Venetian Elections and Lot-based Voting Rules

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

Lirong Xia
Department of Computer Science
Duke University
Durham, NC 27708, USA
lxia@cs.duke.edu

Abstract

Between 1268 and 1797, the Venetian Republic used a complicated voting system that appears designed to resist manipulation. The system starts with randomly drawing voters, followed by 8 rounds of a complicated addition and elimination of voters before the approval voting rule is finally used to select the winner, the new *Doge*. In this paper, we study a family of voting rules inspired by this Venetian election system, which we call *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. Finally, we propose an efficient sampling technique for generating the k runoff voters non-uniformly.

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 for a potential manipulator to compute a beneficial manipulation.Modifications have even been proposed to tweak common voting rules to make manipulation NP-hard. [5; 10]. Such results need to be treated with caution since NP-hardness is only a worst-case notion and "hard" instances may be rare.See [11; 12] for 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 for her to find a beneficial manipulation. In fact, computing the winner is NP-hard for Kemeny's, Dodgson's and Slater's rule [3; 1; 2; 6].

Surprisingly, the idea of intentionally using complexity to prevent manipulation of a voting system goes back at least seven centuries ago. Lines argues that "*The most enduring and perhaps the most complex electoral process is quite likely that used by the Venetian oligarchy to elect their dogi*" [14]. This multi-stage voting procedure was used between 1268 and the end of the Venetian Republic in 1797. The procedure consists of 10 rounds, with all but the last round constructing an electoral college for the next round, and the last round actually electing the Doge, the highest official in Venice. This procedure appears designed to resist manipulation, or at least to offer the appearance of doing so. Wolfson argues that " *The main idea . . . seems to have been to introduce a system of election so complicated that all possibility of corruption should be eliminated*" [18]. On the other hand, Mowbray and Gollmann suggest that it is "*security theatre*", containing "*actions which do not increase security, but which are designed to make the public think that the organization carrying out the actions is taking security seriously*" [15]. Nevertheless they also remark that it "*offers some resistance to corruption of voters*".

Our contributions. Venetian elections have two interesting features in all but the last round: (1) voters are eliminated randomly, and (2) the voters in the current round vote on the voters who go forwards to the next round. In this paper, we report some preliminary results on a family of voting rules inspired by the first feature of such Venetian elections, which we call *lot-based rules*. It would be interesting nevertheless to consider the second feature. 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 the 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. Finally, we propose an efficient sampling technique for generating the k runoff voters from non-uniform distributions. Our results suggest it will be interesting to study further the computational properties of such rules.

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 essentially different from Kemeny's rule, where computing the winner of a given profile is hard. On the other hand, in order for a manipulator to compute a beneficial false vote, she needs to compute the probability for a given candidate to win, which we will show to be computationally hard. The winner evaluation/computation problem we focus on in this paper is from the perspective of a manipulator.

Related work. Lot-based rules are a type of randomized voting rules. Gibbard [13] proved that when there are at least 3 candidates, if a randomized voting rule satisfies *Pareto opti-*

mality 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 [5] and Elkind and Lipmaa [10] studied another type of hybrid voting systems where manipulations are hard to compute. Their systems are composed of 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.

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, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \to \mathcal{C}$. A *randomized voting rule* is a function that maps any profile on $\mathcal{C}$ to a distribution over $\mathcal{C}$, that is, $r : \mathcal{F}_1 \cup \mathcal{F}_2 \cup \ldots \to \Omega(\mathcal{C})$, where $\Omega(\mathcal{C})$ denotes the set of all probability distributions over $\mathcal{C}$. 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* $\vec{s}_m = (\vec{s}_m(1), \ldots, \vec{s}_m(m))$ of m integers, for any vote $V \in L(\mathcal{C})$ and any $c \in \mathcal{C}$, let $\vec{s}_m(V, c) = \vec{s}_m(j)$, where j is the rank of c in V. For any profile $P = (V_1, \ldots, V_n)$, let $\vec{s}_m(P, c) = \sum_{j=1}^{n} \vec{s}_m(V_j, c)$. The rule will select $c \in \mathcal{C}$ so that $\vec{s}_m(P, c)$ is maximized. We assume scores are integers and decreasing. Example of positional scoring rules are *majority*, for which $m = 2$ and the scoring vector is $(1, 0)$; *Borda*, for which the scoring vector is $(m-1, m-2, \ldots, 0)$.

• *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 associated with an candidate. In each round, there is a pairwise election between an candidate c_i and its sibling c_j: if the majority of voters prefer c_i to c_j, then c_j is eliminated, and c_i is associated with the parent of these two nodes. The candidate that is associated with the root of the tree (i.e. wins all its rounds) is the winner. The rule that uses a balanced voting tree is also known as *cup*.

3 Electing the Doge

The electorate (which consisted of around the 1000 or so male members of the Maggior Consiglio aged 30 or over) were first reduced by a lottery to an electoral college of 30 voters. This college was then reduced again by a lottery to 9 voters.[1] These 9 then elected a college of 40 voters chosen from any of the electorate, all of whom had to receive 7 out of 9 approval votes.[2] These 40 were then reduced by a lottery to an electoral college of 12 voters. These 12 then elected a college of 25 voters, all of whom had to receive 9 out of 12 approval votes. These 25 were then reduced by a lottery to an electoral college of 9 voters. These 9 then elected a college of 45 voters, all of whom had to receive 9 out of 12 approval votes. These 45 were then reduced by a lottery to an electoral college of 11 voters. These 11 then elected a college of 41 voters, all of whom had to receive 9 out of 11 approval votes. In the tenth and final round, the electoral college of 41 voters elected the Doge, who was required to receive 25 or more approval votes from the 41 voters.

This itself is still a simplified description of the process. For example, the process of enlarging the electoral college by vote was itself complicated. Consider the third round of the election where the electoral college is enlarged from 9 members to 40. The first 4 of the 9 college members selected by the lottery in the second round each nominated 5 people (who each had to receive 7 out of 9 approval votes) whilst the last 5 of the 9 college members selected by the lottery in the second round each nominated 4 people (who also each had to receive 7 out of 9 approval votes). This gives a total of 40 nominated members in the electoral college for the fourth round. Similarly, in the fifth, seventh and ninth rounds when the electoral college was enlarged, each member of the college nominated in turn a small number of new members. As a second example of the additional complexity, only one person from each family was allowed to be selected by a lottery. All relatives of a person selected by a lottery were removed from the rest of that round. As a third example, none of the members of the electoral colleges of size 9, 11 or 12 were allowed to be members of the final electoral college of size 41. As a fourth example of the additional complexity, the vote in the final round was not a simple approval vote. In addition to their approval votes, each member of this final electoral college also nominated one candidate. These nominated candidates were considered in a random order, and the first candidate who was secured 25 approval votes was elected the Doge.

The voting procedure also changed in several ways over the centuries. For example, the penultimate round originally had an electoral college of 40 voters. However, after a tied vote in 1229, this was increased to 41 to reduce the chance of a tie. As a second example, as explained earlier, the final vote was originally sequential. However, at some later point, voting moved to simultaneous voting.

[1] It has been suggested that two rounds (instead of one) are used to reduce the electoral college of 9 votes largely for procedural ease. That is, it was difficult to reduce the size of election college to 9 with a single lottery.

[2] It was not specified what happens if none of the voters receive 7 approval votes.

4 Lot-based voting rules

Elections involving lotteries are not restricted to Venice. Many other Italian cities have used such elections as well. Lotteries were also used in the election of the Archbishop of Novgorod, one of the oldest offices in the Russian Orthodox Church. Indeed the use of lotteries in elections can be traced back to at least before the birth of Christ with elections in the city-state of Athens. One of the arguments advanced for using lotteries is their fairness and resistance to manipulation [8].

We consider therefore a family of lot-based voting rules that are guaranteed always to elect a winner. These rules are closely related to the procedure used to elect the Doge.

Definition 1. *Let X denote a voting rule (deterministic or randomized). We define a randomized voting rule LotThenX as follows. Let k be a fixed number that is smaller 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 the voting rule X from 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. Lot-based rules are in practical use. For example, the Chair of the Internet Engineering Task Force is selected by a randomly chosen nominating committee of 10 persons who vote (using an unspecified rule) for the new Chair.

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 [5] and Elkind and Lipmaa [10], 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 defined as follows.[3]

Definition 2. A randomized voting rule r satisfies

• *anonymity*, if for any profile $P = (V_1, \dots, V_n)$, any permutation π over $\{1, \dots, n\}$, and any candidate c, we have $r(P)(c) = r(V_{\pi(1)}, \dots, V_{\pi(n)})(c)$, 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*, if for any candidate c and any pair of profiles $P = (V_1, \dots, V_n)$ and $P' = (V'_1, \dots, V'_n)$, 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.

[3]The definitions for the axiomatic properties for approval are omitted due to the space constraints.

When the voting rule is deterministic (i.e. the unique winner wins with probability 1), all these axioms reduce to their counterparts for deterministic rules. The next theorem shows that LotThenX preserves some of these axioms from X.

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

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

Theorem 2. *LotThenX may not be Condorcet consistent even when X is.*

Proof: Suppose $n = 2k+1$, $k+1$ voters vote in one order 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 does not win with probability 1. □

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

Theorem 3. *If LotThenX satisfies an axiomatic property for every k, then X also satisfies the same axiomatic property.*

5 Computing the winner

Lot-based voting rules are non-deterministic. Hence, even if we know all the votes, we can only give a probability that a certain candidate wins. Following [7], given a probability p in $[0, 1]$, we define EVALUATION as the decision problem of deciding whether a given candidate can win with a probability strictly larger than p. In this section, we show that lot-based voting rules 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 her optimal strategy in polynomial time. This issue will be discussed in Section 6.

5.1 Weighted votes

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 problem. In such a SUBSET-SUM problem, we are given $2k'$ integers $\mathcal{S} = \{w_1, \dots, w_{2k'}\}$ 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 any $b \succ a \succ c$, 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 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*[4] .

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

5.2 Unweighted votes

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. 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 $k = q$ and $p = 0$.

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

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 $P = (V_1, \ldots, V_t)$.

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 all 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 $\text{Borda}(P') = c$. 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. □

6 Manipulation

Suppose there are a group of k manipulators, who know the vote of the non-manipulators. There are at least three different dimensions to an analysis of manipulation in lot-based voting rules. The first two dimensions are standard, and the third dimension is specific for the lot-based rules.

The first dimension: weighted or unweighted votes.

The second dimension: constructive or destructive. Given a positive number p, in constructive manipulations, the manipulators seek to cast votes to make a given candidate win with probability at least p; in destructive manipulations, the manipulators seek to cast votes to make a given candidate lose with probability at least p.

The third dimension: fixed or adaptive. The manipulation is fixed, if all agents must declare a fixed preference ordering in advance of the lottery. In particular, the manipulators are not allowed to change their votes after lots are drawn. The manipulation is adaptive, if the manipulators observe the drawing of lotteries and can change their votes in light of which agents remain in the electoral college after the lottery. An adaptive manipulation is then described in terms of a strategy.

In this paper, we consider the manipulation problem where we are also given a positive number $p \leq 1$ and we are asked whether the manipulators can make a favored candidate c win with probability strictly larger than p. We stress that we are not asked how to compute the optimal strategy for the manipulators. These manipulation problems are closely related. For example, if fixed manipulation is possible for some p then adaptive manipulation is also possible for at least the same p. The same strategic vote will ensure this. However, the problems have different computational complexities. Whilst fixed manipulation is in NP, it is not immediately obvious that adaptive manipulation is even in PSPACE. In general, adaptive manipulations seem to be harder to compute than fixed manipulations. However, surprisingly, there are (somewhat artificial) lot-based voting rules where adaptive manipulation is easy to compute but fixed manipulation is intractable.

Theorem 8. *When the number of candidates is unbounded, there exists an instance of LotThenX for which unweighted adaptive constructive manipulation is polynomial for any size of coalition, but unweighted fixed constructive manipulation is* NP*-hard for even a single manipulator.*

Proof sketch: We will use the 1-in-3-HittingSet (denoted by 1-IN-3HS) problem in this proof, which is known to be NP-complete [17]. In a 1-IN-3HS instance, we are given a set of Boolean variables $\mathcal{V} = \{\mathbf{x}_1, \dots, \mathbf{x}_q\}$, and a set of t positive clauses $\mathcal{S} = \{S_1, \dots, S_t\}$, where for each $j \leq t$, $S_j \subseteq \mathcal{V}$ and $|S_j| \leq 3$, that is, S_j contains at most 3 positive literals. We are asked whether there exists a valuation for $\mathcal{V}$ such that for every $j \leq t$, exactly one of the positive literals in S_j is satisfied.

We consider a lottery that picks two votes at random and the following rule X on two votes: the rule always selects either c_1 or c_2. If one vote has c_1 on top, the other vote has c_2 on top, and the vote with c_1 on top encodes a 1-IN-3HS satisfying assignment to the positive clause encoded by the vote with c_2 on top, then the winner is c_2. In any other situations, c_1 wins. To encode a truth assignment within a vote, we let $m = 2l+2$, and for each $i \leq l$, c_{2i+1} is ranked above c_{2i+2} if and only if X_i is true; to encode a positive clause within a vote, for each $i \leq l$, c_{2i+1} is ranked above c_{2i+2} if and only if X_i is in the clause.

Adaptive manipulation is now polynomial to compute since, for any lot containing a manipulator, we can easily compute whether the manipulators can make c_2 (c_1) win, and for any lot not containing a manipulator, we can also easily compute the winner. Thus, we can easily compute the maximum probability with which c_2 (c_1) can be made to win (and a manipulation that will achieve any probability up to this maximum). On the other hand, consider a fixed manipulation problem with a single manipulator in which the votes of the non-manipulators rank c_2 on top, and their votes encode the $t = n - 1$ positive clauses in a 1-IN-3HS instance. The only chance that c_2 can be made to win is when the manipulator is drawn in the random lot and votes with a "satisfying assignment". The probability that the random lot contains the manipulator is $\frac{2}{n}$. Hence, computing a fixed constructive manipulation for c_2 and $p = \frac{2}{n} - \frac{1}{\binom{n}{2}}$ is equivalent to finding a 1-IN-3HS satisfying assignment to all t clauses. □

7 Sampling the runoff voters non-uniformly

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 not clear how to generate k voters with some non-uniform distribution. For example, different voters in a profile may have different voting power [16], 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, $\mathcal{M} = \{(a_{(i,j)}) : a_{(i,j)} \in \{0,1\}, \forall i \leq n, \sum_j a_{(i,j)} \leq 1$ and $\forall j \leq k, \sum_i a_{(i,j)} = 1\}$. Each matrix in $\mathcal{M}$ represents a set of k voters. Formally, we define the sampling problem as follows.

Definition 3. *In a* 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 real numbers $(p_1, \dots, p_n)$ such that for any $j \leq n$, $0 \leq p_j \leq 1$ and $\sum_{j \leq n} p_j = k$. We are asked to compute a sampling technique that chooses k voters each time, and for every $j \leq n$, the probability that vote j is chosen is p_j.*

To solve the LOTSAMPLING problem, we first solve the following equations.

$$\forall i \leq n, \sum_j x_{(i,j)} = p_i \quad \text{and} \quad \forall j \leq k, \sum_i x_{(i,j)} = 1 \qquad (1)$$

We note that $\sum_{i \leq n} p_i = k$. For such equations, a solution where $x_{(i,j)} \geq 0$ for all $i \leq n, j \leq k$ always exists. To see this, we construct the solution by a greedy algorithm. The algorithm tries to settle the values row after row, and in each row, it tries to place as much "mass" as possible to the leftmost variable, as long as it does not violate the column constraints. Algorithm 1 does this.

Proposition 1. *Algorithm 1 returns a solution to Equations (1). Moreover, the number of non-zero entries in* $(x_{(i,j)})$ *is no more than* $n + k$.

Let $(x_{(i,j)})$ denote the outcome of Algorithm 1. Since the number of non-zero entries in $(x_{(i,j)})$ is no more than $n + k$, we can apply any polynomial-time algorithm that implements the Birkhoff-von Neumann theorem [4][5] to obtain a probability distribution over the matrices in $\mathcal{M}$ such that (1) the expectation is $(x_{(i,j)})$, and (2) the support of the distribution has no more than $n + k$ elements. That is, even though $|\mathcal{M}|$ is exponential, we only need to sample over a polynomial number of elements in $\mathcal{M}$. Therefore, we have the following theorem.

Theorem 9. *The* LOTSAMPLING *problem always has a solution that runs in polynomial-time.*

Algorithm 1: SolveEquation

```
   Input: $(p_1, \ldots, p_n)$, where $\sum_j p_j = k$ and $\forall j \leq n$,
          $0 \leq p_j \leq 1$.
   Output: A solution to Equations (1).
1  Let $x_{(i,j)} = 0$, $J = 1$.;
2  for $l = 1$ to $n$ do
3      if $\sum_{i<l} x_{(i,J)} + p_l \leq 1$ then
4          Let $x_{(l,J)} = p_l$.
5      end
6      else
7          Let $x_{(l,J)} = 1 - \sum_{i<l} x_{(i,J)}$,
           $x_{(l,J+1)} = p_l - x_{(l,J)}$, and $J = J + 1$.
8      end
9  end
10 return $(x_{i,j})$.
```

8 Future work

Lot-based voting seems worth further attention. There are many directions for future work in addition to the many questions already raised in this note. For instance, we could consider the computational complexity of EVALUATION for other lot-based voting rules. 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 chooses the outcome of the lottery. Such control is closely related to control by deletion of voters. Other types of control include choosing the size of the lottery and choosing the voting rule used after the lottery. Another interesting direction would be to consider the computation of possible and necessary winners under lot-based voting. Finally, it would be interesting to consider formal properties of the Doge rule.

[5]For example, the Dulmage-Halperin algorithm [9].

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 acknowledges a James B. Duke Fellowship and Vincent Conitzer's NSF CAREER 0953756 and IIS-0812113, and an Alfred P. Sloan fellowship for support.

References

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

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

[3] 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.

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

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

[6] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proc. EC*, pages 82–90, 2006.

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

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

[9] L. Dulmage and I. Halperin. On a theorem of Frobenius-Konig and J. von Neumann's game of hide and seek. *Trans. Roy. Soc. Canada III*, 49:23–29, 1955.

[10] Edith Elkind and Helger Lipmaa. Hybrid voting protocols and hardness of manipulation. In *Proc. ISAAC*, 2005.

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

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

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

[14] Marji Lines. Approval voting and strategy analysis: A Venetian example. *Theory and Decision*, 20:155–172, 1986.

[15] Miranda Mowbray and Dieter Gollmann. Electing the doge of venice: Analysis of a 13th century protocol. In *Proc. IEEE CSF*, pages 295–310, 2007.

[16] David M. Pennock and Lirong Xia. Voting power, hierarchical pivotal sets, and random dictatorships. To be presented at *WSCAI*, 2011.

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

[18] Arthur M. Wolfson. The ballot and other forms of voting in the italian communes. *The American Historical Review*, 5(1):1–21, 1899.