Possible Winners in Noisy Elections

Krzysztof Wojtas
AGH University of Science and Technology, Kraków, Poland

Piotr Faliszewski
AGH University of Science and Technology, Kraków, Poland

Abstract

Predicting election winners (or, election possible winners) is an important topic in computational social choice. Very generally put, we consider the following setting: There is some set of candidates C and some set of voters V (with preferences over C). We either do not know which candidates will take part in the election or we do not know which voters will cast their votes. However, for each set $C' \subseteq C$ (each set $V' \subseteq V$) we know probability $P_C(C')$ that exactly candidates in C' participate in the election (probability $P_V(V')$ that exactly voters in V' cast their votes). Our goal is to compute the probability that a given candidate $c \in C$ wins the election. In its full generality—with unrestricted probability distributions P_C and P_V—these problems can very easily become computationally hard. We provide natural restrictions on P_C and P_V that allow us to obtain positive results for several election systems, including plurality, approval, and Condorcet's rule. On the technical side, our problems reduce to counting solutions to the problems of election control.

1 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? Naturally, predicting winners is a hard task, full of uncertainties. For example, we typically are not sure which voters will eventually cast their votes and, sometimes, even the set of available candidates may be uncertain (consider, e.g., a candidate withdrawing due to personal reasons). Further, typically we do not have complete knowledge regarding each voters' preference order.

Nonetheless, to optimize their behavior, agents involved in an election try to somehow tackle the winner prediction problem. 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, 2008; Betzler and Dorn, 2009; Bachrach *et al.*, 2010]. In this paper we focus on a different type of uncertainty: We consider settings where the set of participating candidates and the set of voters are uncertain. (However, we do assume perfect knowledge regaring voters' preferences.)

Specifically, we study the following setting. We are given a voting rule, a set C of m candidates, and a set V of n voters (for each voter we have perfect knowledge as to how she would vote). We consider two possible scenarios:

1. The set of candidates is fixed, but for each set of voters V', $V' \subseteq V$, we have probability $P_V(V')$ that exactly the voters from V' show up for the vote.
2. The set of voters is fixed, but for each set of candidates C', $C' \subseteq C$, we have probability $P_C(C')$ that exactly the candidates from C' participate in the election.

Our goal is to compute, for each candidate $c \in C$, the probability that c is a winner.

Naturally, our task would very quickly become computationally prohibitive (or, difficult to represent on a computer) if we did not assume anything about P_C and P_V. We use the following restrictions: First, we assume that both P_C and P_V are polynomial-time computable. Second, we would like to assume that for each subset V' of voters (each subset C' of candidates) the value $P_V(V')$ (the value $P_C(C')$) depends only on the cardinality of V' (only on the cardinality of C'). In other words, we have a probability distribution regarding the number of voters (the number of candidates) participating in the election, but each same-cardinality subset is equally likely.

However, this second assumption is slightly too strong. Often, we may have additional knowledge regarding the nature of possible changes in the candidate/voter set. For example, the rules may be such that after a given point of time candidates can withdraw from the election but no new candidates can register. Similarly, we may know that some votes have already been cast and cannot be withdrawn. Thus, we refine our model to be the following: We start with some candidates and voters already in the election and we ask for the probability that a given candidate wins assuming that some random number of voters/candidates is added/deleted.

Formally, it turns out that our winner prediction setting reduces to the counting variants of election control problems; computational study of election control problems 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 [2011], and others (see the survey of Faliszewski, Hemaspaadra, and Hemaspaandra, al. [2010]). However, to the best of our knowledge, this is the first paper to study counting variants of election control. (However, we should mention that Bachrach et al. [2010] consider counting variants of possible-winner problems. Nonetheless, their model and motivation are different from ours; they assume the set of voters is fixed, but the voters are unsure as to how to vote. We assume the voters are certain about their votes, but unsure about participation in the election. The resulting technical problem is very different. Somewhere in the middle between these two approaches is the model of [Hazon *et al.*, 2008], where each voter has a probability distribution among several possible votes.)

Our results are very preliminary. Following Hemaspaandra, Hemaspaandra, and Rothe [2007], we focus on three, quite different in spirit, voting rules: plurality, Condorcet's rule, and approval voting. Our results show that counting variants of constructive control by adding/deleting candidates/voters for these voting rules are polynomial-time solvable whenever the decision variants are. This means that for the respective cases our winner prediction problems are polynomial-time solvable.

The paper is organized as follows. In Section 2 we formally define elections, the voting rules that we study, and provide brief background on complexity theory (focusing on counting problems). In Section 3 we formally define counting variants of election control problems and link them to the winner prediction scenarios that motivate our work. Section 4 contains our technical results. We conclude in Section 5.

2 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.)

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. This is natural as votes may provide inadequate information for a voting system to always pick a single winner (e.q., due to symmetry or due to the fact that a voting rule is so restrictive as to require some sort of a consensus for a decision to be made). In real-life elections there are elaborate rules for dealing with such situations. Here we disregard tie-breaking rules by focusing on the so-called unique winner model (see the next section). However, we point the reader to [Obraztsova *et al.*, 2011] for a discussion regarding the influence of tie-breaking for the case of election manipulation problem.

Let $E = (C, V)$ be an election. For each candidate $c \in C$, we define c's plurality score $score_E^p(c)$ to be the number of voters in V that rank c first. Candidates with highest plurality scores are plurality winners. Under approval voting, the score of candidate $c \in C$, $score_E^a(c)$, is the number of voters that approve of c. Again, candidates with highest scores are winners.

Another, perhaps more involved, election system is *Condorcet's rule*, in which a candidate $c \in C$ is a winner if and only if for each $c' \in C \setminus \{c\}$, more than half of the voters prefer c to 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 $c' \in C \setminus \{c\}$.

Computational Complexity. We assume that the reader is familiar with the basic notions of complexity theory, including such notions as NP and NP-completeness. 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. To reduce counting problems to each other, we use the notion of a parsimonious reduction.

Definition 2. *Let $\#A$ and $\#B$ be two counting problems. We say that $\#A$ parsimoniously reduces to $\#B$ if there exists a polynomial-time computable function f such that for each instance I of $\#A$ the following two conditions hold:*

1. *$f(I)$ is an instance of $\#B$, and*
2. *I has exactly as many solutions as $f(I)$.*

We say that a problem is #P-parsimonious-complete if it belongs to #P and every #P-problem parsimoniously reduces to it. For example, #X3C is #P-parsimonious-complete. Throughout this paper we will write #P-complete to mean #P-parsimonious-complete. We should mention, however, that different authors sometimes use different reduction types to define #P-completeness. For example, Valiant [1979] used Turing reductions, Zankó [1991] used many-one reductions, and Krentel [1988] used metric reductions.

The class of functions computable in polynomial time is called FP. Thus, if a given counting problem can be solved in polynomial time then we will write that it is in FP.

3 Counting Variants of Control Problems

Let us now formally define the counting variants of the election control problems. We are interested in four types of control: control by adding candidates (AC), control by deleting candidates (DC), control by adding voters (AV), and control

by deleting voters (DV). For each of the problems we consider its constructive variant (CC) and its destructive variant (DC). We now formally define the counting variant of constructive control by adding voters and then explain informally how the counting variants of other control problems are defined. As is typical for computational study of control problems, we assume the *unique-winner* model.

Definition 3. *Let R be a voting system. In the counting variant of constructive control by adding voters problem for R (R-#CCAV) we are given a set of candidates C, a set of registered voters V, a set of unregistered voters W, a designated candidate $p \in C$, and a natural number k. 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 $|W'| \leq k$.*

Constructive control by deleting voters (#CCDV) is defined analogously, but we do not have W in the input and we ask how many sets V', $V' \subseteq V$, are there such that p is the unique R-winner of $(C, V \setminus V')$ and $V' \leq k$.

In the constructive control by adding candidates (#CCAC) and the constructive control by deleting candidates (#CCDC) problems the set of voters is fixed but we can vary the set of candidates. In #CCAC we are given an additional set A of unregistered candidates and we ask for how many sets $A' \subseteq A$ of size up to k it holds that p is the unique winner of election $(C \cup A', V)$ (naturally, we assume that the voters have preferences over all candidates in $C \cup A$). In #CCDC we ask how many subsets C' of C are there of size up to k such that p is the unique winner of election $(C \setminus C', V)$.[1]

Destructive variants of our problems are defined analogously, except that we ask for the number of settings where the designated candidate—who in this case is called the despised candidate—is not the unique winner of the election.

Counting variants of control problems are interesting in their own right, but we focus on them because they allow us to model winner prediction problems for settings where the structure of the election is uncertain. We now describe one example scenario, pertaining to #CCAV; the reader can imagine analogous settings for the remaining types of control.

Let us assume that set C of candidates participating in the election is 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 some probability distribution P on the number of voters from W that will participate in the election (from our perspective, each equal-sized subset of voters from W is equally likely; different-sized sets may, of course, have different probabilities of participating in the election).

In other words, for each i, $0 \leq i \leq |W|$, let $P(i)$ be the probability that i voters from W join the election (and assume that we have an easy way of computing this value) and let $Q(i)$ be the probability that a designated candidate p wins under the condition that exactly i voters from W participate (assuming that each i-element subset of W is equally likely). Then, the probability that p wins is simply given by:

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

To compute $Q(i)$, we have to compute for how many sets W, of size exactly i, candidate p wins, and divide it by $\binom{|W|}{i}$. To compute for how many sets of size exactly i candidate p wins, we solve the corresponding #CCAV problem for adding at most i voters from W, then for adding at most $i-1$ voters from W, and then we subtract the results.

[1]Formally, we forbid C' from containing p. In the constructive setting this follows from the definition but in the destructive one we have to assume it separately.

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

Problem	Plurality	Approval	Condorcet
#CCAC	#P-com	–	–
#DCAC	#P-com	FP	FP
#CCDC	#P-com	FP	FP
#DCDC	#P-com	–	–
#CCAV	FP	#P-com	#P-com
#DCAV	FP	?	?
#CCDV	FP	#P-com	#P-com
#DCDV	FP	?	?

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. For the cases of #DCAV and #DCDV under approval voting and under Condorcet voting, we were able to show #P-metric-completeness but not #P-parsimonious-completeness.

4 Results

In this section we present our complexity results regarding counting variants of election control problems, focusing on positive, algorithmic results. We present a summary of our results in Table 1. In all constructive cases where a decision variant of a given problem is polynomial-time solvable, so is the counting variant. In all cases where a decision variant of a given problem is NP-complete, the counting variant is #P-complete. We do not present our #P-completeness proofs/theorems as they are mostly easy extensions of the constructions already present in the literature. Our #P-completeness results follow by reductions from #X3C.

4.1 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. We believe that our approach can be used for several other voting systems.[3]

[3]Most glaring example of such a rule would be veto. For example, under veto adding voters is essentially the same as deleting

Theorem 4. *Plurality-#CCAV is in* FP.

Proof. 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 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 $score^p_{(C,V)}(p) + j - 1$.

The pseudocode for our algorithm is given below.

PLURALITY-#CCAV(C, V, W, p, k)

if p is the unique winner of (C, V)
  then $k_0 := 0$
  else $k_0 := \max_{c_i \in C}(score^p_{(C,V)}(c_i) - score^p_{(C,V)}(p) + 1)$,
$result := 0$
for $j := k_0$ **to** $\min(|A_p|, k)$
  do $result := result + \binom{|A_p|}{j} \cdot count(C, V, W, p, k, j)$
return $result$

At the beginning, the algorithm computes 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 $\max_{c_i \in C}(score^p_{(C,V)}(c_i) - score^p_{(C,V)}(p) + 1)$. After we compute k_0, 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, $k_0 \leq j \leq \min(k, |A_p|)$ and 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 $score^p_{(C,V)}(p) + j - 1$.

voters under plurality.

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. □

Using this algorithm, we can easily derive one for the destructive setting.

Theorem 5. *Plurality-#DCAV is in* FP.

Proof. Let $I = (C, V, W, p, k)$ be an instance of plurality-#CCAV. There are exactly $\sum_{i=0}^{k} \binom{|W|}{i}$ sets W' such that $W' \subseteq W$ and $|W'| \leq k$. Of these, there are exactly PLURALITY-#CCAV(C, V, W, p, k) sets of voters whose inclusion in the election ensures that p is the unique winner. Thus, there are exactly

$$\sum_{i=0}^{k} \binom{|W|}{i} - \text{PLURALITY-\#CCAV}(C, V, W, p, k)$$

subsets of W of cardinality at most k whose inclusion in the election ensures that p is not the unique winner. Clearly, we can compute this value in polynomial time. □

Given the results for control by adding voters, it is not surprising that similar results hold for the case of deleting voters.

Theorem 6. *Plurality-#CCDV is in* FP.

Proof. Let $I = (C, V, p, k)$ be an instance of plurality-#CCDV, where $C = \{p, c_1, \ldots, c_{m-1}\}$ is the set of candidates, V is the set of voters, p is the designated candidate, and k is the upper bound on the number of voters that can be deleted. We will now give a polynomial-time algorithm that computes the number of solutions for I.

Let A_p be the subset of V containing those voters that rank p first. Similarly, for each $c_i \in C$, let A_{c_i} be the subset of voters that rank c_i first. For each integer j, $0 \leq j \leq k$, we define $count(C, V, p, k, j)$ to be the number of subsets V' of $V - A_p$ such that:

1. $|V'| \leq k - j$, and
2. in election $(C, V - V')$ each candidate $c_i \in C$ has score at most $score^p_{(C,V)}(p) - j - 1$.

The algorithm given below returns the number of solutions for I.

PLURALITY-#CCDV(C, V, p, k)

1 $result := 0$
2 **for** $j := 0$ **to** $\min(|A_p|, k)$
3 **do** $result := result + \binom{|A_p|}{j} \cdot count(C, V, p, k, j)$
4 **return** $result$

In each iteration of the main loop we consider deleting exactly j voters from A_p (there are $\binom{|A_p|}{j}$ ways to pick these j voters) . Assuming we remove from V exactly j members of A_p, we must also remove some number of voters from $V - A_0$, to make sure that p is the unique winner of the resulting election. The number of ways in which this can be achieved is $count(C, V, p, k, j)$. It is easy to verify that indeed our algorithm works correctly. It remains to show how to compute $count(C, V, p, k, j)$.

Let us fix some value j, $0 \leq j \leq \min(k, |A_p|)$. We will show how to compute $count(C, V, p, k, j)$. For each $c_i \in C$, we define:

$$l_i = \max\big(0, j + score^p_{(C,V)}(c_i) - score^p_{(C,V)}(p) + 1\big).$$

Intuitively, l_i is the minimal number of voters from A_{c_i} that need to be removed from the election for p to have score higher than c_i (assuming j voters from A_p have been already removed from the election).

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 $V' \subseteq A_{c_1} \cup A_{c_2} \cup \cdots \cup A_{c_i}$ such that $|V'| = t$ and each candidate $c_1, \ldots, c_i$ has score at most $score^p_{(C,V)}(p) - j - 1$ in election $(C, V - V')$. It is easy to see that $count(C, V, p, k, j) = \sum_{t=0}^{k-j} a_{t,m-1}$. Further, the following recursive relation holds:

$$a_{t,i} = \begin{cases} \sum_{s=l_i}^{\min(|A_{c_i}|,t)} \binom{|A_{c_i}|}{s} a_{t-s,i-1}, & \text{if } t \geq l_i,\ i > 1, \\ 0, & \text{if } t < l_i, \\ \binom{|A_1|}{t}, & \text{if } t \geq l_1,\ i = 1. \end{cases}$$

Thus, for each t, i we can compute $a_{t,i}$ in polynomial time using dynamic programming. As a result, we can compute $count(C, V, p, k, j)$ and the proof is complete. □

4.2 Approval Voting and Condorcet Voting

Let us now consider approval voting and Condorcet voting. While these two systems are very different in many respects, their behavior with respect to election control is very 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 7. *Both approval-#DCAC and Condorcet-#DCAC are in* FP.

Proof. We first consider the case of approval voting. Let $I = (C, A, V, p, k)$ be an instance of approval-#DCAC, where $C = \{p, c_1, \ldots, c_{m-1}\}$ is the set of registered candidates, $A = \{a_1, \ldots, a_{m'}\}$ is the set of additional candidates, V is the set of voters (with approval vectors over $C \cup A$), p is the designated candidate, and k is the upper bound on the number of candidates that we can add. We will give a polynomial-time algorithm that counts the number of up-to-k-element subsets A' of A such that p is not the unique winner of election $(C \cup A', V)$.

Let A_0 be the set of candidates in A that are approved by at least as many voters as p is. To ensure that p is not the unique winner of the election (assuming p is the unique winner prior to adding any candidates), it suffices to include at least one candidate from A_0. Thus, we have the following algorithm.

APPROVAL-#DCAC(C, A, V, p, k)

1 **if** p is not the unique winner of (C, V)
2 **then return** $\sum_{i=0}^{k} \binom{|A|}{i}$
3 Let A_0 be the set of candidates $a_i \in A$,
s.t. $score^a_{(C \cup A, V)}(a_i) \geq score^a_{(C \cup A, V)}(p)$.
4 $result := 0$
5 **for** $j := 1$ **to** k
6 **do** $result := result + \sum_{i=1}^{\min(|A_0|,j)} \binom{|A_0|}{i} \binom{|A - A_0|}{j-i}$
7 **return** $result$

The loop from line 5, for every j, counts the number of ways in which we can choose exactly j candidates from A; it can be done by first picking i of the candidates in A_0 (who beat p), and then $j - i$ of the candidates in $A - A_0$. It is clear that the algorithm is correct and runs in polynomial time.

Let us now move on to the case of Condorcet voting. It is easy to see that the same algorithm works correctly, provided that we make two changes: (a) in the first two lines, instead of testing if p is an approval winner we need to test if p is a Condorcet winner, and (b) we redefine the set A_0 to be the set of candidates $a_i \in A$ such that $N_{C \cup A}(p, a_i) \leq N_{C \cup A}(a_i, p)$. To see that these two changes suffice, it is enough to note that to ensure that p is not a Condorcet winner of the election we have to have that either p already is not a Condorcet winner (and then we can freely add any number of candidates), or we have to add at least one candidate from A_0. □

Theorem 8. *Both approval-#CCDC and Condorcet-#CCDC are in* FP.

Proof. Let us handle the case of approval voting first. 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. Based on this observation we provide the following simple algorithm.

APPROVAL-#CCDC(C, V, p, k)

1 Let k_0 be the number of candidates $c \in C - \{p\}$,
s.t. $score^a_{(C,V)}(c) \geq score_{(C,V)}(p)$.
2 **return** $\sum_{i=0}^{k-k_0} \binom{|C|-k_0-1}{i}$

Clearly, the algorithm is correct and runs in polynomial-time.

For the case of Condorcet voting, it suffices to note that if p is to be a winner, we have to delete all candidates $c \in C - \{p\}$ such that $N_{C,V}(p,c) \leq N_{C,V}(c,p)$. Thus, provided that we let k_0 be the number of candidates $c \in C - \{p\}$ such that $N_{C,V}(p,c) \leq N_{C,V}(c,p)$, the same algorithm as for the case of approval voting works for Condorcet voting. □

5 Conclusions and Future Work

We have considered a natural 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).

Following the paper of Hemaspaandra, Hemaspaandra, and Rothe [2007], we have considered three voting rules: plurality, approval, and Condorcet voting. It turned out that the complexity of counting the number of solutions for constructive control problems under these systems is analogous to the complexity of verifying if any solution exists. That is, whenever the decision variant of the constructive problem is in P, the counting variant is in FP; whenever the decision variant is NP-complete, the counting variant is #P-complete. While the latter is not too surprising, sometimes easy decision problems correspond to hard counting problems, and thus the former is less trivial. However, perhaps this behavior is due to the simplicity of the election systems we have considered. Thus, the most natural research direction currently is to study counting variants of control under further election systems.

Currently, we are working on results for simplified variant of Dodgson and for maximin. The former is interesting because it is used to efficiently approximate the Dodgson rule [Caragiannis *et al.*, 2010]. The latter is interesting because it is known that several constructive control problems are easy for it [Faliszewski *et al.*, 2011]. A more involved research direction is to consider more involved probability distributions of candidates/voters that join/leave the election.

Acknowledgements. We are very grateful to WSCAI referees for helpful, thorough reports.

References

[Bachrach *et al.*, 2010] Y. Bachrach, N. Betzler, and P. Faliszewski. Probabilistic possible winner determination. In *Proceedings of AAAI 2010*, pages 697–702, July 2010.

[Bartholdi *et al.*, 1989] J. Bartholdi, III, C. Tovey, and M. Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6(2):157–165, 1989.

[Bartholdi *et al.*, 1992] 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.

[Betzler and Dorn, 2009] N. Betzler and B. Dorn. Towards a dichotomy of finding possible winners in elections based on scoring rules. In *Proceedings of MFCS 2009*, pages 124–136, August 2009.

[Caragiannis *et al.*, 2010] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and A. Procaccia. Socially desirable approximations for Dodgson's voting rule. In *Proceedings of EC 2010*, pages 253–262, June 2010.

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

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

[Faliszewski *et al.*, 2011] P. Faliszewski, E. Hemaspaandra, and L. Hemaspaandra. Multimode attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011.

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

[Hemaspaandra *et al.*, 1997] E. Hemaspaandra, L. Hemaspaandra, and J. Rothe. Exact analysis of Dodgson elections: Lewis Carroll's 1876 voting system is complete for parallel access to NP. *Journal of the ACM*, 44(6):806–825, 1997.

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

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

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

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

[Obraztsova *et al.*, 2011] S. Obraztsova, E. Elkind, and N. Hazon. Ties matter: Complexity of voting manipulation revisited. In *Proceedings of AAMAS 2011*, 2011. To appear.

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

[Xia and Conitzer, 2008] L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *Proceedings of AAAI 2008*, pages 196–201, July 2008.

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