Lot-based Voting Rules

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

Lirong Xia
SEAS, Harvard University
Cambridge, MA 02138, USA
lxia@seas.harvard.edu

ABSTRACT

The Internet Engineering Task Force develops and promotes Internet standards like TCP/IP. The chair of the Task Force is chosen by an election which starts with a set of voters being selected at random from the electorate of volunteers. Selecting decision makers by lottery like this has a long and venerable history, having been used in Athenian democracy over two millennia ago, as well as for over 500 years from the 13th Century to elect the Doge of Venice. In this paper, we consider using such lotteries in multi-agent decision making. We study a family of voting rules called lot-based voting rules. Such rules have two steps: in the first step, k votes are selected by a lottery, then in the second round (the runoff), a voting rule is applied to select the winner based on these k votes. We study some normative properties of such lot-based rules. We also investigate the computational complexity of computing the winner with weighted and unweighted votes, and of computing manipulations. We show that for most lot-based voting rules winner determination and manipulation are computationally hard. Our results suggest that this general technique (using lotteries to selecting some voters randomly) may help to prevent strategic behavior of the voters from a computational point of view.

Categories and Subject Descriptors

J.4 [**Computer Applications**]: Social and Behavioral Sciences–Economics; I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems

General Terms

Algorithms, Economics, Theory

Keywords

social choice, voting, manipulation

1. INTRODUCTION

A central question in computational social choice is whether computational complexity can protect elections from manipulation. For certain voting rules it is NP-hard to compute a beneficial manipulation. Modifications like hybridizing together voting rules have also been proposed to make manipulations NP-hard to compute [8, 12]. Of course, NP-hardness results about the complexity of computing manipulations needs to be treated with caution since NP-hardness is only a worst-case notion and "hard" instances may be rare. See [13, 15] for some recent surveys. Of course, if it is already computationally hard for a manipulator to compute the winner, then intuitively it is likely to be computationally hard also to find a beneficial manipulation. Indeed, there are voting rules like Kemeny's, Dodgson's and Slater's where just computing the winner is NP-hard [4, 1, 2, 9].

In this paper, we show that a simple form of non-determinism also offers a potential escape from manipulation. We study a simple "tweak" to a voting rule that uses a lottery to select a subset of the voters before applying the original voting rule. This tweak is inspired by the election procedure for the Chair of the Internet Engineering Task Force (IETF). The IETF develops and promotes Internet standards like TCP/IP. Every two years, ten people are randomly selected from among the 100 or so eligible volunteers to be the voting members of the nominations committee. This committee then nominates the new Chair using some (unspecified) voting rule.

Similar elections have been used in several other settings. For instance, a complex election procedure involving multiple lotteries was used to select the Doge of Venice for over 500 years. Lotteries were also used in the election of the Archbishop of Novgorod, one of the oldest offices in the Russian Orthodox Church. More recently, a lottery was used in 2004 to select a Citizens' Assembly on Electoral Reform which then voted on changing British Columbia's provincial voting system. In 2006, Ontario ran a similar lottery to decide on its provincial voting system. Several Spanish savings banks use a lottery amongst their account holder to select an assembly that then elects representatives for the account holders. Finally, elections involving lotteries have also been proposed as a means to reform both the British House of Lords, and the US House of Representatives.

It has been suggested that lotteries are more democratic than elections since they are inherently egalitarian and arguably less corruptible [11]. On the other hand, lotteries are not without their issues. For instance, the electorate needs to be confident in the randomness of the selection process. To this end, the IETF has defined a robust, general, public method for making random selections (RFC 3797 - Publicly Verifiable Nominations Committee Random Selection).

Our contributions. We study a family of voting rules, called *lot-based rules*, motivated by the election procedure used to select the Chair of the IETF. Lot-based rules are composed of two steps: in the first step, k votes are selected by a lottery, then in the second step (the runoff), a voting rule (called the *runoff rule*) is applied to select the winner based on these k votes. We study some normative properties of lot-based rules. We investigate the computational complexity of computing the winner of lot-based rules with weighted and unweighted votes, respectively, and of computing a manipulation. We show that for most lot-based voting rules win-

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), June, 4–8, 2012, Valencia, Spain.

ner determination and manipulation are computationally hard. Our results suggest that this general technique (using lotteries to selecting some voters randomly) prevents strategic behavior of the voters from a computational point of view.

Related work. Lot-based rules are a type of randomized voting rule. Gibbard [17] proved that when there are at least 3 candidates, if a randomized voting rule satisfies *Pareto optimality* and a probabilistic version of strategy-proofness, then it must be a probability mixture of dictatorships (called *random dictatorships*). We note that any random dictatorship is a lot-based rule, where $k = 1$, and the runoff rule selects the top-ranked candidate as the winner when there is a single vote.

Conitzer and Sandholm [8] and Elkind and Lipmaa [12] studied another type of hybrid voting systems where manipulations are hard to compute. Their systems have two steps: in the first step, a (possibly randomized) voting rule is used to rule out some candidates, and in the second step another voting rule (not necessarily the same as the one used in the first step) is used to select the winner from the remaining candidates. We note that in the first step of their systems, some *candidates* are eliminated, while in the first step of our lot-based rules, some *voters* are eliminated. In that sense, lot-based rules can also be seen as a universal tweak that adds a pre-round that randomly eliminates some voters, to make voting rules hard to manipulate. It would therefore be interesting to consider even more complex voting systems which do both.

Technically, the winner determination problem studied in this paper is also closely related to the problem of constructive control by adding/deleting votes (CCAV/CCDV) [5]. In the winner determination problem studied in this paper, we are given a lot-based rule, a profile P, a candidate c, and a number $0 \leq p \leq 1$. We are asked to decide whether the probability for c to win is larger than p. In CCAV we are also given a set of new votes P', and we are asked whether c can be made win by adding no more than T votes in P'. In CCDV we are asked whether c can be made win by deleting no more than T votes in P. Suppose for some voting rule, it is NP-hard to compute CCDV where exactly T votes are deleted. Then, winner determination for the corresponding lot-based rule is also NP-hard, where $|P| - T$ votes are randomly selected in the runoff, and we are asked whether the probability for c to win is strictly larger than 0. On the other hand, an algorithm for the counting variant of CCAV (that computes how many ways c can be made win by adding exactly T votes in P') can be used to compute the probability for c to win in P' for the corresponding lot-based rule, where T votes are randomly selected in the runoff.

Finally, we note that for lot-based rules, it is easy for the chair to compute the winner provided computing the winner for the runoff rule is easy. This is different from a voting rule like Kemeny's where computing the winner of a given profile is hard. Whilst computing the winner for lot-based rules is computationally easy, it is nevertheless computationally hard to manipulate such rules. In order for a manipulator to compute the benefits of a false vote, she needs to compute the probability for a given candidate to win, which we will show to be computationally intractable.

2. PRELIMINARIES

Let $\mathcal{C} = \{c_1, \ldots, c_m\}$ be the set of *candidates* (or *alternatives*). A linear order $\succ$ on $\mathcal{C}$ is a transitive, antisymmetric, and total relation on $\mathcal{C}$. The set of all linear orders on $\mathcal{C}$ is denoted by $L(\mathcal{C})$. An n-voter profile P on $\mathcal{C}$ consists of n linear orders on $\mathcal{C}$. That is, $P = (V_1, \ldots, V_n)$, where for every $j \leq n$, $V_j \in L(\mathcal{C})$. The set of all n-profiles is denoted by $\mathcal{F}_n$. We let m denote the number of candidates. A (deterministic) *voting rule* r is a function that maps any profile on $\mathcal{C}$ to a unique winning candidate, that is, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \rightarrow \mathcal{C}$. A *randomized voting rule* is a function that maps any profile on $\mathcal{C}$ to a distribution over $\mathcal{C}$, that is, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \rightarrow \Omega(\mathcal{C})$, where $\Omega(\mathcal{C})$ denotes the set of all probability distributions over $\mathcal{C}$. For any randomized scoring rule r, any profile P, and any alternative c, $(r(P))(c)$ is the probability for c to win.

For any profile P and any pair of candidates $\{c, d\}$, let $D_P(c, d)$ denote the number of times that $c \succ d$ in P minus the number of times that $d \succ c$ in P. The *weighted majority graph (WMG)* is a directed graph whose vertices are the candidates, and there is an edge between every pair of vertices, where the weight on $c \rightarrow d$ is $D_P(c, d)$. We note that in the WMG of any profile, all weights on the edges have the same parity (and whether this is odd or even depends on the number of votes), and $D_P(c, d) = -D_P(d, c)$.

The following are some common voting rules. If not mentioned specifically, ties are broken in the fixed order $c_1 \succ c_2 \succ \cdots \succ c_m$.

• *Positional scoring rules*: Given a *scoring vector* of m integers, $\vec{s}_m = (\vec{s}_m(1), \ldots, \vec{s}_m(m))$, for any vote $V \in L(\mathcal{C})$ and any $c \in \mathcal{C}$, we let $\vec{s}_m(V, c) = \vec{s}_m(j)$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, we let $\vec{s}_m(P, c) = \sum_{j=1}^{n} \vec{s}_m(V_j, c)$. The rule selects $c \in \mathcal{C}$ so that $\vec{s}_m(P, c)$ is maximized. We assume scores are decreasing. Examples of positional scoring rules are *plurality*, for which the scoring vector is $(1, 0, \ldots, 0)$, *majority* which is the special case of plurality in which $m = 2$, and *Borda*, for which the scoring vector is $(m - 1, m - 2, \ldots, 0)$.

• *STV*: This rule requires up to $m - 1$ rounds. In each round, the candidate with the least number of voters ranking them first is eliminated until one of the remaining candidates has a majority.

• *Approval*: Each voter submits a set of candidates (that is, the candidates that are "approved" by the voter). The winner is the candidate approved by the largest number of voters. Every voter can approve any number of candidates.

• *Voting trees*: A voting tree is a binary tree with m leaves, where each leaf is labelled with a candidate. Each internal node is labelled with the child candidate that wins a pairwise election. The candidate labelling the root of the tree (i.e. wins all its rounds) is the winner. The rule that uses a balanced voting tree is the *Cup* rule.

• *Copeland*: We compare every pair of candidates. Each candidate gets 1 point every time it is preferred by more than half the voters. The candidate with the highest total score wins.

• *Maximin*: A candidate's score in a pairwise election is the number of voters that prefer it over the opponent. A candidate's overall score is the lowest score it gets in any pairwise election. The candidate with the highest overall score wins.

• *Ranked pairs*: We consider every pair of candidates in turn, starting with the pair not yet considered in which there is the greatest majority of voters who prefer the first candidate to the second. We construct a ranking which fixes the first candidate above the second unless, by transitivity, this contradicts a previous decision. The candidate at the top of this ranking wins.

In this paper, when we define a linear order, we sometimes do not explicitly specify the rankings among a set of candidates. In such cases, the candidates are ranked according to ascending order of their subscripts. For example, let $m = 4$, $[c_2 \succ \text{Others}]$ represents the linear order $[c_2 \succ c_1 \succ c_3 \succ c_4]$. For any set of candidates C, Rev(C) represents the linear order where candidates in C are ranked according to the descending order of their subscripts. For example, $[c_2 \succ \text{Rev(Others)}]$ represents the linear order $[c_2 \succ c_4 \succ c_3 \succ c_1]$.

3. LOT-BASED VOTING RULES

We define lot-based voting rules as follows.

DEFINITION 1. *Let X denote a voting rule (deterministic or*

randomized). We define a randomized voting rule LotThen$X$ as follows. Let k be a fixed number that is no more than the number of voters. The winner is selected in two steps: in the first step, k voters are selected uniformly at random, then, in the second step, the winner is chosen by applying the voting rule X to the votes of the k voters selected in the first step.

For instance, LotThenApproval is an instance of this rule in which the set of voters is first reduced by a lottery, and then a winner is chosen by approval voting. All lot-based rules are parameterized by k, which is the number of randomly selected runoff voters. We emphasize that in the first step of lot-based rules, some *voters* are eliminated, while in the first step of voting systems studied by Conitzer and Sandholm [8] and Elkind and Lipmaa [12], some *candidates* are eliminated.

We first consider the axiomatic properties possessed by lot-based voting rules. As the rules are non-deterministic, we need probabilistic versions of the usual axiomatic properties.[1]

DEFINITION 2. *A randomized voting rule r satisfies*

• anonymity*, if for any profile $P = (V_1, \ldots, V_n)$, any permutation π over $\{1, \ldots, n\}$, and any candidate c, we have $r(P)(c) = r(V_{\pi(1)}, \ldots, V_{\pi(n)})(c)$, where $r(P)(c)$ is the probability of c in the distribution $r(P)$;*

• neutrality*, if for any profile P, any permutation M over $\mathcal{C}$, and any candidates c, we have $r(P)(c) = r(M(P))(M(c))$;*

• unanimity*, if for any profile P where all voters rank c in their top positions, we have $r(P)(c) = 1$;*

• weak monotonicity*, if for any candidate c and any pair of profiles P and P', where P' is obtained from P by raising c in some votes without changing the orders of the other candidates, we have $r(P)(c) \leq r(P')(c)$;*

• strong monotonicity *(a.k.a.* Maskin monotonocity*), if for any candidate c and any pair of profiles $P = (V_1, \ldots, V_n)$ and $P' = (V'_1, \ldots, V'_n)$, such that for every $j \leq n$ and every $d \in \mathcal{C}$, $c \succ_{V_j} d \Rightarrow c \succ_{V'_j} d$, we have $r(P)(c) \leq r(P')(c)$;*

• Condorcet consistency*, if whenever there exists a candidate who beats all the other candidates in their pairwise elections, this candidate wins the election with probability* 1.

When the voting rule is deterministic (i.e. the unique winner wins with probability 1), all these properties reduce to their counterparts for deterministic rules. The next two theorems show that LotThen$X$ preserves some (but not all) of the axiomatic properties of X.

THEOREM 1. *If the voting rule X satisfies anonymity/ neutrality/ (strong or weak) monotonicity/ unanimity, then for every k, LotThen$X$ also satisfies anonymity/ neutrality/ (strong or weak) monotonicity/ unanimity.*

The proofs are quite straightforward, and are omitted due to space constraints. However, there are other properties that can be lost like, for instance, Condorcet consistency.

THEOREM 2. *LotThen$X$ may not be Condorcet consistent even when X is.*

Proof: Suppose $n = 2k + 1$, $k + 1$ voters vote in one way and the remaining k voters vote in the reverse order. The lottery may select only the votes of the minority, which means that the Condorcet winner loses. □

We note that when $n = k$, LotThen$X$ becomes exactly X. Therefore, if X does not satisfy some axiomatic property, neither does LotThen$X$.

[1]Definitions of the axiomatic properties for approval are omitted due to the space constraints.

THEOREM 3. *If LotThen$X$ satisfies an axiomatic property for every k, then X also satisfies the same axiomatic property.*

4. COMPUTING THE WINNER

In the remainder of this paper, we focus on the case where $k < n$, that is, when lot-based voting rules are non-deterministic. Hence, even if we know all the votes, we can only give a probability in general that a certain candidate wins. The EVALUATION problem we study is defined similar to the evaluation problem defined in [10].

DEFINITION 3. *In an* EVALUATION *problem, we are given a lot-based rule r, a profile P, a number p in $[0, 1]$, and a candidate c. We are asked to compute whether $(r(P))(c) > p$.*

We note that in EVALUATION, the number of runoff voters k is a part of the input. In this section, we show that lot-based voting rules may provide some resistance to strategic behavior by making it computationally hard even to evaluate who may have won. In particular, we show that there exist deterministic voting rules for which computing the winner is in P, but EVALUATION of the corresponding lot-based voting rule is NP-hard. As is common in computational social choice, we consider both weighted voted with a small number of candidates, and unweighted votes with an unbounded number of candidates. Of course, even if EVALUATION is hard, the manipulator may still be able to compute an optimal strategy in polynomial time. This issue will be discussed in Section 5.

4.1 Weighted votes

With weighted votes, computing who wins the Cup or Approval rule is polynomial. On the other hand, deciding if a candidate wins LotThenCup or LotThenApproval with greater than some probability is computationally intractable.

THEOREM 4. EVALUATION *for LotThenCup is* NP*-hard when votes are weighted and there are three or more candidates.*

Proof: We give a reduction from a special SUBSET-SUM. In such a SUBSET-SUM problem, we are given $2k'$ integers $\mathcal{S} = \{w_1, \ldots, w_{2k'}\}$ and another integer W. We are asked whether there exists $S \subset \mathcal{S}$ such that $|S| = k'$ and the integers in S sum up to W. We consider the cup rule (balanced voting tree) where ties are broken in lexicographical order. We only show the proof for three candidates; other cases can be proved similarly. For any SUBSET-SUM instance, we construct an EVALUATION for LotThenCup instance as follows.
Candidates: $\mathcal{C} = \{a, b, c\}$. The cup rule has a play b and the winner of this play c. Let $k = k' + 1$.
Profile: For each $i \leq 2k'$, we have a vote $c \succ a \succ b$ of weight w_i. In addition, we have one vote $b \succ a \succ c$ of weight W. We consider the problem of evaluating whether candidate a can win with some probability strictly greater than zero.

If the lottery does not pick $b \succ a \succ c$, then c wins for sure. If the lottery picks the vote $b \succ a \succ c$, then there are three cases to consider. In the first case, the sum of the weights of the other k' votes is strictly less than W. Then, b beats a in the first round, so a does not win. In the second case, the sum of the weights of the other k' votes is strictly more than W. Then, a beats b in the first round, but then loses to c in the second round, so a does not win. In the third case, the sum of weights of the other k' votes is exactly W. Then, a wins both rounds due to tie-breaking. Hence a wins if and only if the sum of the weights of the remaining k' votes is exactly W. Thus the probability that a wins is greater than zero if and only if there is a subset of k' integers with sum W. □

THEOREM 5. *There is a polynomial-time Turing reduction from* SUBSET-SUM *to* EVALUATION *for LotThenApproval with weighted votes and two candidates.*[2]

Proof sketch: Given any SUBSET-SUM instance $\{w_1, \ldots, w_{2k'}\}$ and W, we construct the following two types of EVALUATION for LotThenApproval instances: the profiles in both of them are the same, but the tie-breaking mechanisms are different. For each $i \leq 2k'$, there is a voter with weight w_i who approves candidate a. In addition, there is voter with weight W who approves b. Let P denote the profile and $k = k' + 1$. For any $p \in [0, 1]$, we let $A(p)$ (respectively, $B(p)$) denote the EVALUATION instance where ties are broken in favor of a (respectively, b), and we are asked whether the probability that a (respectively, b) wins for P is strictly larger than p. Then, we use binary search to search for an integer i such that $i \in [0, \binom{2k'+1}{k'} - \binom{2k'}{k'+1}]$ and the answers to both $A\left(1 - \frac{i+1}{\binom{2k'+1}{k'+1}}\right)$ and $B\left(\frac{i}{\binom{2k'+1}{k'+1}}\right)$ are "yes". If such an i can be found, then the SUBSET-SUM instance is a "yes" instance; otherwise it is a "no" instance. □

It follows that, with weighted votes and two candidates, if EVALUATION for LotThenApproval is in P then P=NP.

4.2 Unweighted votes

When the number of candidates is bounded above by a constant, computing the probability for a candidate to win for LotThen$X$ is in P for any anonymous voting rule X. Algorithm 1 uses dynamic programming, and exploits the fact that when the number of candidates is bounded above by a constant, the number of different profiles of n votes for anonymous voting rules is polynomial in n (no more than $n^{m!}$). For anonymous rules, it suffices to characterize a profile by an $m!$ dimensional vector (called a *voting situation*), where each dimension corresponds to a linear order V, and the component represents how many copies of V in the profile. For each natural number t, we let $D_t \subseteq \mathbb{N}_{\geq 0}^{m!}$ denote the set of all vectors whose components sum up to t.

Algorithm 1: Evaluation

Input: LotThenX, a profile $P \in D_n$, a candidate c.
Output: The probability $p(P)$ for c to win.

for *each* $P_t \in D_t$ **do**
  Let $p(P - P_t) = \begin{cases} 1 & \text{if } P - P_t \geq \vec{0} \text{ and } X(P_t) = c \\ 0 & \text{otherwise} \end{cases}$,
  where $(P - P_t)$ is a vector in $\mathbb{N}_{\geq 0}^{m!}$;
end
for $l = t - 1$ *to* 0 **do**
  for *each* $P_l \in D_l$ **do**
    Let $p(P - P_l) = \sum_{\vec{e} \in D_1} \frac{1}{m!} \cdot p(P - P_l - \vec{e})$;
  end
end
return $p(P)$;

Even though Algorithm 1 runs in polynomial time, its complexity is still very high in the worst case. For example, when $m = 5$, Algorithm 1 runs in time $\Theta(n^{120})$. We next show that when the number of candidates is unbounded, EVALUATION for LotThen$X$ is hard to compute for many common voting rules including Borda, Copeland, Maximin and Ranked Pairs.

[2]The proof can be easily extended to any LotThen$X$ where X is the same as the majority rule when there are only two candidates.

THEOREM 6. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenBorda is* NP*-hard.*

Proof: We prove the NP-hardness by a reduction from the EXACT 3-COVER (X3C) problem [16]. In an X3C instance, we are given a set $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ of $3q$ elements and $\mathcal{S} = \{S_1, \ldots, S_t\}$ such that for every $i \leq t$, $S_i \subseteq \mathcal{V}$ and $|S_i| = 3$. We are asked whether there exists a subset $J \subseteq \{1, \ldots, t\}$ such that $|J| = q$ and $\bigcup_{j \in J} S_j = \mathcal{V}$.

For any X3C instance $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, we construct an EVALUATION instance for LotThenBorda as follows.

Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D$, where $D = \{d_1, \ldots, d_{3q^2}\}$. Let $k = q$.

Profile: For each $j \leq t$, we let $V_j = [(\mathcal{S} \setminus S_j) \succ c \succ D \succ S_j]$. The profile is $P = (V_1, \ldots, V_t)$. We are asked to compute whether the probability for c to win is larger than zero ($p = 0$).

Suppose the EVALUATION instance has a solution. Then, there exists a sub-profile P' of P such that $|P'| = q$ and Borda$(P') = c$. Let $P' = (V_{i_1}, \ldots, V_{i_q})$. We claim that $J = \{i_1, \ldots, i_q\}$ constitutes a solution to the X3C instance. Suppose there exists a candidate $v \in \mathcal{V}$ that is not covered by any S_j where $j \in J$. Then, v is ranked above c in each vote in P', which contradicts the assumption that c is the Borda winner.

Conversely, let $J = \{i_1, \ldots, i_q\}$ be a solution to the X3C instance. Let $P' = (V_{i_1}, \ldots, V_{i_q})$. It follows that for each $v \in \mathcal{V}$, the Borda score of c minus the Borda score of v is at least $3q^2 - (3q - 3) \times q > 0$. For each $d \in D$, c is ranked above d in each vote in P'. Therefore, c is the Borda winner, which means that the EVALUATION instance is an "yes" instance. □

THEOREM 7. *With unweighted votes and an unbounded number of candidates, computing the probability for a given candidate to win under LotThenBorda is* #P*-complete.*

Proof: We prove the theorem by a reduction from the #PERFECT-MATCHING problem. Given three sets $X = \{x_1, \ldots, x_t\}$, $Y = \{y_1, \ldots, y_t\}$, and $E \subseteq X \times Y$, a *perfect matching* is a set $J \subseteq E$ such that $|J| = t$, and all elements in X and Y are covered by J. In a #PERFECT-MATCHING instance, we are asked to compute the number of perfect matchings. Given any #PERFECT-MATCHING instance X, Y, and E, we construct the following instance of computing the winning probability of a given candidate for LotThenBorda.

Candidates: $\mathcal{C} = \{c, b\} \cup X \cup Y \cup A$, where $A = \{a_1, \ldots, a_{2t}\}$. Let $k = 2t$. Suppose ties are broken in the following order: $X \succ Y \succ c \succ$ Others. We are asked to compute the probability that c wins.

Profile: For each edge $(x_i, y_j) \in E$, we first define a vote $W_{i,j} = [X \succ a_i \succ c \succ Y \succ b \succ \text{Others}]$, where elements within Y, X, A_i and B_j are ranked in ascending order of their subscripts. Then, we obtain $V_{i,j}$ from $W_{i,j}$ by exchanging the positions of the following two pairs of candidates: (1) x_i and a_i; (2) y_j and b. Let $P_V = \{V_{i,j} : \forall (x_i, y_j) \in E\}$.

For each $j \leq t$, we define a vote $U_j = [\text{Rev}(Y) \succ c \succ a_{t+j} \succ \text{Rev}(X) \succ \text{Others}]$, where Rev$(X)$ is the linear order where the candidates in X are ranked in descending order of their subscripts. Let $P_U = \{U_1, \ldots, U_t\}$. Let the profile be $P = P_V \cup P_U$.

Let P' be a sub-profile of P such that $|P'| = k = 2t$. We first claim that if Borda$(P') = c$, then $P_U \subseteq P'$. For the sake of contradiction, suppose $P_V \cap P' = \{V_{i_1,j_1}, \ldots, V_{i_l,j_l}\}$, where $l > t$. Because $|X| = t$, there exists $i \leq t$ such that i is included in the multiset $\{i_1, \ldots, i_l\}$ at least two times. For any candidate c', let $s(P, c')$ denote the Borda score of c' in P. It follows that $s(P, x_i) > s(P, c)$, which contradicts the assumption that c is the Borda winner.

Next, we prove that for any $P' = P_U \cup \{V_{i_1,j_1}, \ldots, V_{i_t,j_t}\}$ such that Borda$(P') = c$, $J = \{(x_{i_1}, y_{j_1}), \ldots, (x_{i_t}, y_{j_t})\}$ is a perfect matching. Suppose J is not a perfect matching. If $x \in X$ (respectively, $y \in Y$) is not covered by J, then we have $s(P, x) = s(P, c)$ (respectively, $s(P, y) = s(P, c)$), which means that c is not the Borda winner due to tie-breaking. This contradicts the assumption. We note that different P' correspond to different perfect matchings. Similarly, any perfect matching corresponds to a different profile P' such that $|P'| = 2t$ and Borda$(P') = c$. We note that the probability that c wins is the number of such P' divided by $\binom{t+|E|}{2t}$. Therefore, computing the probability for c to win is #P-hard. It is easy to check that computing the probability for c to win is in #P. □

It has been shown that computing constructive control by deleting votes is NP-complete [14]. Therefore, we have the following corollary.

COROLLARY 1. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenCopeland and for LotThenMaximin is* NP*-hard.*

THEOREM 8. *With unweighted votes and an unbounded number of candidates,* EVALUATION *for LotThenRankedPairs is* NP*-hard.*

Proof: We prove the NP-hardness by a reduction from a special X3C problem $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, where $t \geq 3q$ and t is even. (If $t < 3q$ then we add $3q - t$ copies of S_1 to $\mathcal{S}$, and if t is odd then we add 1 copy of S_1 to $\mathcal{S}$.) For any element v_i, we let $\Delta(v_i)$ denote the number of times c_i is covered by S_j. For any X3X instance where $t \geq 3q$ and t is even, we construct the following EVALUATION instance.

Candidates: $\mathcal{V} \cup \{c, d, e\}$. Ties are broken in the order $d \succ e \succ c \succ \mathcal{V}$. Let $k = |P| - q$.

Profile: Let P denote a profile composed of the votes shown in Table 1. We are asked to compute whether the probability for c to win is larger than zero ($p = 0$).

#	Votes
P_1: for each $j \leq t$	$d \succ e \succ S_j \succ c \succ (\mathcal{V} \setminus S_j)$
P_2: $t/2 - q + 1$	$c \succ d \succ e \succ \mathcal{V}$ Rev$(\mathcal{V}) \succ c \succ e \succ d$
P_3: $q - 1$	$c \succ d \succ e \succ \mathcal{V}$ Rev$(\mathcal{V}) \succ e \succ c \succ d$
P_4: for each $i \leq 3q$, $t/2 - \Delta(v_i)$	$d \succ v_i \succ c \succ e \succ$ Others Rev(Others) $\succ v_i \succ e \succ c \succ d$
P_5: for each $i \leq 3q$, $t/2 - \Delta(v_i)$	$d \succ c \succ e \succ v_i \succ$ Others Rev(Others) $\succ e \succ v_i \succ c \succ d$

Table 1: The profile P for LotThenRankedPairs.

In the profile, P_1 is used to encode the X3C instance; P_2 and P_3 are used to reduce the weights on the edge $d \to c$ and $e \to c$ in the weighted majority graph; P_4 is used to reduce the weights on the edges $v_i \to c$, and P_5 is used to balance the weight loss on $e \to v_i$ introduced in P_4. We make the following observation on the weighted majority graph of P.

• There is an edge $d \to e$ with weight t, an edge $e \to c$ with weight $2q - 2$. The edge between c and d has zero weight.

• For any $i \leq 3q$, there is an edge $d \to v_i$ with weight t, an edge $e \to v_i$ with weight t, and all edges between c and v_i have zero weight.

Suppose we remove q votes from P, then because $t \geq 3q$, we have that $d \to e$, $d \to v_i$ and $e \to v_i$ are fixed in the final order. We note that $\{d, e\} \succ c$ only in votes in P_1 . Therefore, if q votes can be eliminated to make c win for ranked pairs, then in all of them, we must have $\{d, e\} \succ c$, otherwise $d \to e$ and $e \to c$ will be fixed before $c \to d$ is considered. It follows that the q eliminated votes must come from P_1. Moreover, in order for c to win, the weight on each edge from $\mathcal{V}$ to c should be no more than $q - 2$, otherwise a path from d via some candidates in $\mathcal{V}$ will be fixed before $c \to d$ is considered. This means that the eliminated q votes in P_1 correspond to an exact cover of $\{v_1, \ldots, v_{3q}\}$. Therefore, EVALUATION for LotThenRankedPairs is NP-hard. □

For LotThenPlurality, we have the following corollary, which follows from a polynomial-time dynamic programming algorithm that solves the counting variant of CCAV in [22].

COROLLARY 2. *With unweighted votes and an unbounded number of candidates, computing the probability for a given candidate to win under LotThenPlurality can be solved in polynomial time.*

5. MANIPULATION

Suppose there are a group of manipulators who know the vote of the non-manipulators. We consider the computational complexity for the manipulators to compute (perhaps non-truthful) votes so that a preferred candidates wins the election. We limit our attention to unweighted votes. We consider two types of manipulation problem defined as follows.

DEFINITION 4. *In a* fixed manipulation *problem, given the votes of the non-manipulators, a favoured candidate and a probability p, we ask if the manipulator(s) can cast fixed vote(s) so that the candidate wins with probability greater than p. In an* improving manipulation *problem, we are not given any p but are given the truthful vote of the manipulator(s) and we ask if the manipulator(s) can cast fixed vote(s) so that the probability of the given candidate winning increases.*

An interesting extension, which we leave for future work, is when the manipulator(s) can decide how to vote after the lottery has taken place. This will increase the opportunities for manipulation.

It is easy to see that the improving manipulation problem for LotThenPlurality can be computed in polynomial time: the optimal strategy for the manipulator(s) is to vote for c. Therefore, it follows from Corollary 2 that fixed manipulation for LotThenPlurality is in P. By Algorithm 1, when the number of candidates and the number of manipulators are bounded above and the voting rule X is anonymous, both the fixed and the improving manipulation problems are in P.

COROLLARY 3. *When the number of candidates and the number of manipulators are bounded and votes are unweighted, fixed or improving manipulation of LotThen$X$ is in* P *for any anonymous rule X.*

When the number of candidates is not bounded, adding a lottery can increase the complexity of computing a manipulation. For example, when there is only one manipulator, computing a manipulation for Borda is in P, but it is NP-hard to compute both fixed and improving manipulations of LotThenBorda.

THEOREM 9. *When the number of candidates is unbounded and votes are unweighted, fixed manipulation of LotThenBorda is* NP*-hard for even a single manipulator.*

Proof: We prove the NP-hardness by a reduction from X3C that is similar to the reduction in the proof of Theorem 6. Given an X3C instance $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, we construct the following manipulation instance for LotThenBorda as follows.

Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D$, where $D = \{d_1, \ldots, d_{3q^2}\}$. Let $k = q$.
Profile: For each $j \leq t$, we let $V_j = [(\mathcal{S} \setminus S_j) \succ c \succ D \succ S_j]$. The profile is $P = (V_1, \ldots, V_t)$. Let $p = \binom{t-1}{q-1}/\binom{t}{q}$.

We claim that in this instance the optimal strategy for the manipulator is to vote for $[c \succ D \succ \mathcal{S}]$. We note that if the manipulator is not eliminated by the lottery, then c must win. This happens with probability $\binom{t-1}{q-1}/\binom{t}{q}$. If the manipulator is eliminated by the lottery, then following the same reasoning in the proof of Theorem 6, the probability for c to wins is strictly larger than 0 if and only if the X3C instance has a solution. This proves that fixed manipulation of LotThenBorda is NP-hard. □

THEOREM 10. *When the number of candidates is unbounded and votes are unweighted, improving manipulation of LotThenBorda is* NP*-hard for even a single manipulator.*

Proof: We prove the NP-hardness by a reduction from X3C. Given an X3C instance $\mathcal{V} = \{v_1, \ldots, v_{3q}\}$ and $\mathcal{S} = \{S_1, \ldots, S_t\}$, we construct the following manipulation instance for LotThenBorda as follows. W.l.o.g. q is even (otherwise we add three new elements $\{v_{3q+1}, v_{3q+2}, v_{3q+3}\}$ to $\mathcal{V}$ and $\{v_{3q+1}, v_{3q+2}, v_{3q+3}\}$ to $\mathcal{S}$).
Candidates: $\mathcal{C} = \{c\} \cup \mathcal{V} \cup D \cup E$, where $D = \{d_1, \ldots, d_{3q^2}\}$, $E = \{e_1, \ldots, e_{3q^2}\}$. Let $k = 3q/2$.
Profile: We will construct the profile in the way such that (1) if the manipulator vote for $[c \succ D \succ E \succ \mathcal{S}]$ and is not eliminated by the lottery, then the probability for c to win is non-zero if and only if the X3C instance has a solution, and (2) if the manipulator vote for $[\mathcal{S} \succ D \succ E \succ c]$ and is not eliminated by the lottery, then c never wins. Therefore, even though $[c \succ D \succ E \succ \mathcal{S}]$ seems better than $[\mathcal{S} \succ D \succ E \succ c]$, it is hard for the manipulator to figure out whether the former is strictly better. More precisely, for each $j \leq t$, we let

$$V_j = [(\mathcal{S} \setminus S_j) \succ D \succ c \succ E \succ S_j]$$

$$\text{and } U_j = [(\mathcal{S} \setminus S_j) \succ \text{Rev}(E) \succ c \succ \text{Rev}(D) \succ S_j]$$

Let $P_1 = \{V_1, \ldots, V_t\}$ and $P_2 = \{U_1, \ldots, U_t\}$. Let $E' = \{e_1, \ldots, e_{3q+20}\}$. Let P_3 consist of $q/2 - 1$ copies of

$$[c \succ D \succ (E \setminus E') \succ \mathcal{S} \succ E']$$

The profile is $P = P_1 \cup P_2 \cup P_3$. We are asked whether the manipulator can find a vote better than $W = [\mathcal{S} \succ D \succ E \succ c]$.

Suppose the X3C instance has a solution, w.l.o.g. denoted by $\{S_1, \ldots, S_q\}$. We prove that if the manipulator votes for $W' = [c \succ D \succ E \succ \mathcal{S}]$, then the probability for c to win is higher than in the case where she votes for W. We note that if the lottery eliminates the manipulator, the probability for c to win cancels out. Therefore, we only need to focus on the lotteries where the manipulator is selected. We note that if the manipulator votes for V, then c cannot win. If the manipulator votes for W' and the lottery chooses her and $\{V_1, \ldots, V_{q/2}, U_{q/2+1}, \ldots, U_q\} \cup P_3$, then c is the Borda winner, which means that there is an improving manipulation.

On the other hand, suppose that there is an improving manipulation. We claim that if the lottery selects the manipulator and c is the Borda winner, then (1) all votes in P_3 must be selected, and (2) the votes selected in $P_1 \cup P_2$ constitute an exact cover. If (1) or (2) is not satisfied, then in the profile after the lottery without the manipulator's vote, there exists a candidate in $\mathcal{V}$ whose Borda score is higher than the Borda score of c by at least $2|D| + 3q$, which contracts the assumption that c is the Borda winner when we also take into account the manipulator's vote. Therefore, the X3C instance has a solution. □

LotThen$X$ often inherits any computational resistance to manipulation that the voting rule X may have. For example, LotThenSTV inherits the computational complexity of STV against manipulation [3].

THEOREM 11. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenSTV.*

Proof: We prove the NP-hardness by a reduction from a special unweighted coalitional manipulation problem for STV with one manipulator (UCM_1) where c is ranked in the top position in at least one vote in P^{NM}. This problem is NP-complete [3]. For any UCM_1 instance (STV, P^{NM}, c) where c is ranked in the top position in at least one vote in P^{NM} ($|P^{NM}| = n - 1$), we construct the following manipulation problem. Let $\mathcal{C}'$ denote the set of candidates in the UCM_1 instance.

Candidates: $\mathcal{C}' \cup \{d\}$, where d is an auxiliary candidate.

Profile: Let P denote a profile of $2n - 1$ votes as follows. The first $n-1$ votes, denoted by P_1, are obtained from P^{NM} by putting d right below c. The next n votes, denoted by P_2, all rank d in the first position (other candidates are ranked arbitrarily). Let $k = |P| - 1$ and $p = 0$. For the improving manipulation problem, we let W be an arbitrary vote where d is ranked in the top position.

We note that if none of votes in P_2 is eliminated, then d is the winner, because it is already ranked in the top position by more than half the votes. Therefore, the only way c can win is if some voter in P_2 is eliminated in the first round.

Suppose the UCM_1 instance has a solution, denoted by V. Then, let V' denote the linear order over $\mathcal{C}' \cup \{d\}$ obtained from V by ranking d in the bottom position. Let P' denote the profile where a vote in P_2 is eliminated by the lot. We note that d is ranked in the top position $n - 1$ times in P'. Therefore, d is never eliminated in the first $|\mathcal{C}'| - 1$ rounds. Moreover, for any $j \leq |\mathcal{C}'| - 1$, the candidate that is eliminated in the jth round for P' is exactly the same as the candidate that is eliminated in the jth round for $P^{NM} \cup \{V\}$. In the last round, c is ranked in the top position n times, which means that $\text{STV}(P') = c$. Hence, the probability c wins is strictly larger than 0, which is the probability c wins if the manipulator votes for W.

On the other hand, suppose the manipulator can cast a vote V' to make c win with a non-zero probability. As we have shown above, c wins only when a voter in P_2 is eliminated in the first round. Let $P' = (P_{-n}, V')$. We note in STV for P', d must be eliminated in the last round, because d is ranked in the top position at least $n - 1$ times. Moreover, we recall that c is ranked in the first position in at least one vote in P^{NM}, and d is ranked right below c in the corresponding vote in P'. Therefore, d beats all candidates in $\mathcal{C}' \setminus \{c\}$ in their pairwise elections, which means that in the last round the only remaining candidates must be c and d. Let V be a linear order obtained from $V'_{n'}$ by removing d. It follows that V is a solution to the UCM_1 instance.

Therefore, it is NP-hard to compute a fixed or improving manipulation for LotThenSTV, even with a single manipulator. □

Similarly LotThenRankedPairs inherits the computational complexity of RankedPairs against manipulation [23].

THEOREM 12. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenRankedPairs.*

Proof: We prove the NP-hardness by a reduction from a special UCM_1 problem for ranked pairs, where n is odd ($|P^{NM}| = n-1$), and no weight in the majority graph is larger than $n - 5$ (if there is, then we tweak the instance by adding two pairs of votes $\{[c \succ c_1 \succ$

$\cdots \succ c_{m-1}]$, $[c_{m-1} \succ c_{m-2} \succ \cdots \succ c_1 \succ c]\}$). This problem is NP-complete [23]. For any such UCM_1 instance (RP, P^{NM}, c) where n is odd, we construct the following manipulation problem. Let $\mathcal{C}' = \{c, c_1, \ldots, c_{n-1}\}$ denote the set of candidates in the UCM_1 instance.

Candidates: $\mathcal{C}' \cup \{d, e\}$, where d and e are auxiliary candidates.

Profile: Let P denote a profile of $3n-2$ votes as follows.

• The first $n-1$ votes are obtained from P^{NM} by putting $d \succ e$ right below c.

• The remaining votes are defined in the following table.

#	Votes
n	$d \succ e \succ$ Others $\succ c$
$(n-1)/2$	$d \succ c \succ e \succ$ Rev(Others)
$(n-1)/2$	$e \succ c \succ d \succ$ Rev(Others)

Let $k = |P| - 1$ and $p = 0$. For the improving manipulation problem, let $W = [d \succ e \succ \text{Others} \succ c]$.

Let P' denote the profile obtained from P by removing one vote of $[d \succ e \succ \text{Others} \succ c]$. We make the following observation on the weighted majority of P'.

• The sub-graph for candidates in $\mathcal{C}'$ is the same as the weighted majority graph of the UCM_1 instance.

• There is an edge from d to e with weight $2(n-1)$.

• There are no edges between d and c, and e and c.

• The weights on the edges from d or e to $\mathcal{C}' \setminus \{c\}$ is $n-1$.

Therefore, in the final ranking, it is fixed that $d \succ e \succ (\mathcal{C}' \setminus \{c\})$. Suppose ties among edges are broken in the order where $e \to c$ is fixed before $c \to d$, whenever there is a tie (or alternatively, we can first obtain a set of candidates who win with some ways to break ties among edges, and then use the fixed tie-breaking $d \succ e \succ c \succ$ Others to select the winner). We note that if no vote for $[d \succ e \succ \text{Others} \succ c]$ is eliminated by the lottery, then the winner must be d, because no matter what the manipulator votes for, in the resulting majority graph either $e \to c$ with weight 1 or $d \to c$ with weight 1, and in cases where there is an edge $c \to d$, its weight must be 1 (we recall that $e \to c$ will be fixed before considering $c \to d$, due to the tie-breaking mechanism). Therefore, the only cases where c wins is when a vote of $[d \succ e \succ \text{Others} \succ c]$ is eliminated in the first round.

If the UCM_1 instance has a solution, denoted by V, then we let the manipulator vote for $[c \succ e \succ d \succ V]$. This makes c win with non-zero probability $(n/(3n-1))$, which is strictly larger than 0 (the probability c wins when the manipulator votes for W).

On the other hand, suppose the manipulator can cast a vote V' to make c win with non-zero probability. We have already argued that in the cases where c wins, a vote for $[d \succ e \succ \text{Others} \succ c]$ must be eliminated in the first round. In such cases c is the winner under ranked pairs if and only if (1) both d and e are ranked below c in V', and (2) the vote obtained from V' by removing d and e is a solution to the UCM_1 instance.

Therefore, it is NP-hard to compute a fixed or improving manipulation for LetThenRankedPairs, even with a single manipulator. □

Finally, we prove that LotThenCopeland and LotThenMaximin are both intractable to manipulate.

THEOREM 13. *With unweighted votes and an unbounded number of candidates, for even a single manipulator, fixed and improving manipulation are* NP*-hard for LotThenCopeland and LotThenMaximin.*

Proof sketch: The proof is similar to the proof of Theorem 8. For both rules, we use a profile P illustrated in Table 2 to show the reduction. In this proof, w.l.o.g. $(t-q)$ is even.

#	Votes
for each $j \le t$	$d \succ e \succ S_j \succ c \succ$ Others
$(t-q)/2$	$c \succ d \succ e \succ \mathcal{V}$ $\text{Rev}(\mathcal{V}) \succ c \succ e \succ d$
for each $i \le 3q$, $(t+2-q)/2 - \Delta(v_i)$	$d \succ v_i \succ c \succ e \succ$ Others Rev(Others) $\succ v_i \succ e \succ c \succ d$
for each $i \le 3q$, $(t+2-q)/2 - \Delta(v_i)$	$d \succ c \succ e \succ v_i \succ$ Others Rev(Others) $\succ e \succ v_i \succ c \succ d$

Table 2: The profile P for fixed or improving manipulation.

Let $k = |P| + 1 - q$. For the fixed manipulation problem, we let $p = 0$. We note that the manipulator can make c win with positive probability only if she ranks c in the top, and the votes eliminated in P corresponds to an exact cover of $\mathcal{V}$. For the improving manipulation problem, we let $W = [d \succ e \succ \mathcal{V} \succ c]$. It follows that if the manipulator's vote is W, then c wins with 0 probability. Therefore, there is an improving manipulation if and only if the fixed manipulation problem (with $p = 0$) has a solution. □

6. SAMPLING THE RUNOFF VOTERS

So far we have not discussed in details how to select the runoff voters. Of course if we only need to select k voters uniformly at random, then we can perform a naïve k-round sampling: in each round, a voter is drawn uniformly at random from the remaining voters, and is then removed from the list. However, it is more difficult to generate k voters with some non-uniform distribution. For example, different voters in a profile may have different voting power [20], and we may therefore want to generate the voters in the runoff according to this voting power. More precisely, we want to compute a probability distribution over all sets of k voters, and each time we randomly draw a set (of k voters) according to this distribution to meet some constraints. Let $\mathcal{M}$ denote the set of all $n \times k$ 0-1 matrices, in each of which the sum of each row is no more than 1 and the sum of each column is exactly 1. That is, $\mathcal{M} = \{(a_{(i,j)}) : a_{(i,j)} \in \{0,1\}, \forall i \le n, \sum_j a_{(i,j)} \le 1$ and $\forall j \le k, \sum_i a_{(i,j)} = 1\}$. Each matrix in $\mathcal{M}$ represents a set of k voters. Formally, we define the sampling problem as follows.

DEFINITION 5. *In the* LOTSAMPLING *problem , we are given a natural number n (the number of initial voters), a natural number k (the number of runoff voters), and a vector of positive rational numbers $(p_1, \ldots, p_n)$ such that for any $j \le n$, $0 \le p_j \le 1$ and $\sum_{j \le n} p_j = k$. We are asked to compute a sample of k voters such that, and for every $j \le n$, the probability that vote j is chosen is p_j.*

Using algorithms in [7]. we have the following corollary.

COROLLARY 4. LOTSAMPLING *can be solved in polynomial time.*

7. CONCLUSIONS

Our main computational complexity results are summarized in Table 3. The prevalence of computational intractable results in this table suggests that lot-based voting is worth further attention. This simple non-deterministic tweak to voting rules appears to provide considerable (worst-case) resistance to manipulation. There are many directions for future work in addition to the questions already raised. For instance, we could consider the computational complexity of EVALUATION for other lot-based voting rules. In particular, we conjecture that EVALUATION for LotThenPlurality is NP-hard. We also intend to look at the cost of computing manipulations of

Rule X	LotThen$X$		
	EVALUATION	Fixed Manipulation	Improving Manipulation
Borda	NP-hard (Theorem 6)	NP-hard (Theorem 9)	NP-hard (Theorem 10)
STV	?	NP-hard (Theorem 11)	
Ranked pairs	NP-hard (Theorem 8)	NP-hard (Theorem 12)	
Copeland	NP-hard (Corollary 1)	NP-hard (Theorem 13)	
Maximin			

Table 3: Summary of our complexity results.

lot-based voting rules in practice [24, 25, 27]. We could also consider the control of lot-based voting by the chair. In addition to the usual forms of control like addition of candidates or of voters, we have another interesting type of control where the chair influences the outcome of the lottery. Such control is closely related to control by deletion of voters. Other types of control include the chair choosing the size of the lottery and the chair choosing the voting rule used in the runoff after the lottery. Another interesting direction would be to consider the computation of possible and necessary winners for lot-based voting rules [18, 26].

Acknowledgements

Toby Walsh is supported by the Australian Department of Broadband, Communications and the Digital Economy, the ARC, and the Asian Office of Aerospace Research and Development (AOARD-104123). Lirong Xia is supported by NSF under Grant #1136996 to the Computing Research Association for the CIFellows Project. We thank Ronald Rivest, anonymous WSCAI and AAMAS reviewers for very helpful suggestions and comments.

8. REFERENCES

[1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. In *Proc. STOC 2005*, pages 684–693, 2005.

[2] Noga Alon. Ranking tournaments. *SIAM Journal of Discrete Mathematics*, 20:137–142, 2006.

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

[4] John Bartholdi, III, Craig Tovey, and Michael Trick. Voting schemes for which it can be difficult to tell who won the election. *Social Choice and Welfare*, 6:157–165, 1989.

[5] John Bartholdi, III, Craig Tovey, and Michael Trick. How hard is it to control an election? *Math. Comput. Modelling*, 16(8-9):27–40, 1992. Formal theories of politics, II.

[6] Garrett Birkhoff. Tres observaciones sobre el algebra lineal. *Univ. Nac. Tucumn Rev, Ser. A, no. 5*, pages 147–151, 1946.

[7] Min-Te Chao. A general-purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982.

[8] Vincent Conitzer and Tuomas Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proc. 18th IJCAI*, pages 781–788, 2003.

[9] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proc. 7th ACM Conference on Electronic Commerce*, pages 82–90, 2006.

[10] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. When are elections with few candidates hard to manipulate? *JACM*, 54(3):1–33, 2007.

[11] Oliver Dowlen. Sorting out sortition: A perspective on the random selection of political officers. *Political Studies*, 57:298–315, 2009.

[12] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proc. 16th ISAAC*, pages 206–215, 2005.

[13] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Using complexity to protect elections. *Commun. ACM*, 53:74–82, 2010.

[14] Piotr Faliszewski, Edith Hemaspaandra, and Lane A. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011.

[15] Piotr Faliszewski and Ariel D. Procaccia. AI's war on manipulation: Are we winning? *AI Magazine*, 31(4):53–64, 2010.

[16] Michael Garey and David Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[17] Allan Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45:665–681, 1977.

[18] Kathrin Konczak and Jérôme Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-2005 Workshop on Advances in Preference Handling*, 2005.

[19] Miranda Mowbray and Dieter Gollmann. Electing the Doge of Venice: Analysis of a 13th century protocol. In *Proc. IEEE Symposium on Computer Security Foundations*, pages 295–310, 2007.

[20] David M. Pennock and Lirong Xia. Voting power, hierarchical pivotal sets, and random dictatorships. In *Proc. IJCAI Workshop on Social Choice and Artificial Intelligence*, 2011.

[21] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proc. STOC 1978*, pages 216–226, 1978.

[22] Krzysztof Wojtas and Piotr Faliszewski. Possible winners in noisy elections. In *Proc. IJCAI Workshop on Social Choice and Artificial Intelligence*, 2011.

[23] Lirong Xia, Michael Zuckerman, Ariel D. Procaccia, Vincent Conitzer, and Jeffrey Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *Proc. 21st IJCAI*, pages 348–353, 2009.

[24] Toby Walsh. Where Are the Really Hard Manipulation Problems? The Phase Transition in Manipulating the Veto Rule. In *Proc. 21st IJCAI*, pages 324–329, 2009.

[25] Toby Walsh. An Empirical Study of the Manipulability of Single Transferable Voting. In *Proc. 19th ECAI*, pages 257–262, 2010.

[26] Toby Walsh. Uncertainty in preference elicitation and aggregation. In *Proc. 22nd AAAI*, pages 3–8, 2007.

[27] Toby Walsh. Where Are the Hard Manipulation Problems? *Journal of AI Research*, 42: 1–39, 2011.