alpha_2 > 0$.

3. $f = (\alpha_1, \alpha_2, \ldots, \alpha_2, \alpha_5, 0)$ *with* $\alpha_1 > \alpha_2 > \alpha_5$.

4. $f = (\alpha_1, \ldots, \alpha_1, \alpha_5, 0)$ *with* $\alpha_1 > \alpha_5 > 0$.

4.3 Proof of Dichotomy Theorem

We now use the individual results from Sections 4.1 and 4.2 to prove our main dichotomy result, Theorem 4.2:

Proof. The polynomial cases follow from Theorems 4.3, 4.4, and 4.5, we prove hardness. If $\alpha_4^{f,m} > \alpha_{m-2}^{f,m}$ for some m, hardness follows from Theorem 4.10. So assume $\alpha_4^{f,m} = \cdots = \alpha_{m-2}^{f,m}$ for all $m \geq 6$. As argued in the discussion after Theorem 4.10, we assume $\alpha^{m,f} = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \ldots, \alpha_4, \alpha_5, \alpha_6)$ for each $m \geq 6$. Due to Proposition 2.1, we can assume $\alpha_6 = 0$. We reduce the number of relevant coefficients from 5 to 3:

- If $\alpha_4 = 0$, then, since f does not generate 3-approval and is not equivalent to f_4, $\alpha_1 > \alpha_3 > 0$. Hardness follows from Theorem 4.11.
- If $\alpha_2 > \alpha_4 > 0$, hardness follows from Theorem 4.12.1.

So assume $\alpha_2 = \alpha_3 = \alpha_4 > 0$, i.e., f is of the form $(\alpha_1, \alpha_2, \ldots, \alpha_2, \alpha_5, 0)$. We make a further case distinction:

- If $\alpha_2 = \alpha_5$, then since f does not generate 1-veto, we know that $\alpha_1 \neq \alpha_5 = \alpha_2$. Since f is not equivalent to $(2, 1, \ldots, 1, 0)$, we know that $\alpha_1 \neq 2\alpha_2$. Thus NP-hardness follows from Theorem 4.12.2.
- If $\alpha_2 > \alpha_5$, then depending on whether $\alpha_1 > \alpha_2 > \alpha_5$ or $\alpha_1 = \alpha_2 > \alpha_5$, hardness follows from Theorem 4.12.3 or Theorem 4.12.4 (note that in the latter case, we know that $\alpha_5 \neq 0$, since f does not generate 2-veto). □

Acknowledgments

We thank the AAAI 2014 and COMSOC 2014 reviewers for helpful comments and suggestions.

References

[BD10] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. *Journal of Computer and System Sciences*, 76(8):812–836, 2010.

[BNW11] 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*, pages 55–60. AAAI Press, July 2011.

[BR12] 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.

[BTT92] 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.

[DKNW11] 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*, pages 657–662. AAAI Press, August 2011.

[FHH09] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. How hard is bribery in elections? *Journal of Artificial Intelligence Research*, 35:485–532, 2009.

[FHH13] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Weighted electoral control. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems*, pages 367–374. International Foundation for Autonomous Agents and Multiagent Systems, May 2013.

[HH07] E. Hemaspaandra and L. Hemaspaandra. Dichotomy for voting systems. *Journal of Computer and System Sciences*, 73(1):73–83, 2007.

[HHS14a] E. Hemaspaandra, L. Hemaspaandra, and H. Schnoor. A control dichotomy for pure scoring rules. Technical Report arXiv:1404.4560 [cs.GT], Computing Research Repository, arXiv.org/corr/, April 2014.

[HHS14b] E. Hemaspaandra, L. Hemaspaandra, and H. Schnoor. A control dichotomy for pure scoring rules. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, July 2014. To appear.

[HU79] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[Lin12] A. Lin. *Solving Hard Problems in Election Systems.* PhD thesis, Rochester Institute of Technology, Rochester, NY, 2012.

[Rik86] W. Riker. *The Art of Political Manipulation.* Yale University Press, 1986.

[Tay05] A. Taylor. *Social Choice and the Mathematics of Manipulation.* Cambridge University Press, 2005.