Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

Possible Winners in Noisy Elections

Krzysztof Wojtas and **Piotr Faliszewski**
AGH University of Science and Technology
Krakow, Poland

Abstract

We consider the problem of predicting winners in elections given complete knowledge about all possible candidates, all possible voters (together with their preferences), but in the case where it is uncertain either which candidates exactly register for the election or which voters cast their votes. Under reasonable assumptions our problems reduce to counting variants of election control problems. We either give polynomial-time algorithms or prove #P-completeness results for counting variants of control by adding/deleting candidates/voters for Plurality, k-Approval, Approval, Condorcet, and Maximin voting rules.

Introduction

Predicting election winners is always an exciting activity: Who will be the new president? Will the company merge with another one? Will taxes be higher or lower? The goal of this paper is to establish the computational complexity of a family of problems modelling a certain type of winner-prediction problems.

Naturally, predicting winners is a hard task, full of uncertainties. For example, we typically are not sure which voters will eventually cast their votes or, sometimes, even which candidates will in fact participate in the election (consider, e.g., a candidate withdrawing due to personal reasons). Further, typically we do not have complete knowledge regarding each voters' preferences. Nonetheless, the problem of predicting election winners is far too important to be abandoned due to technical difficulties as elections are in everyday use both among humans and among software agents (see, e.g., (Ephrati and Rosenschein 1997; Ghosh et al. 1999; Dwork et al. 2001). People and software agents may wish to optimize their behavior in voting scenarios, and for that they need to compute good estimates of election winners.

In this paper, we focus on a variant of the winner-prediction problem where we have complete knowledge regarding all possible candidates and all eligible voters (including knowledge of their preferences[1]), but we are uncertain as to which candidates and which voters turn up for the actual election (see "Related Work" section for other approaches to the problem). However, modelling uncertainty regarding both the candidate set and the voter collection on one hand almost immediately leads to computationally hard problems for typical election systems, and on the other hand does not seem to be as well motivated as focusing on each of these sets separately. Thus, we consider the following two settings:

1. The set of candidates is fixed, but for each possible subset of voters we are given a probability that exactly these voters show up for the vote.
2. The set of voters is fixed, but for each possible subset of candidates we are given a probability that exactly these candidates register for the election.

The former setting, in particular, corresponds to political elections (e.g., to presidential elections), where the candidate set is typically fixed well in advance due to election rules, the set of all possible voters (i.e., the set of all citizens eligible to vote) is known, but it is not clear as to which citizens choose to cast their votes. The latter setting may occur, for example, if one considers agents voting on a joint plan. The set of agents participating in the election is typically fixed, but various variants of the plan can be put forward or dismissed dynamically. In either case, our goal is to compute each candidate's probability of victory.

However, our task would very quickly become computationally prohibitive (or, difficult to represent on a computer) if we allowed arbitrary probability distributions. Thus, we have to choose some restriction on the distributions we consider. Let us consider the following example.

Let the set C of candidates participating in the election be fixed (for example, because the election rules force all candidates to register well in advance). We know that some set V of voters will certainly vote (for example, because they have already voted and this information is public[2]). The set of voters who have not decided to vote yet is W. From some source (e.g., from prior experience) we have a probability distribution P on the number of voters from W that will participate in the election.

[1] Note that while full-knowledge assumption regarding voters' preferences might seem very unrealistic, it is standard within computational social choice literature, and in our case can often be justified (e.g., election polls can provide a good approximation of such knowledge).

[2] Naturally, in typical political elections such information would not be public and we would have to rely on polls. On the other hand, in multiagent systems there can be cases where votes are public.

ticipate in the election (we assume that each equal-sized subset from W is equally likely to joint the election; we have no prior knowledge as to which eligible voters are more likely to vote). That is, for each i, $0 \leq i \leq |W|$, by $P(i)$ we denote the probability that exactly i voters from W join the election (we assume that $P(i)$ is easily computable). By $Q(i)$ we denote the probability that a certain designated candidate p wins provided that exactly i, randomly chosen, voters from W participate in the election. The probability that p wins is given by:

$$P(0)Q(0) + P(1)Q(1) + \cdots P(|W|)Q(|W|).$$

We use this formula to compute the probability of each candidate's victory, which gives some idea as to who is the likely winner of the election.

To compute $Q(i)$ we have to compute for how many sets W of size exactly i candidate p wins and divide it by $\binom{|W|}{i}$. However, this is essentially a variant of control by adding voters, where instead of asking if it is possible to ensure p's victory by adding at most i voters, we ask how many ways are there to add i voters so that p is a winner. Thus, our winner prediction problem reduces to a counting variant of an election control problem. Analogous reasoning can be given for the case of deleting voters and adding/deleting candidates. Thus, formally our winner prediction problems reduce to counting variants of election control problems.

Computational study of election control was initiated by Bartholdi, Tovey, and Trick (1992) and was continued by Hemaspaandra, Hemaspaandra, and Rothe (2007), Meir et al. (2008), Erdélyi, Nowak, and Rothe (2009), Faliszewski, Hemaspaandra, and Hemaspaandra (2011b), and others (see the survey of Faliszewski, Hemaspaandra and Hemaspaandra (2010)). However, to the best of our knowledge, this is the first paper to study counting variants of election control.

Preliminaries

Elections and Voting Systems. An *election* E is a pair (C, V) such that C is a finite set of candidates and V is a finite collection of voters. We typically use m to denote the number of candidates and n to denote the number of voters. Each voter has a preference order in which he or she ranks candidates, from the most desirable one to the most despised one. For example, if $C = \{a, b, c\}$ and a voter likes b most and a least, then this voter would have preference order $b > c > a$. (However, under approval voting, instead of ranking the candidates each voter simply indicates which candidates he or she approves of.) We sometimes use the following notation. Let A be some subset of the candidate set. Putting A in a preference order means listing members of A in lexicographic order and putting $\overleftarrow{A}$ in a preference order means listing members of A in the reverse of the lexicographic order. For example, if $C = \{a, b, c, d\}$ then $a > C - \{a, b\} > d$ means $a > c > d > b$ and $a > \overleftarrow{C - \{a, b\}} > d$ means $a > d > c > b$.

A *voting system* is a rule which specifies how election winners are determined. We allow an election to have more than one winner, or even to not have winners at all. In practice, tie-breaking rules are used, but here we disregard this issue by simply using the unique winner model (see Definition 4). However, we point the reader to (Obraztsova, Elkind, and Hazon 2011; Obraztsova and Elkind 2011) for a discussion regarding the influence of tie-breaking for the case of election manipulation problems (see (Faliszewski, Hemaspaandra, and Hemaspaandra 2010; Faliszewski and Procaccia 2010) for overviews of the manipulation problem).

Let $E = (C, V)$ be an election. For each candidate $c \in C$, we define c's k-Approval score, $score_E^k(c)$, to be the number of voters in V that rank c among the top k candidates; the candidates with highest scores win. Plurality rule is 1-Approval (and we write $score_E^p(c)$ to denote the Plurality score of candidate c). Under Approval (without the qualifying "k-"), the score of a candidate $c \in C$, $score_E^a(c)$, is the number of voters that approve of c. Again, the candidates with highest scores win. Under Condorcet's rule, a candidate $c \in C$ is a winner if and only if for each $c' \in C - \{c\}$, more than half of the voters prefer c to c' (i.e., more than half of the voters have preference orders where $c > c'$). There can be at most one winner under Condorcet's rule and he or she is called the *Condorcet winner*. We write $N_E(c, c')$ to denote the number of voters in V that prefer c to c'; c is a Condorcet winner exactly if $N_E(c, c') > N_E(c', c)$ for each $c' \in C - \{c\}$. Under Maximin, a candidate c's score, $score_E^m(c)$, is defined to be $\min_{d \in C - \{c\}} N_E(c, d)$. The candidates with highest Maximin score are Maximin winners.

Notation for Graphs. We assume familiarity with basic concepts of graph theory. Given an undirected graph G, by $V(G)$ we mean its set of vertices and by $E(G)$ we mean its set of edges.[3] Whenever we discuss a bipartite graph G, we assume that $V(G)$ is partitioned into two subsets, X and Y. We write $X(G)$ to denote X, and $Y(G)$ to denote Y.

Computational Complexity. We assume that the reader is familiar with standard notions of complexity theory. Let us, however, briefly review notions regarding the complexity theory of counting problems. Let A be some computational problem where for each instance I we ask if there exists some mathematical object satisfying a given condition. In the counting variant of A, denoted $\#A$, we ask how many such mathematical objects exist. For example, consider the following definition.

Definition 1. *An instance of X3C is a pair $(B, \mathcal{S})$, where $B = \{b_1, \ldots, b_{3k}\}$ and $\mathcal{S} = \{S_1, \ldots, S_n\}$ is a family of 3-element subsets of B. In X3C we ask if it is possible to find exactly k sets in $\mathcal{S}$ whose union is exactly B. In #X3C we ask how many k-element subsets of $\mathcal{S}$ have B as their union.*

The class of counting variants of NP-problems is called #P and the class of functions computable in polynomial time is called FP. To reduce counting problems to each other, we will use one of the following reducibility notions.

Definition 2. *Let $\#A$ and $\#B$ be two counting problems.*

1. *We say that $\#A$ Turing reduces to $\#B$ if there exists an algorithm that solves $\#A$ in polynomial time given oracle access to $\#B$.*

[3] We use this slightly nonstandard notation is to be able to also use symbols V and E to denote voter collections and elections.

2. *We say that $\#A$ metric reduces to $\#B$ if there exist two polynomial-time computable functions, f and g, such that for each instance I of $\#A$ it holds that (1) $f(I)$ is an instance of $\#B$, and (2) if $\#B(f(I))$ is the number of solutions for $f(I)$, then $g(I, \#B(f(I)))$ outputs the number of solutions for I.*
3. *We say that $\#A$ parsimoniously reduces to $\#B$ if $\#A$ metric reduces to $\#B$ via functions f and g such that for each instance I and each integer k, $g(I, k) = k$.*

For a given reducibility notion R, we say that a problem is #P-R-complete if it belongs to #P and every #P-problem R reduces to it. For example, #X3C is #P-parsimonious-complete (Hunt et al. 1998). Throughout this paper we will write #P-complete to mean #P-parsimonious-complete. Turing reductions were used, e.g., by Valiant (1979) to show #P-hardness of computing a permanent of a 0/1 matrix. As a result, he also showed #P-Turing-hardness of the following problem (but see also (Zankó 1991)).

Definition 3. *In #PerfectMatching we are given a bipartite graph $G = (G(X), G(Y), G(E))$ with $\|G(X)\| = \|G(Y)\|$ and we ask how many perfect matchings does G have.*

Metric reductions were introduced by Krentel (1988), and parsimonious reductions were defined by Simon (1975).

Counting Variants of Control Problems

Let us now formally define counting variants of election control problems. We are interested in control by adding candidates (AC), control by deleting candidates (DC), control by adding voters (AV), and control by deleting voters (DV). For each of these problems, we consider its constructive variant (CC) and its destructive variant (DC).

Definition 4. *Let R be a voting system. In each of the counting variants of constructive control problems we are given a candidate set C, a voter collection V, a nonnegative integer k, and a designated candidate $p \in C$. In constructive control by adding voters we are additionally given a collection W of unregistered voters, and in constructive control by adding candidates we are additionally given a set A of unregistered candidates. In these problems we ask for the following quantities:*

1. *In control by adding voters (R-#CCAV), we ask how many sets W', $W' \subseteq W$, are there such that p is the unique winner of R-election $(C, V \cup W')$, where $|W'| \leq k$.*
2. *In control by deleting voters (R-#CCDV), we ask how many sets V', $V' \subseteq V$ are there such that p is the unique winner of R-election $(C, V - V')$, where $|V'| \leq k$.*
3. *In control by adding candidates (R-#CCAC), we ask how many sets A', $A' \subseteq A$, are there such that p is the unique winner of R-election $(C \cup A', V)$, where $|A'| \leq k$.*
4. *In control by deleting candidates (R-#CCDC), we ask how many sets C', $C' \subseteq C$, are there such that p is the unique winner of R-election $(C - C', V)$, where $|C'| \leq k$ and $p \notin C'$.*

Destructive variants are defined identically, except that we ask for the number of settings where the designated candidate is not the unique winner.

The above problems are interesting in their own right and because they model winner-prediction scenarios.

Results

We now present our complexity results regarding counting variants of election control problems. Not surprisingly, for each NP-complete control problem that we have considered, its counting variant turned out to be complete for #P (for some reducibility type). However, interestingly, we have also found examples of election systems and control types where the decision variant is easy, but the counting variant is hard. This happens, e.g., for k-Approval, $k \geq 2$, and destructive voter control, and for Maximin and control by deleting candidates. These results are quite unexpected (especially for the case of Maximin).

We omit many of our proofs due to space restrictions (further, some of the omitted hardness results, but not all, are easy adaptations of the decision-variant NP-hardness proofs already given in the literature). We obtain all of our destructive-case results via the following observation.

Theorem 5. *Let R be a voting system, let $\#\mathcal{C}$ be one of R-#CCAC, R-#CCDC, R-#CCAV, R-#CCDV, and let $\#\mathcal{D}$ be the destructive variant of $\#\mathcal{C}$. Then, $\#\mathcal{C}$ metric reduces to $\#\mathcal{D}$.*

Proof. We give a metric reduction from $\#\mathcal{C}$ to $\#\mathcal{D}$. Let I be an instance of $\#\mathcal{C}$, where the goal is to make some candidate p the unique winner. We define $f(I)$ to be an instance of $\#\mathcal{D}$ that is identical to I, except that the goal is to ensure that p is not the unique winner. Let s_I be the number of all possible solutions for I[4] (and, naturally, also the number of all possible solutions for $f(I)$). It is easy to see that s_I is polynomial-time computable and that the number of solutions for I is exactly $s_I - \#\mathcal{D}(f(I))$. Thus, we define $g(I, \#\mathcal{D}(f(I))) = s_I - \#\mathcal{D}(f(I))$. We see that the reduction is polynomial-time and correct. □

In the following we present our results specific to Plurality, k-Approval, Approval, Condorcet, and Maximin.

Plurality Voting. Under plurality voting, counting variants of both control by adding voters and control by deleting voters are in FP. In both cases our algorithms are based on dynamic programming.

Theorem 6. *Plurality-#CCAV, Plurality-#DCAV, Plurality-#CCDV, and Plurality-#DCDV are in* FP.

Proof. Due to space restrictions we will only give proofs for the adding-voters cases. The proofs for the deleting-voters cases are similar in spirit.

Let $I = (C, V, W, p, k)$ be an input instance of Plurality-#CCAV, where $C = \{p, c_1, \ldots, c_{m-1}\}$ is the candidate set, V is the set of registered voters, W is the set of unregistered voters, p is the designated candidate, and k is the upper bound on the number of voters that can be added. We now describe a polynomial-time algorithm that computes the number of solutions for I.

Let A_p be the set of voters from W that rank p first. Similarly, for each $c_i \in C$, let A_{c_i} be the set of voters from W that rank c_i first. We also define $count(C, V, W, p, k, j)$ to be

[4] For example, if $\#\mathcal{C}$ was $\#CCDV$ and $I = (C, V, p, k)$, then s_I would be the number of up-to-size-k subsets of V.

the number of sets $W' \subseteq W - A_p$ such that (1) $|W'| \leq k-j$, and (2) in election $(C, V \cup W')$ each candidate $c_i \in C$, $1 \leq i \leq m-1$, has score at most $score^p_{(C,V)}(p) + j - 1$.

Our algorithm works as follows. First, we compute k_0, the minimum number of voters from A_p that need to be added to V to ensure that p has plurality score higher than any other candidate (provided no other voters are added). Clearly, if p already is the unique winner of (C, V) then k_0 is 0, and otherwise k_0 is $\max_{c_i \in C}(score^p_{(C,V)}(c_i) - score^p_{(C,V)}(p) + 1)$. Then, for each j, $k_0 \leq j \leq \min(k, |A_p|)$, we compute the number of sets W', $W' \subseteq W$, such that W' contains exactly j voters from A_p, at most $k - j$ voters from $W - A_p$, and p is the unique winner of $(C, V \cup W')$. It is easy to verify that for a given j, there is exactly $h(j) = \binom{|A_p|}{j} \cdot count(C, V, W, p, k, j)$ such sets. Our algorithm returns $\sum_{j=k_0}^{\min(k,|A_p|)} h(j)$. The reader can easily verify that this indeed is the correct answer. To complete the proof it suffices to show a polynomial-time algorithm for computing $count(C, V, W, p, k, j)$.

Let us fix j, $k_0 \leq j \leq \min(k, |A_p|)$. We now show how to compute $count(C, V, W, p, k, j)$. Our goal is to count the number of ways in which we can add at most $k - j$ voters from $W - A_p$ so that no candidate $c_i \in C$ has score higher than $score^p_{(C,V)}(p) + j - 1$. For each candidate $c_i \in C$, we can add at most $l_i = \min\big(|A_{c_i}|, j + score^p_{(C,V)}(p) - score^p_{(C,V)}(c_i) - 1\big)$ voters from A_{c_i}; otherwise c_i's score would exceed $score^p_{(C,V)}(p) + j - 1$.

For each i, $1 \leq i \leq m-1$, and each t, $0 \leq t \leq k - j$, let $a_{t,i}$ be the number of sets $W' \subseteq A_{c_1} \cup A_{c_2} \cup \cdots \cup A_{c_i}$ that contain exactly t voters and such that each candidate $c_1, c_2, \ldots, c_i$ has score at most $score^p_{(C,V)}(p) + j - 1$ in the election $(C, V \cup W')$. Naturally, $count(C, V, W, p, k, j) = \sum_{t=0}^{k-j} a_{t,m-1}$. It is easy to check that $a_{t,i}$ satisfies the following recursion:

$$a_{t,i} = \begin{cases} \sum_{s=0}^{\min(l_i,t)} \binom{|A_{c_i}|}{s} a_{t-s,i-1}, & \text{if } t > 0, i > 1, \\ 1, & \text{if } t = 0, i > 1, \\ \binom{|A_1|}{t}, & \text{if } t \leq |A_{c_1}|, i = 1, \\ 0, & \text{if } t > |A_{c_1}|, i = 1. \end{cases}$$

Thus, for each t, i we can compute $a_{t,i}$ using standard dynamic programming techniques in polynomial time. Thus, $count(C, V, W, p, k, j)$ also is polynomial-time computable. This completes the proof that Plurality-#CCAV is in FP.

For the case of Plurality-#DCAV, it suffices to invoke Theorem 5 and the just-proved result for #CCAV. □

On the other hand, for Plurality voting #CCAC and #CCDC are #P-complete and this follows from proofs already given in the literature (Faliszewski, Hemaspaandra, and Hemaspaandra 2011a).

Theorem 7. *Plurality-#CCAC and Plurality-#CCDC are* #P*-complete.*

Now, Corollary 8 follows by combining Theorems 7 and 5.

Corollary 8. *Plurality-#DCAC and Plurality-#DCDC are* #P*-metric-complete.*

***k*-Approval Voting.** While k-Approval is in many respects a simple generalization of the Plurality rule, it turns out that for $k \geq 2$ all counting variants of control problems are intractable for k-approval. This is quite expected for candidate control as decision variants of these problems are NP-complete (see (Lin 2011; Elkind, Faliszewski, and Slinko 2010)), but is more intriguing for voter control (as shown by Lin, for 2-Approval all voter control decision problems are in P, and, as one can verify, for k-Approval all destructive voter control decision problems are in P).

Theorem 9. *For each k, $k \geq 2$, k-Approval-#CCAC and k-Approval-#CCDC are* #P*-complete, and k-Approval-#DCAC and k-Approval-#DCDC are* #P*-metric-complete.*

The proof of this theorem follows by padding the construction used in the proof of Theorem 7 and by applying Theorem 5 (for the destructive variants). However, to show hardness of voter control we need some new ideas.

Theorem 10. *For each k, $k \geq 2$, k-Approval-#CCAV is* #P*-Turing-complete and k-Approval-#CCDV is* #P*-metric-complete.*

Proof. Due to space restriction, We will give a proof for $k = 2$ and for #CCAV only. The case of #CCDV follows by a similar proof (though with some extra twists) and the cases for $k > 2$ follow by padding.

We give a Turing reduction from #PerfectMatching to 2-Approval-#CCAV. Let $G = (G(X), G(Y), G(E))$ be our input bipartite graph, where $G(X) = \{x_1, \ldots, x_n\}$ and $G(Y) = \{y_1, \ldots, y_n\}$ are sets of vertices, and $G(E) = \{e_1, \ldots, e_m\}$ is the set of edges. We form an election $E = (C, V)$ and a collection W of unregistered voters as follows. We set $C = \{p, b_1, b_2\} \cup G(X) \cup G(Y)$ and we let $V = (v_1, v_2)$, where v_1 has preference order $p > b_1 > C - \{p, b_1\}$ and v_2 has preference order $p > b_2 > C - \{p, b_2\}$. We let $W = (w_1, \ldots, w_m)$, where for each ℓ, $1 \leq \ell \leq m$, if $e_\ell = \{x_i, y_j\}$ then w_ℓ has preference order $x_i > y_j > C - \{x_i, y_j\}$.

Thus, in election E candidate p has score 2, candidates b_1 and b_2 have score 1, and candidates in $G(X) \cup G(Y)$ have score 0. We form an instance I of 2-Approval-#CCAV with election $E = (C, V)$, collection W of unregistered voters, designated candidate p, and the number of voters that can be added set to n. We form instance I' to be identical, except we allow to add at most $n - 1$ voters.

It is easy to verify that the number of 2-Approval-#CCAV solutions for I (for I') is the number of matchings in G of cardinality at most n (the number of matchings in G of cardinality at most $n-1$). (Each unregistered voter corresponds to an edge in G and one cannot add two edges that share a vertex as then p would no longer be the unique winner.) The number of perfect matchings in G is exactly the number of solutions for I minus the number of solutions for I'. □

By applying a technical trick on top of the proof of the above theorem, we get Corollary 11 below, and by Theorem 5 we obtain Corollary 12.

Corollary 11. *For each k, $k \geq 2$, $(k+1)$-Approval-#CCAV and k-Approval-#CCDV are* #P*-metric-complete.*

Corollary 12. *2-Approval-#DCAV is* #P*-Turing-complete and for each k, $k \geq 2$, $(k+1)$-Approval-#DCAV and k-Approval-#DCDV are* #P*-metric-complete.*

Approval Voting and Condorcet Voting. Let us now consider Approval voting and Condorcet voting. While these two systems may seem very different in spirit, their behavior with respect to election control is similar. Specifically, for both systems #CCAV and #CCDV are #P-complete, for both systems it is impossible to make some candidate a winner by adding candidates, and for both systems it is impossible to prevent someone from winning by deleting candidates. Yet, for both systems #DCAC and #CCDC are in FP via almost identical algorithms.

Theorem 13. *Approval-#CCAV, Approval-#CCDV, Condorcet-#CCAV, and Condorcet-#CCDV are* #P*-complete. Their destructive variants are* #P*-metric-complete.*

In Theorem 13, the results for Approval follow from the work of Hemaspaandra, Hemaspaandra, and Rothe (2007), the result for Condorcet-#CCDV follows from the proofs of Theorems 5.1 and 4.19 of Faliszewski et al. (2009), and for Condorcet-#CCAV we have found a new reduction.

Theorem 14. *Approval-#DCAC, Condorcet-#DCAC, Approval-#CCDC, and Condorcet-#CCDC are in* FP.

Proof. We will give the proof for the case of Approval-#CCDC. We omit the other cases due to space restrictions.

Let $I = (C, V, p, k)$ be an instance of approval-#CCDC. The only way to ensure that $p \in C$ is the unique winner is to remove all candidates $c \in C - \{p\}$ such that $score^a_{(C,V)}(c) \geq score^a_{(C,V)}(p)$. Such candidates can be found immediately. Let's assume that there are k_0 such candidates. After removing all of them, we can also remove $k - k_0$ or less of any remaining candidates other than p. Thus, the result is $\sum_{i=0}^{k-k_0} \binom{|C|-k_0-1}{i}$. □

Maximin Voting. The complexity of decision variants of control for Maximin was studied by Faliszewski, Hemaspaandra, and Hemaspaandra (2011b). In particular, they showed that under Maximin all voter control problems are NP-complete and an easy adaptation of their proofs gives the following theorem.

Theorem 15. *Maximin-#CCAV and Maximin-#CCDV are* #P*-complete, and Maximin-#DCAV and Maximin-#DCDV are* #P*-metric-complete.*

On the other hand, among the candidate control problems for Maximin, only Maximin-CCAC is NP-complete (DCAC, CCDC, and DCDC are in P). Still, this hardness of control by adding candidates translates into the hardness of all the counting variants of candidate control.

Theorem 16. *Maximin-#CCAC is* #P*-complete and Maximin-#DCAC is* #P*-metric-complete.*

(We omit the proof—an adaptation of Maximin-CCAC NP-completeness proof of Faliszewski at al.—due to space restriction.) The cases of Maximin-#CCDC and Maximin-#DCDC are more complicated and require new ideas because decision variants of these problems are in P.

Theorem 17. *Both Maximin-#CCDC and Maximin-#DCDC are* #P*-Turing-complete.*

Proof. We consider the #CCDC case first. Clearly, the problem belongs to #P and it remains to show hardness. We will do so by giving a Turing reduction from #PerfectMatching.

Let $G = (G(X), G(Y), G(E))$ be our input graph, where $G(X) = \{x_1, \ldots, x_n\}$ and $G(Y) = \{y_1, \ldots, y_n\}$ are sets of vertices, and $E = \{e_1, \ldots, e_m\}$ is the set of edges. For each nonnegative integer k, define $g(k)$ to be the number of matchings in G that contain exactly k edges (e.g., $g(n)$ is the number of perfect matchings in G).

We define the following election $E = (C, V)$. We set $C = G(E) \cup S \cup B \cup \{p\}$, where $S = \{s_0, \ldots, s_n\}$ and $B = \{b^\ell_{i,j} \mid 0 \leq \ell \leq n, i < j$, and e_i and e_j share a vertex$\}$. To build voter collection V, for each two candidates $a, b \in C$, we define $v(a, b)$ to be a pair of voters with preference orders $a > b > C - \{a, b\}$ and $\overleftarrow{C - \{a, b\}} > a > b$. We construct V as follows:

1. For each $s_i \in S$, we add pair $v(s_i, p)$.
2. For each $s_i \in S$, we add two pairs $v(s_i, s_{i+1)})$, where $i + 1$ is taken modulo $n + 1$.
3. For each $s_i \in S$ and each $e_t \in E$, we add two pairs $v(s_i, e_t)$.
4. For each $e_i, e_j \in E, i < j$, where e_i and e_j share a vertex, and for each ℓ, $0 \leq \ell \leq n$, we add two pairs $v(e_i, b^\ell_{i,j})$ and two pairs $v(e_j, b^\ell_{i,j})$.

Let T be the total number of pairs $v(a, b)$, $a, b \in C$, included in V. By our construction, the following properties hold:

1. $score^m_E(p) = T - 1$ and it is impossible to change the score of p be deleting n candidates or fewer (this is because there are $n + 1$ candidates $s_i \in S$ such that $N_E(p, s_i) = T - 1$.
2. For each $s_i \in S$, $score^m_E(s_i) = T - 2$, but deleting s_{i-1} (where we take $i - 1$ modulo $n + 1$) increases the score of s_i to $T + 1$.
3. For each $e_t \in E$, $score^m_E(e_t)$ is $T - 2$.
4. For each $b^\ell_{i,j} \in B$, $score^m_E(b^\ell_{i,j}) = T - 2$ and it remains $T - 2$ if we delete either e_i or e_j, but it becomes T if we delete both e_i and e_j.

Note that p is the unique winner of E. For each k, $0 \leq k \leq n$, we form instance $I(k) = (C, V, p, k)$ of Maximin-#CCDC. We define $f(k) = \#I(k) - \#I(k-1)$. That is, $f(k)$ is the number of solutions for $I(k)$ where we delete exactly k candidates. We claim that for each k, $1 \leq k \leq n$, it holds that $f(k) = \sum_{j=0}^{k} \binom{\|B\|}{j} g(k-j)$. Why is this so? First, note that by the listed-above properties of E, deleting any subset C' of candidates from C that contains some member of S prevents p from being a winner. Thus, we can only delete subsets C' of C that contains candidates in $G(E) \cup B$. Let us fix a nonnegative integer r, $0 \leq r \leq n$. Let $C' \subseteq G(E) \cup B$ be such that p is the unique Maximin winner of $E' = (C - C', V)$ and $\|C'\| = r$. Let $r_B = \|C' \cap B\|$

Problem	Plurality	k-Approval $k \geq 2$	Approval	Condorcet	Maximin
#CCAC	#P-complete	#P-complete	–	–	#P-complete
#DCAC	#P-metric-complete	#P-metric-complete	FP	FP	#P-metric-complete
#CCDC	#P-complete	#P-complete	FP	FP	#P-Turing-complete
#DCDC	#P-metric-complete	#P-metric-complete	–	–	#P-Turing-complete
#CCAV	FP	#P-Turing-complete	#P-complete	#P-complete	#P-complete
#DCAV	FP	#P-Turing-complete	#P-metric-complete	#P-metric-complete	#P-metric-complete
#CCDV	FP	#P-metric-complete	#P-complete	#P-complete	#P-complete
#DCDV	FP	#P-metric-complete	#P-metric-complete	#P-metric-complete	#P-metric-complete

Table 1: The complexity of counting variants of control problems. A dash in an entry means that the given system is *immune* to the type of control in question (i.e., it is impossible to achieve the desired effect by the action this control problem allows; technically this means the answer to the counting question is always 0). Immunity results were established by Bartholdi, Tovey, and Trick (1989) for the constructive cases, and by Hemaspaandra, Hemaspaandra, and Rothe (2007) for the destructive cases.

and $r_{G(E)} = \|C' \cap G(E)\|$. It must be the case that for each $e_i, e_j \in G(E)$, $i < j$, where e_i and e_j share a vertex, C' contains at most one of them. Otherwise, E' would contain at least one of the candidates $b^{\ell}_{i,j}$, $0 \leq \ell \leq n$, and this candidate would have score higher than p. Thus, the candidates in $C' \cap G(E)$ correspond to a matching in G of cardinality $r_{G(E)}$. On the other hand, since $r_B \leq n$, $C \cap B$ contains an arbitrary subset of B. Thus, there are exactly $\binom{\|B\|}{r_B} g(r_{G(E)})$ such sets C'. Our formula for $f(k)$ is correct.

Now, using standard algebra (a process similar to Gauss elimination), it is easy to verify that given values $f(1), f(2), \ldots, f(n)$, it is possible to compute (in this order) $g(0), g(1), \ldots, g(n)$. Together with the fact that constructing each $I(k)$, $0 \leq k \leq n$, requires polynomial time with respect to the size of G, this proves that given oracle access to Maximin-#CCDC, we can solve #PerfectMatching. Thus, Maximin-#CCDC is #P-Turing-complete and, by Theorem 5, so is Maximin-#DCDC. □

Related Work

From the high-level perspective, the focus of this paper is on the complexity predicting election winners. However, our model is just one of many approaches to this problem, which in various forms and shapes has been studied in the literature for some years already. For example, to model imperfect knowledge regarding voters' preferences, Konczak and Lang (2005) introduced the possible winner problem, further studied by many other researchers (see, e.g., (Xia and Conitzer 2011; Betzler and Dorn 2010; Bachrach, Betzler, and Faliszewski 2010; Chevaleyre et al. 2010; Xia, Lang, and Monnot 2011)). In the possible winner problem, each voter is represented via a partial preference order and we ask if there is an extension of these partial orders to total orders that ensures a given candidate's victory. Bachrach, Betzler, and Faliszewski (2010) extended the model by considering counting variants of possible winner problems. Namely, they asked for how many extensions of the votes a given candidate wins, in effect obtaining the probability of the candidate's victory. This is very similar to our approach, but there are also important differences. In the work of Bachrach et al., we have full knowledge regarding the identities of candidates and voters participating in the election, but we are uncertain about voters' preference orders. In our setting, we have full knowledge about voters' preference orders, but we are uncertain about the identities of candidates/voters participating in the election.

Hazon et al. (2008) considered a setting where each voter has a probability distribution among several possible votes and where the task is to compute the probability that a given candidate wins.

Going in a different direction, our work is related to the paper of Walsh and Xia (2011) on lot-based elections, where winner determination reduces to counting variants of control by adding/deleting voters.

Conclusions and Future Work

We have considered a model of predicting election winners in settings where there is uncertainty regarding the structure of the election (that is, regarding the exact set of candidates and the exact collection of voters participating in the election). We have shown that our model corresponds to the counting variants of election control problems (specifically, we have focused on election control by adding/deleting candidates and voters). We have considered Plurality, Approval, Condorcet, k-Approval, and Maximin (see Table 1 for our results). For the former three, the complexity of counting variants of control is analogous to the complexity of decision variants of respective problems, but for the latter two, some of the counting control problems are more computationally demanding than their decision counterparts.

Many of our results indicate computational hardness of winner prediction problems. Thus, in practice one might have to seek heuristic algorithms or approximate solutions (e.g., sampling-based algorithms similar to the one of Bachrach, Betzler, and Faliszewski (2010, Theorem 6)).

There are many ways to extend our work. For example, it would be natural to consider settings where for each voter v_i (for each candidate c_i) we have a probability p_i that this voter casts a vote (that this candidate participates in the election), and we would be to compute the probability of a given candidate winning. (For the case of all the probabilities being equal, our model already captures this setting.)

Acknowledgements. An early version of this paper appeared in IJCAI Workshop on Social Choice and Artificial

Intelligence (2011). We thank both the AAAI and WSCAI reviewers for very helpful feedback. Piotr Faliszewski was in part supported by AGH University of Technology Grant no. 11.11.120.865 and by Foundation for Polish Science's program Homing/Powroty.

References

Bachrach, Y.; Betzler, N.; and Faliszewski, P. 2010. Probabilistic possible winner determination. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 697–702. AAAI Press.

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

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

Chevaleyre, Y.; Lang, J.; Maudet, N.; and Monnot, J. 2010. Possible winners when new candidates are added: The case of scoring rules. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 762–767. AAAI Press.

Dwork, C.; Kumar, R.; Naor, M.; and Sivakumar, D. 2001. Rank aggregation methods for the web. In *Proceedings of the 10th International World Wide Web Conference*, 613–622. ACM Press.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2010. Cloning in elections. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 768–773. AAAI Press.

Ephrati, E., and Rosenschein, J. 1997. A heuristic technique for multi-agent planning. *Annals of Mathematics and Artificial Intelligence* 20(1–4):13–67.

Erdélyi, G.; Nowak, M.; and Rothe, J. 2009. Sincere-strategy preference-based approval voting fully resists constructive control and broadly resists destructive control. *Mathematical Logic Quarterly* 55(4):425–443.

Faliszewski, P., and Procaccia, A. 2010. AI's war on manipulation: Are we winning? *AI Magazine* 31(4):52–64.

Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2009. Llull and Copeland voting computationally resist bribery and constructive control. *Journal of Artificial Intelligence Research* 35:275–341.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2010. Using complexity to protect elections. *Communications of the ACM* 53(11):74–82.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2011a. The complexity of manipulative attacks in nearly single-peaked electorates. Technical Report arXiv:1105.5032v1, arXiv.org.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. 2011b. Multimode control attacks on elections. *Journal of Artificial Intelligence Research* 40:305–351.

Ghosh, S.; Mundhe, M.; Hernandez, K.; and Sen, S. 1999. Voting for movies: The anatomy of recommender systems. In *Proceedings of the 3rd Annual Conference on Autonomous Agents*, 434–435. ACM Press.

Hazon, N.; Aumann, Y.; Kraus, S.; and Wooldridge, M. 2008. Evaluation of election outcomes under uncertainty. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 959–966.

Hemaspaandra, E.; Hemaspaandra, L.; and Rothe, J. 2007. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence* 171(5–6):255–285.

Hunt, H.; Marathe, M.; Radhakrishnan, V.; and Stearns, R. 1998. The complexity of planar counting problems. *SIAM Journal on Computing* 27(4):1142–1167.

Konczak, K., and Lang, J. 2005. Voting procedures with incomplete preferences. In *Proceedins of the Multidisciplinary IJCAI-05 Worshop on Advances in Preference Handling*, 124–129.

Krentel, M. 1988. The complexity of optimization problems. *Journal of Computer and System Sciences* 36(3):490–509.

Lin, A. 2011. The complexity of manipulating k-approval elections. In *Proceedings of the Third International Conference on Agents and Artificial Intelligence*, 212–218.

Meir, R.; Procaccia, A.; Rosenschein, J.; and Zohar, A. 2008. The complexity of strategic behavior in multi-winner elections. *Journal of Artificial Intelligence Research* 33:149–178.

Obraztsova, S., and Elkind, E. 2011. On the complexity of voting manipulation under randomized tie-breaking. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 319–324.

Obraztsova, S.; Elkind, E.; and Hazon, N. 2011. Ties matter: Complexity of voting manipulation revisited. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, 71–78.

Simon, J. 1975. *On Some Central Problems in Computational Complexity*. Ph.D. Dissertation, Cornell University, Ithaca, N.Y. Available as Cornell Department of Computer Science Technical Report TR75-224.

Valiant, L. 1979. The complexity of computing the permanent. *Theoretical Computer Science* 8(2):189–201.

Walsh, T., and Xia, L. 2011. Venetian elections and lot-based voting rules. In *Proceedings of IJCAI Workshop on Social Chocie and Aritficial Intelligence*, 93–98.

Xia, L., and Conitzer, V. 2011. Determining possible and necessary winners given partial orders. *Journal of Artificial Intelligence Research* 41:25–67.

Xia, L.; Lang, J.; and Monnot, J. 2011. Possible winners when new alternatives join: New results coming up! In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, 829–836.

Zankó, V. 1991. #P-completeness via many-one reductions. *International Journal of Foundations of Computer Science* 2(1):76–82.