Probabilistic Possible Winner Determination

Yoram Bachrach
Microsoft Research Ltd.
Cambridge, UK

Nadja Betzler[*]
Institut für Informatik
Friedrich-Schiller-Universität Jena
Jena, Germany

Piotr Faliszewski[†]
Department of Computer Science
AGH Univ. of Science and Technology
Kraków, Poland

Abstract

We study the computational complexity of the counting version of the POSSIBLE-WINNER problem for elections. In the POSSIBLE-WINNER problem we are given a profile of voters, each with a partial preference order, and ask if there are linear extensions of the votes such that a designated candidate wins. We also analyze a special case of POSSIBLE-WINNER, the MANIPULATION problem. We provide polynomial-time algorithms for counting manipulations in a class of scoring protocols and in several other voting rules. We show #P-hardness of the counting variant of POSSIBLE-WINNER for plurality and veto and give a simple yet general and practically useful randomized algorithm for a variant of POSSIBLE-WINNER for all voting rules for which a winner can be computed in polynomial time.

Introduction

Voting and elections are natural tools when a group of self-interested agents must come up with a joint decision that will be satisfactory to all of them, or, at least, will take each agent's opinion into account; see (Ephrati and Rosenschein 1997) for a classic example. A key issue is the determination of a winner (or a set of winners) which is accomplished by applying a voting rule. There are many different voting rules coming with different advantages and disadvantages.

In the standard election model, one has a set of voters and a set of candidates, and every voter provides a linear order of all candidates according to his or her preferences. However, in many realistic settings some of the voters may only provide partial instead of linear orders due to various reasons. One way to deal with such situations is to consider information obtained from extensions of the partial orders. For example, given a set of partial votes, it may be important to know which candidates still have a chance of winning or which candidate necessarily is a winner, irrespective of how votes will be completed. The corresponding problems, called POSSIBLE-WINNER and NECESSARY-WINNER, were introduced by Konczak and Lang (2005) and, e.g., studied by (Lang et al. 2007; Walsh 2007; Xia and Conitzer 2008a; Betzler, Hemmann, and Niedermeier 2009).

In this work, we go one step further and propose a quantitative approach which instead of just investigating the existence of a "winning extension" counts all possible extensions leading to a designated candidate's victory. This allows for a better comparison of the candidates. For example, there might be situations in which all candidates are possible winners but one candidate wins in most of the extensions. We provide a simple model for computing the probability with which a designated candidate wins, assuming that the voters choose their votes' extensions uniformly at random (a variant of the impartial culture assumption). If we do not have any additional knowledge, this is our best guess. An interesting approach might be to learn a distribution of votes from those seen so far (see (Hazon et al. 2008) for a related piece of research). However, this is not always feasible. For example, we might only see aggregated results (e.g., candidates' scores) of those votes that have already been fully specified. All our results hold in this limited setting.

Besides focusing on the counting version of POSSIBLE-WINNER in general, we take a closer look at the counting variant of a prominent special case, the MANIPULATION problem. Herein, we have two sets of voters, one with linear orders and one with completely unspecified orders. In contrast to our motivation, the MANIPULATION problem originates from the "negative" view that a group of voters, the coalition, might have the incentive to vote differently from their sincere orders to obtain a more desirable outcome for the whole coalition. Such a process is undesirable but cannot be avoided for any non-trivial voting system (Gibbard-Satterthwaite theorem). Bartholdi, Tovey, and Trick (1989) proposed computational hardness (focussing on worst case analysis) as a way out. They argued that if MANIPULATION for a given voting system is computationally hard, then for practical purposes this system is resistant to manipulation. In recent years, researchers have throughly expanded this approach (see (Walsh 2007; Conitzer, Sandholm, and Lang 2007; Faliszewski, Hemaspaandra, and Schnoor 2008; Xia et al. 2009) for a few examples) and analyzed its limits (most notably, via the approach of Friedgut, Kalai, and Nisan (2008); see also (Xia and Conitzer 2008b)).

[*]Supported by the DFG, research project PAWS, NI 369/10.

[†]Supported by AGH University of Technology Grant no. 11.11.120.865, by Polish Ministry of Science and Higher Education grant N-N206-378637, and by Foundation for Polish Science's program Homing/Powroty.

Our analysis is somewhat similar to the work on voting in cooperative game theory. For example, in *Weighted Voting Games (WVGs)* each agent has a weight, representing a number of votes, and a coalition (agent subset) wins if the sum of its members' weights exceeds a threshold. In such games, so-called power indices measure the influence each agent has on the outcome. These indices can be viewed as the probability of an agent to be pivotal in a random coalition, in a way that is similar to the impartial culture assumption we use. Computing such indices in WVGs is #P-Complete (Deng and Papadimitriou 1994), but recent work (Bachrach et al. 2008) approximates them using algorithms similar to our Theorem 6.

Our main results are as follows: (a) We provide a polynomial-time algorithm that counts voting manipulations for an interesting class of scoring protocols. (b) We show that the counting variant of POSSIBLE-WINNER is #P-hard, even for simple rules (e.g., plurality) and simple partial votes (e.g., votes with only a single undetermined pair of candidates per vote). (c) We provide a randomized, approximation algorithm for the counting variant of POSSIBLE-WINNER and thus, for the counting variant of MANIPULATION as well. Several (parts of) proofs have to be omitted due to space restrictions.

Preliminaries

Elections. An election consists of a set $C = \{c_1, \ldots, c_m\}$ of candidates and a collection $V = (v_1, \ldots, v_n)$ of voters over C. Each voter v_i, $1 \leq i \leq n$, has a linear order $>$ over C (often called a *preference order*). Intuitively, these orders represent preferences of the voters. For example, for $C = \{c_1, c_2, c_3\}$, the order $c_3 > c_2 > c_1$ says that c_3 is the best candidate, c_1 is the worst, and c_2 is ranked in between. Sometimes, when specifying a preference order we write $c > D$, where c is some candidate and D is a subset of candidates. This means that c is ranked ahead of each member of D and members of D are ranked in an arbitrary but fixed order. Typically, we will be interested in the performance of some designated candidate, denoted p.

A *voting rule* R (or, a *social choice correspondence* R) is a function such that given an election (C, V), $R(C, V) = W \subseteq C$ is a set of this election's winners. A *scoring protocol* for elections with m candidates is a voting rule represented by a *vector* α of m nonnegative, nonincreasing integers $(\alpha_1, \ldots, \alpha_m)$, where each candidate c receives α_j points for each voter that ranks c on position j in his or her preference order; the candidates with most points win. Prominent families of scoring protocols (with one scoring protocol per each number of candidates) include *plurality* (with vectors $(1, 0, \ldots, 0)$), *veto* (with vectors $(1, \ldots, 1, 0)$), and *Borda* (with vectors $(m-1, m-2, \ldots, 0)$, where m is the number of candidates), *k-approval* (with vectors $(1^k, 0^{m-k})$), and *k-veto* (with vectors $(1^{m-k}, 0^k)$) where 1^i and 0^i mean, respectively, i repetitions of 1 and i repetitions of 0.

Possible Winners and Manipulation. Let $C = \{c_1, \ldots, c_m\}$ be a set of candidates. A *partial order* on C is a reflexive, transitive, and antisymmetric relation on C. A linear order $>$ *extends* a partial order $\succ$ ($>$ is a *linear extension* of $\succ$) if for each $c_i, c_j \in C$ it holds that $c_i \succ c_j$ implies $c_i > c_j$. Given a collection $V = (v_1, \ldots, v_n)$ of voters with partial preference orders, a linear extension of V is a collection $V' = (v'_1, \ldots, v'_n)$ of voters with linear orders such that for each i, $1 \leq i \leq n$, preference order of v'_i is a linear extension of the partial order of v_i.

Let R be a voting rule. Given a set of candidates C, a collection of voters V (where voters may have partial preference orders) and a designated candidate $p \in C$, the R-POSSIBLE-WINNER problem asks, whether there is a linear extension V' of V such that $R(C, V') = \{p\}$.[1] That is, we ask if p is a *possible winner* of (C, V). In this paper we are interested in the following generalization of the POSSIBLE-WINNER problem.

Definition 1. *Let R be a voting rule. Given a set of candidates C, a collection V of voters with partial preference orders over C, and a candidate $p \in C$, the R-#POSSIBLE-WINNER problem asks how many linear extensions V' of V there are such that $R(C, V') = \{p\}$.*

An important special case of the POSSIBLE-WINNER problem is the coalitional manipulation problem (in short, MANIPULATION), where the voters are divided into *nonmanipulators* with fully specified linear preference orders, and *manipulators* whose preference orders are completely unspecified. The counting version of the MANIPULATION problem, #MANIPULATION, is the analogous special case of #POSSIBLE-WINNER.

Computational Complexity. We assume familiarity with standard notions of complexity theory such as classes P, NP, and NP-completeness. The class #P is the counting analog of the class NP, where instead of asking if a solution exists, we ask how many solutions exist. Throughout this paper we use the most typical notion of #P-hardness, introduced by Valiant (1979). Basically, a counting problem A is #P-hard if for each problem B in #P there is an algorithm that solves B in polynomial time given oracle access to A.

Counting the Number of Manipulations

In this section, we consider the #MANIPULATION problem. For a number of voting rules, e.g., STV (Bartholdi and Orlin 1991), Copeland (Faliszewski, Hemaspaandra, and Schnoor 2008), and maximin (Xia et al. 2009), the decision version of MANIPULATION is NP-hard and, in consequence, so is the counting variant. Here, we focus on rules for which the decision version of MANIPULATION is in P. We first examine #MANIPULATION in general and then consider the case of a single manipulator (for a wider range of voting rules).

Unbounded number of manipulators. We show that for a practically important family of scoring protocols including plurality, veto, and many others, #MANIPULATION is in P.

Definition 2. *A scoring protocol $\alpha = (\alpha_1, \ldots, \alpha_m)$ is k-valued if there is a nonnegative integer β and a sequence*

[1] That is, we assume the unique-winner model. Otherwise, in the *co-winner* model, one only requires that p is one of the winners. All our results can be easily adapted to work for the co-winner case.

of nonincreasing integers $\beta_1, \ldots, \beta_m$ such that $\alpha = (\beta + \beta_1, \ldots, \beta + \beta_m)$ and all but k of $\beta_1, \ldots, \beta_m$ are zero.

Clearly, plurality and veto are both defined via 1-valued scoring protocols and, for each positive integer k, k-approval and k-veto are defined via k-valued scoring protocols. An example of a 2-valued scoring protocol with $\beta = 1$ is $(2, 1, \ldots, 1, 0)$.

Theorem 1. *Let k be a constant positive integer. For each voting rule R defined via a polynomial-time computable family of k-valued scoring protocols, R-#MANIPULATION is in* P.

Proof (sketch). Let (C, V) be an instance of #MANIPULATION where $C = \{p, c_1, \ldots, c_m\}$ and $V = (v_1, \ldots v_{n'}, w_1, \ldots, w_n)$ such that the voters $v_1, \ldots, v_{n'}$ have fully specified preference orders, and the voters $w_1, \ldots, w_n$ are the manipulators, having completely unspecified preference orders.

Our algorithm first computes (by assumption, in polynomial time) the scoring protocol α for $m + 1$ candidates. Since α is k-valued, it is easy to convert it to the form $B = (\beta+\beta_0, \ldots, \beta+\beta_m)$ fulfilling the requirements of Definition 2. Due to the form of B, there are exactly k positions in each preference order, $p_1, \ldots, p_k$, that receive amounts of points different from β. Let $\mathcal{P} = \{p_1, \ldots, p_k\}$ and, for each $p_i \in \mathcal{P}$, let δ_i be the number of points a candidate receives for being ranked in position p_i.

For each candidate $c \in C$, compute $s(c)$, the number of points that c receives from voters $v_1, \ldots v_{n'}$. For each nonnegative integer ℓ, we define $F(\ell)$ to be the number of ways in which the voters $w_1, \ldots, w_n$ can choose their preference orders such that in the resulting election (C, V) (a) p receives exactly ℓ points, and (b) no other candidate receives ℓ points or more. Naturally, our algorithm should output $\sum_{\ell=s(p)}^{s(p)+n\cdot(\beta+\beta_0)} F(\ell)$.[2] It remains to show that $F(\ell)$ can be computed in polynomial time.

We show how to compute $F(\ell)$, for any fixed $\ell \in \{s(p), \ldots, n \cdot (\beta + \beta_0)\}$. The "relevant part" of the votes $w_1, \ldots, w_n$ is represented by an $n \times k$ table (called *vote table*), where each row corresponds to a vote, and each column corresponds to one of the positions in $\mathcal{P}$. Each candidate appears at most once in each row. Intuitively, our algorithm will place the candidates p, $c_1, \ldots, c_m$, one by one in the table in a way such that p gets exactly ℓ points and every other candidate at most $\ell - 1$ points (keeping in mind that for each row of the table where a candidate is not placed, this candidate gets β points). It is important to describe the situation in the vote table after we have processed a given subset of candidates. A *vote table situation* is a function g from all subsets P from $\mathcal{P}$ to integers from $[0, \ldots, n]$ with $\sum_{P\subseteq\mathcal{P}} g(P) = n$. For a subset P of $\mathcal{P}$, $g(P)$ is the number of rows in the vote table where exactly the positions from P are unoccupied by the already processed set of candidates. The condition $\sum_{P\subseteq\mathcal{P}} g(P) = n$ is necessary since there are n manipulators. Clearly, there are less than $(n + 1)^{2^k}$ vote table situations, a polynomial for constant k.

We compute $F(\ell)$ using dynamic programming. For each $j \in \{0, \ldots, m\}$, and for each vote table situation g, we define $f(j, g)$ to be the number of ways in which we can place the candidates p, $c_1, \ldots, c_j$ in the vote table so that p has exactly ℓ points, $c_1, \ldots, c_j$ have at most $\ell - 1$ points each (including the points they get from being ranked at other positions than those in $\mathcal{P}$), and for each $P \subseteq \mathcal{P}$ the number of rows in the vote table with exactly the positions from P unoccupied is $g(P)$.

Initialization: We compute $f(0, g)$ for each vote table situation g as follows. Since for $f(0, g)$ the only candidate that can occupy a position of $\mathcal{P}$ is p, all entries for a vote table situation g with $g(P) \neq 0$ for any $P \subsetneq \mathcal{P}$ not of the form $\mathcal{P} - \{p_i\}$, $1 \leq i \leq k$, are initialized with zero. For all remaining vote table situations g, if $s(p) + \sum_{p_i\in\mathcal{P}} \delta_i g(\mathcal{P} - \{p_i\}) + g(\mathcal{P}) \cdot \beta \neq \ell$ (that is, if p does not receive exactly ℓ points), then set $f(0, g) = 0$; otherwise set $f(0, g) = \prod_{p_i\in\mathcal{P}} \binom{(n-t_i)}{g(\mathcal{P}-\{p_i\})}$ with $t_i = \sum_{j=1}^{i-1} g(\mathcal{P} - \{p_j\})$ (that is, the number of ways by which the vote table situation g can be realized by n manipulators).

Update: To complete the description of our dynamic programming algorithm, it remains to show how to compute $f(j, g)$, where g is a vote table situation and $1 \leq j \leq m$, given the values $f(j - 1, h)$ for each vote table situation h. To do so, we define a *candidate placement* to be a function t such that for each subset $P \subseteq \mathcal{P}$, and each position $p_i \in \mathcal{P}$, $t(P, p_i)$ denotes the number of votes in which the current candidate c_j is placed at position p_i and after this all positions from $P - p_i$ are unoccupied. Clearly, for a specific vote table situation not all candidates placements are possible; this will be checked below. Since $t(P, p_i)$ is an integer between 0 and n, the number of candidate placements is bounded from above by $(n+1)^{k2^k}$, a polynomial for constant k. We initialize $f(j, g)$ with the value 0 and for each candidate placement t we do the following:

1. Set $q = n - \sum_{P\subseteq\mathcal{P}} \sum_{p_i\in P} t(P, p_i)$; q is the number of votes in which c_j does not assume any position from $\mathcal{P}$. If $q < 0$ then jump to the next t.
2. Compute $s = s(c_j) + \beta q + \sum_{P\subseteq\mathcal{P}} \sum_{p_i\in P} \delta_i t(P, p_i)$; the score that c_j would obtain from this placement. If $s \geq \ell$ then jump to the next t.
3. Compute a vote table situation h such that for each $P \subseteq \mathcal{P}$, $g(P) = h(P) - \sum_{p_i\in P} t(P, p_i) + \sum_{p_i\in\mathcal{P}-P}(P \cup \{p_i\}, p_i)$. If $h(P) < 0$ for some P, then jump to the next t. Intuitively, h is the vote table situation that leads to g via placing candidates according to t.
4. Add to $f(j, g)$ a value $K \cdot f(j - 1, h)$, where K is the number of ways in which a vote table corresponding to h can be transformed to a vote table corresponding to g by adding at most one candidate per vote. It is not hard to see that K can be computed in polynomial time.

The value $F(\ell)$ is equal to $((m - k)!)^n \sum_g f(m, g)$, that is, the sum of values $f(m, g)$ for each vote table situation g, times $((m - k)!)^n$. The $((m - k)!)^n$ factor is necessary

[2] If either β or β_0 are not polynomially bounded in the input size, this summation would require a superpolynomial number of steps. Fortunately, one can easily identify at most $(n+1)^{k+1}$ values of ℓ for which $F(\ell)$ is nonzero. We sum over these values of ℓ only.

to count all ways in which the candidates in positions other than $\mathcal{P}$ are placed. This finishes the proof sketch. □

As an immediate corollary, we have that #MANIPULATION is in P for plurality, veto, and—for each fixed k—for k-approval and k-veto. Our algorithm is somewhat complicated because we use vote table situations instead of simply keeping track of how many votes have each position free. However, the simpler approach would not guarantee that each candidate appears in each vote exactly once.

Now, we consider other voting rules for which MANIPULATION can be decided in polynomial time.

Theorem 2. #MANIPULATION *is in* P *for plurality with run-off and for the cup voting rule with a fixed agenda.*

We omit the proof and the definitions of the voting rules. Briefly speking, plurality with run-off is a two-round election system where we first pick two candidates with highest plurality scores, and—among them—select the one preferred by most voters. (We assume that ties in the first round are broken lexicographically.) Our algorithm is conceptually similar to that from the proof of Theorem 1. For the cup rule, we use the definition from (Conitzer, Sandholm, and Lang 2007) and our counting algorithm is an extension of the decision algorithm given there.

Single Manipulator. Bartholdi, Tovey, and Trick (1989) investigated the computational complexity of MANIPULATION for the case of a single manipulator. They gave a natural greedy algorithm solving the decision variant of single-manipulator MANIPULATION for many voting rules. It seems difficult to provide such a general algorithm for the counting version of the problem. However, we provide algorithms for several interesting voting rules. Note that there are voting rules for which single-manipulator MANIPULATION is NP-complete; see, e.g., (Bartholdi and Orlin 1991).

Theorem 3. *For every voting rule defined via a polynomial-time computable family of scoring protocols,* #MANIPULATION *for the case of a single manipulator is in* P.

Proof. Let $E = (C, V)$ be our input election where $C = \{p, c_1, \ldots, c_m\}$, and $V = (v_1, \ldots, v_n, w)$. Our designated candidate is p and the single manipulator is w. We define $F(k)$ to be the number of preference orders that w can submit such that w ranks p at position k and p is the unique winner of the election. Our algorithm outputs $\sum_{k=1}^{m+1} F(k)$ and it remains to show that $F(k)$ is computable in polynomial time. We do so in the remainder of this proof.

For any $k \in \{1, \ldots, m+1\}$, compute $F(k)$ as follows. For $1 \leq j \leq m$, let $\ell_j = j$ if $j < k$ and let $\ell_j = j + 1$, otherwise. For each $j \in \{1, \ldots, m\}$, define C_j to be the set of candidates $c \in C - \{p\}$ such that if w puts c at position ℓ_j and p at position k, then, in the resulting election, p has more points than c. Since the entries of a scoring vector are nonincreasing by definition, $C_j \subseteq C_{j+1}$ for each $j \in \{1, \ldots, m\}$.

If p is the unique winner after being ranked at position k by w, then each remaining position ℓ_j, $1 \leq j \leq m$, of w's vote has to contain a member of C_j. Clearly, this is only possible if for each j, $1 \leq j \leq m$, it holds that $\|C_j\| \geq j$. Then, the number of ways of filling w's vote satisfying this constraint is exactly $F(k) := \prod_{j=1}^{m} (\|C_j\| - (j-1))$. □

Additionally, we can show that single-manipulator #MANIPULATION is in P for Bucklin (see (Xia et al. 2009) for the definition of Bucklin; proof is omitted due to space restrictions). In contrast, for some other voting system, such as Copeland, for which the decision variant can be solved in polynomial time by the greedy algorithm from Bartholdi, Tovey, and Trick (1989), the computational complexity of the counting variant is elusive.

Counting Version of Possible Winner

Let us now move on to the POSSIBLE-WINNER problem, where each voter is allowed to have an arbitrary partial order. For the decision variant, the only scoring rules for which it is known that POSSIBLE-WINNER is in P are plurality and veto while for almost all other natural scoring rules it becomes NP-complete (Betzler and Dorn 2009). We will see that even with modest assumptions regarding the type of partial order required, #POSSIBLE-WINNER is #P-hard even for plurality and veto. We complement this result by providing a randomized, approximation algorithm for #POSSIBLE-WINNER problem that works for polynomial-time computable rules.

Negative Results. A direct way to show #P-hardness of #POSSIBLE-WINNER even for plurality is based on the fact that counting the linear extensions of a partial order is #P-complete; see (Brightwell and Winkler 1991).

Proposition 4. *Plurality-*#POSSIBLE-WINNER *is* #P*-hard even for profiles consisting of a single partial vote, and veto-*#POSSIBLE-WINNER *is* #P*-hard even for profiles with one partial vote and an unbounded number of linear votes.*

Proof. We give the proof for plurality and omit the similar proof for veto. Given a partially ordered set P over a set C, construct a plurality-#POSSIBLE-WINNER instance by identifying the elements of C with the candidates and adding one new candidate, p, such that p is preferred to all other candidates. Clearly the number of extensions in which p wins is exactly the number of linear extensions of P. □

Intuitively, the above reduction is unsatisfying. Often we want to count the number of linear extensions to see how frequently our designated candidate wins, but in the constructed profile p *always* wins. Also, the reduction relies on constructing a partial vote with an unbounded number of undetermined comparisons. The next theorem avoids these issues.

Theorem 5. *For plurality and for veto,* #POSSIBLE-WINNER *is* #P*-hard even if there is at most one undetermined pair of candidates in every partial vote.*

Proof. We consider the case of plurality and omit the case of veto which follows via a simple adaptation.

For a bipartite graph, a *perfect matching* is a subset of edges such that every vertex is part of exactly one edge. In #PERFECT-BIPARTITE-MATCHING we are given a bipartite graph $G = (V, U; E)$ with $U = \{u_1, \ldots, u_t\}$,

$V = \{v_1, \ldots, v_t\}$, and $E \subseteq U \times V$ and ask how many perfect matchings this graph has. Our reduction is from #PERFECT-BIPARTITE-MATCHING, which is well-known to be #P-complete (Valiant 1979).

Let $G = (V, U; E)$ be an instance of #PERFECT-BIPARTITE-MATCHING. For each $v_i \in V$, let $d(v_i)$ be the degree of v_i and $d = \max_{v_i \in V} d(v_i)$. For each $v_i \in V$, let $N(v_i) = \{n_1(v_i), \ldots, n_{d(v_i)}(v_i)\}$ denote v_i's neighbors.

We construct a plurality-#POSSIBLE-WINNER instance (C, P, p) as follows. The set of candidates is $C = \{p\} \cup \{v_1, \ldots, v_t\} \cup \{u_1, \ldots, u_t\}$ where p is the designated candidate. The collection P of votes consists of a set of linear votes and a set of partial votes. The linear votes are as follows: (a) There are d votes with $p > C\backslash\{p\}$, (b) for every $v_i \in V$, there are $d - d(v_i)$ votes with $v_i > C\backslash\{v_i\}$, and (c) for every $u_j \in U$, there are $d - 2$ votes with $u_j > C\backslash\{u_j\}$.

The partial votes are as follows. For each $v_i \in V$ and for $1 \leq j \leq d(v_i)$, we have a voter t_{ij} with partial order $\succ$ such that $v_i \succ C\backslash\{v_i, n_j(v_i)\}, n_j(v_i) \succ C\backslash\{v_i, n_j(v_i)\}$. That is, in t_{ij} one can put either v_i first and $n_j(v_i)$ second or the other way round, and the other candidates are fixed.

Claim: For each perfect matching there is a corresponding extension in which p wins and for every extension in which p wins there is a corresponding perfect matching.

"$\Rightarrow$": Let $E' \subseteq E$ denote a solution for the matching-instance. Then extend the partial votes as follows. For every $\{v_i, n_j(v_i)\} \in E'$ extend the partial vote t_{ij} to $n_j(v_i) > v_i > C\backslash\{v_i, n_j(v_i)\}$ and for every $\{v_i, n_j(v_i)\} \notin E'$ extend t_{ij} to $v_i > n_j(v_i) > C\backslash\{v_i, n_j(v_i)\}$. This gives the following situation. The candidate p scores d points. Each $v_i \in V$ score $d - d(v_i)$ points in the linear votes and $d(v_i) - 1$ points in the partial votes. Each $u_j \in U$ scores $d - 2$ points in the linear votes and exactly one point in the partial votes (more precisely, in the vote that corresponds to its matching-edge). Thus, p is the unique winner.

"$\Leftarrow$": Consider an extension in which p wins. We now show that for every subset of partial votes that corresponds to a vertex v_i, there is exactly one vertex u_j that takes a first position and that v_i and u_j "match" in the bipartite graph. We consider the partial votes in more detail. The number of partial votes is $\sum_{i=1}^{t} d(v_i)$ and, in every partial vote, one candidate must score one point. Every candidate $u_j \in U$ can score at most one point and every candidate $v_i \in V$ can score at most $d(v_i) - 1$ points. Thus, in total all candidates together can score $t + \sum_{i=1}^{t}(d(v_i) - 1) = \sum_{i=1}^{t} d(v_i)$ points. Hence every candidate must end up with exactly $d-1$ points, otherwise it or another candidate would beat-or-tie p. In particular, for every $v_i \in V$, in the set of corresponding partial votes, candidate v_i must score one point in all but one of these votes. In the remaining vote one candidate $u \in N(v_i)$ must score one point. Since u does not beat-or-tie p, this is the only vote in which u takes the first position, i.e. vertex u "matches" the vertex v_i in the matching instance.

The correspondence between extensions and matchings is 1-to-1, thus the number of extensions where p wins equals the number of perfect matchings of G. □

Randomized Approximation Algorithm. Despite Theorem 5, in this section we give a randomized approximation algorithm for the problem of computing a proportion of extensions of a profile where our designated candidate wins.

Theorem 6. *Let R be a polynomial-time computable voting rule. There is an algorithm that, given an instance (C, V, p) of R-#POSSIBLE-WINNER and two positive rational numbers ϵ and δ, outputs in time polynomial in $\|C\|$, $\|V\|$, $\frac{1}{\varepsilon}$, and $\ln \frac{1}{\delta}$ a value $\widetilde{\alpha}$ such that the proportion α of the number of linear extensions of V where p is the unique R-winner to the number of all linear extensions of V is in the interval $[\widetilde{\alpha} - \varepsilon, \widetilde{\alpha} + \varepsilon]$ with probability greater or equal $1 - \delta$.*

Proof. Let the notation be as in the statement of the theorem. Our sampling algorithm is given below (PWP stands for "possible-winner proportion").

procedure R-PWP(C , V, p)
begin
  $r = \left\lceil \frac{\ln \frac{2}{\delta}}{2\,\epsilon^2} \right\rceil$, $X = 0$
  for $i = 1$ **to** r:
    choose a random linear extension V' of V
    if $R(V') = \{p\}$ **then** $X = X + 1$
  return $\frac{X}{r}$.
end

We claim the algorithm works in polynomial time with respect to $\|C\|$, $\|V\|$, $\frac{1}{\varepsilon}$, and $\ln \frac{1}{\delta}$. This is easy to see for all steps except choosing uniformly at random the linear extension V' of V. Doing so seems far from trivial, but due to Huber (2006) it is possible to uniformly sample linear extensions of partial orders and so we can sample linear extensions of collections of voters easily (for a short survey of early linear extensions sampling algorithms also see (Brightwell and Winkler 1991)).

We now explain why our algorithm is correct and our reason for our chosen number r of sampling iterations. Let $\mathcal{V}$ be the set of all linear extensions of V and let $\mathcal{V}_p$ be the set of all linear extensions of V where p is the unique winner according to R. Let $\alpha = \frac{\|\mathcal{V}_p\|}{\|\mathcal{V}\|}$. We seek to approximate α. Consider the i'th iteration in the algorithm. We view the sampling of the linear extension as a Bernoulli trial which is successful exactly if the sampled profile is in $\mathcal{V}_p$. Let X_i be the random variable where $X_i = 1$ if we choose a profile from $\mathcal{V}_p$ in the i'th iteration and $X_i = 0$ otherwise. Clearly, $E[X_i] = \alpha$. Our algorithm computes the random variable $X = \sum_{i=1}^{r} X_i$. It is easy to see that X has a binomial distribution of r trials with probability α of success, so $X \sim B(r, \alpha)$. It is known that $\frac{X}{r}$ is an unbiased, maximum likelihood estimator for α, and in particular $\frac{E[X]}{r} = \alpha$. To show that our algorithm achieves the desired accuracy and confidence, we must show that the number of samples we choose is sufficient. It holds that $\Pr\left(\left|\frac{X}{r} - \frac{\mathrm{E}[X]}{r}\right| \geq \varepsilon\right) \leq 2e^{-2r\varepsilon^2}$ (Hoeffing's inequality (1963)). Substituting $\frac{\ln \frac{2}{\delta}}{2\,\epsilon^2}$ for r, we get $\Pr\left(\left|\frac{X}{r} - \frac{\mathrm{E}[X]}{r}\right| \geq \varepsilon\right) \leq \delta$, which is what we want. This completes the proof. □

The work of Huber (2006) (and some of the works referenced there) provide a fully polynomial-time randomized approximation scheme for counting linear extensions, so the above algorithm can approximate R-#POSSIBLE-WINNER. However, it might give poor results due to the *additive* nature of the approximations in Theorem 6.

On the positive side, in many situations we are more interested in proportions as produced by our algorithm than in the exact counts of linear extensions. After all, often we want to know which candidate is most likely to win and how big her advantage is (assuming a variant of impartial culture).

Also, our algorithm constitutes a natural sampling-based solution to the decision variant of the POSSIBLE-WINNER problem. If our algorithm produces a nonzero answer, we know that the designated candidate is a possible winner, and if our algorithm outputs zero, we know that either our candidate is not a possible winner, or there are very few extensions of the votes that lead to him or her being the winner. In a sense, this is an algorithmic counterpart of the results on frequency of hardness given by Friedgut, Kalai, and Nisan (2008), and others, e.g., (Xia and Conitzer 2008b).

Conclusions and Open Problems

We provided polynomial-time algorithms for counting manipulations in various voting systems. In contrast, we showed that #POSSIBLE-WINNER is #P-hard even for plurality and veto for which the decision version is in P. Nonetheless, we provided a randomized algorithm for a practically useful variant of #POSSIBLE-WINNER. Our algorithm is well-suited for approximately solving non-counting variants of MANIPULATION and POSSIBLE-WINNER.

There remain several concrete questions for future research. Do there exist natural voting rules for which MANIPULATION is in P whereas #MANIPULATION becomes #P-hard? Reasonable candidates for such a voting rule might be Bucklin (in general) or Copeland (for the case of a single manipulator). It also is interesting to investigate the computational complexity of Borda-#MANIPULATION for which the NP-hardness of the decision variant is still open. Regarding #MANIPULATION for k-valued scoring rules, we provided a polynomial-time algorithm for constant k. Is it possible to obtain a fixed-parameter algorithm with respect to k, that is, is there an algorithm with running time $f(k)\cdot$ poly for a computable function f? We end with a more conceptual question concerning the computation of the winning probability for a candidate under other probability distributions. Specifically, how can we obtain reasonable distributions?

References

Bachrach, Y.; Markakis, E.; Procaccia, A. D.; Rosenschein, J. S.; and Saberi, A. 2008. Approximating power indices. In *Proc. of AAMAS '08*, 943–950.

Bartholdi, III, J., and Orlin, J. 1991. Single transferable vote resists strategic voting. *Social Choice and Welfare* 8(4):341–354.

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

Betzler, N., and Dorn, B. 2009. Towards a dichotomy of finding possible winners in elections based on scoring rules. In *Proc. of MFCS '09*, 124–136.

Betzler, N.; Hemmann, S.; and Niedermeier, R. 2009. A multivariate complexity analysis of determining possible winners given incomplete votes. In *Proc. of IJCAI '09*.

Brightwell, G., and Winkler, P. 1991. Counting linear extensions. *Order* 8(3):225–242.

Conitzer, V.; Sandholm, T.; and Lang, J. 2007. When are elections with few candidates hard to manipulate? *J. ACM* 54(3):1–33.

Deng, X., and Papadimitriou, C. H. 1994. On the complexity of cooperative solution concepts. *Math. Oper. Res.* 19(2):257–266.

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

Faliszewski, P.; Hemaspaandra, E.; and Schnoor, H. 2008. Copeland voting: Ties matter. In *Proc. of 7th AAMAS 08*, 983–990.

Friedgut, E.; Kalai, G.; and Nisan, N. 2008. Elections can be manipulated often. In *Proc. of FOCS'08*, 243–249.

Hazon, N.; Aumann, Y.; Kraus, S.; and Wooldridge, M. 2008. Evaluation of election outcomes under uncertainty. In *Proc. 7th AAMAS '08*, 959–966.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association* 58(301):13–30.

Huber, M. 2006. Fast perfect sampling from linear extensions. *Discrete Mathematics* 306(4):420–428.

Konczak, K., and Lang, J. 2005. Voting procedures with incomplete preferences. In *Multidisciplinary Workshop on Advances in Preference Handling*.

Lang, J.; Pini, M.; Rossi, F.; Venable, K.; and Walsh, T. 2007. Winner determination in sequential majority voting. In *Proc. of IJCAI '07*, 1372–1377.

Valiant, L. 1979. The complexity of computing the permanent. *Theor. Comp. Sci.* 8(2):189–201.

Walsh, T. 2007. Uncertainty in preference elicitation and aggregation. In *Proc. of AAAI '07*, 3–8.

Xia, L., and Conitzer, V. 2008a. Determining possible and necessary winners under common voting rules given partial orders. In *Proc. of AAAI '08*, 196–201.

Xia, L., and Conitzer, V. 2008b. Generalized scoring rules and the frequency of coalitional manipulability. In *Proc. of EC '08*, 109–118.

Xia, L.; Zuckerman, M.; Procaccia, A.; Conitzer, V.; and Rosenschein, J. 2009. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proc. 21st IJCAI '09*.