How to Rig Elections and Competitions

Noam Hazon and Paul E. Dunne and Sarit Kraus and Michael Wooldridge

Abstract

We investigate the extent to which it is possible to rig the agenda of an election or competition so as to favor a particular candidate in the presence of imperfect information about the preferences of the electorate. We assume that what is known about an electorate is the *probability* that any given candidate will beat another. As well as presenting some analytical results relating to the complexity of finding and verifying agenda, we develop heuristics for agenda rigging, and investigate the performance of these heuristics for both randomly generated data and real-world data from tennis and basketball competitions.

1 Introduction

The impossibility theorems of Arrow and Gibbard-Satterthwaite are perhaps the most famous results of social choice theory [1]. At first sight, these results seem to tell us that the design of social choice mechanisms is a quixotic enterprise, since any mechanism we care to define will fail to satisfy a basic "reasonableness" axiom, or will be prone to strategic manipulation. However, in a seminal 1989 paper, Bartholdi, Tovey, and Trick argued that the fact that a voting rule in susceptible to manipulation *in theory* does not mean that is manipulable *in practice*, since it may be computationally infeasible (NP-complete or worse) to determine *how* to manipulate an election optimally [3]. Since the work of Bartholdi *et al*, many researchers have investigated the extent to which voting mechanisms can be manipulated (see, e.g., [6] for a recent collection of papers on this and related topics). Most work on the manipulation of voting procedures has considered the manipulation of elections by *voters*; specifically, the strategic misrepresentation of preferences in order to bring about a more favored outcome. However, manipulation is also possible by election officers – those responsible for organising an election, in which context it is sometimes called "control" [4].

In this paper, we consider election control by rigging the ballot agenda in order to favor a particular candidate. It is well-known that some sequential pairwise majority elections may be rigged in this way – see, e.g., [5, p.177] and [9]. In such an election, we fix an ordering of the candidates (the voting agenda), so that the first two candidates in the ordering will be in a simple majority ballot against each other, with the winner then going on to face a ballot against the third candidate, and so on, until the winner of the final ballot is the overall winner. If we know the preferences of the electorate – or more specifically, who would win in every possible ballot – then it may be possible to fix the election agenda to the benefit of a favored candidate [8].

However, the assumption that we know exactly how a voter would vote in any given ballot is very strong, and ultimately unrealistic. It ignores the possibility of strategic voting, for one thing, but more generally, the preferences of voters will *not* be public – we will have at best only *partial* information about them. In light of this, the present paper considers the extent to which it is possible to rig an election agenda (and, more generally, running orders for competitions) in the manner described above in the presence of *imperfect* information. We assume that an election officer knows the *probability* that a given candidate will beat another in a pairwise ballot. This probability may be obtained from opinion polls, in the case of governmental elections or similar; or it may be from form tables, in the case of sporting competitions. Given this, we investigate the problem of rigging an election agenda in the more realistic setting of imperfect information. We study the complexity of

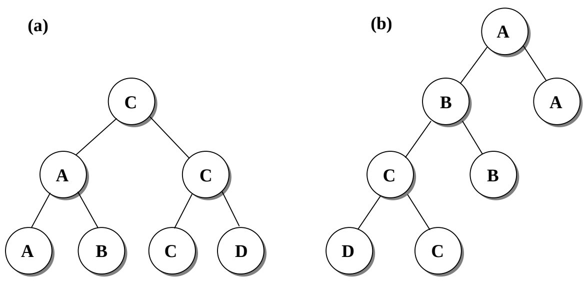


Figure 1: Organising a series of pairwise majority ballots into trees (a) and linear orders (b).

finding and verifying agendas, and present heuristics for agenda rigging. We investigate the performance of these heuristics for both randomly generated data sets and real-world data sets from tennis and basketball competitions. Finally, it is worth clarifying the motivation for this work. We are, of course, *not* advocating election manipulation, or trying to develop techniques to make it easier! If we can identify cases where election manipulation is easy in practice (even if it is hard in theory), then we can use this information to design elections so as to avoid the possibility of manipulation.

2 Preliminary Definitions

Although our problem is closely related to voting procedures, the way we choose to model the imperfect information here places it on a wider scope which includes also sports tournament. We assume we have a set $\Omega = \{\omega_1, \ldots, \omega_n\}$ of *outcomes* or *candidates* – the things the voters are trying to decide over. We are concerned with settings in which we simply want to select one outcome from Ω; more specifically, we are concerned with procedures to select such an outcome through a *sequence of pairwise simple majority ballots*. Such selection procedures are used in both political elections and sporting competitions. We often summarise voter preferences in a *majority graph*, $G \subseteq \Omega \times \Omega$, where $(\omega, \omega') \in G$ means that ω would beat ω' in a pairwise simple majority ballot. An election officer will not usually know the preferences of individual voters, but they may know the *probability* that one candidate will beat another. If all probabilities are 0 or 1 then we say the scenario is one of *perfect information*, otherwise it is one of *imperfect information*. An *imperfect information ballot matrix* M is an $|\Omega| \times |\Omega|$ matrix of probabilities, such that if $M[\omega_i, \omega_j] = p$, then in a ballot between ω_i and ω_j, candidate ω_i will win with probability p. We require that $M[\omega_i, \omega_j] + M[\omega_j, \omega_i] = 1$.

The most obvious way to organise a series of pairwise majority elections is in a *voting tree* In Figure 1(a), we see how ballots between candidates A, B, C and D may be organised into such a tree. The idea is that candidates A and B face each other in a ballot, while candidates C and D face each other in a ballot. The winner of the first ballot (A in this case) then faces the winner of the second (C), and the winner of this third ballot (C) is declared the overall winner. The voting tree in Figure 1 is said to be *fair* because it is *balanced*, and as a consequence every possible overall winner would have to win the same number of ballots. The task of the election officer may thus be seen as generating such a binary tree, with candidates Ω allocated to leaves of the tree. More formally, a ballot tree for an n candidate election is an n leaf tree, $T(V, E)$, having a distinguished root $r(T)$ and in which every node with the exception of the leaves has exactly two children. An *instantiated agenda* for a binary ballot tree is a bijective mapping $\alpha : \Omega \leftrightarrow \mathit{leaves}(T)$, which associates a candidate with each leaf node in T. Given $\langle T, \alpha, M \rangle$ the *outcome evaluation* of T with respect to α and M is a mapping $\eta : V \rightarrow [0,1]^n$ such that for any $v \in V$, $\eta(v) = \langle v_1, \ldots, v_n \rangle$ if and only $Pr[\ \omega_i$ is the winner at $v\ ] = v_i$. Thus $\eta(r(T))$ will contain at index i the probability that ω_i will be the overall winner of T.

The opportunity for manipulation arises in such a setting because the election officer can, for example, place a favored candidate against candidates it is likely to beat. If we relax the fairness constraint, then the possibilities for an election officer to manipulate the election increase. Figure 1(b) shows a rather unfair voting tree; in fact, it defines a *linear order* (C, D, B, A) for the candidates, with the first ballot taking place between C and D, the winner facing B, and so on, until the winner of the final ballot (A in this case) is the overall winner. The unfairness arises because it is possible for a candidate to win overall despite only participating in one ballot (as is the case in Figure 1). We will denote linear voting orders (i.e., permutations of Ω) by $\pi, \pi', \ldots$.

A different model of imperfect information was studied in [8], where it was assumed that for each voter we have a correct but incomplete model of their preference relation. For this incomplete information setting, they considered questions such as whether there was some completion of the incompletely known preferences and some voting tree for the candidates that would make a desired candidate a winner. Roughly, our aim in this paper is to study election manipulation in much the same way as [8], but with the probability model of imperfect information described above, rather than the incomplete profile model.

3 Voting with a Linear Order of Ballots

We start by considering *linear* voting orders. Given such an order $\pi = (\omega_1, \ldots, \omega_n)$ and outcome $\omega^* \in \Omega$, the probability of ω^* being the overall winner of π is denoted by $Pr[w(\pi) = \omega^* \mid M]$, and is given as follows. For a voting order $\pi = (\omega_{i_1}, \omega_{i_2}, \ldots, \omega_{i_n})$ with $\omega^* = \omega_{i_k}$, (i.e., the preferred winner occurs k'th in the order π),

$$P[w(\pi) = \omega^* \mid M] = \varphi \times \psi$$

where

$$\varphi = \left(\prod_{j=k+1}^{n} M[\omega_{i_k}, \omega_{i_j}] \right)$$
$$\psi = \left(\sum_{j=1}^{k-1} P[w(\omega_{i_1}\omega_{i_2}\ldots\omega_{i_{k-1}}) = \omega_{i_j} \mid M] \times M[\omega_{i_k}, \omega_{i_j}] \right)$$

That is, in order for ω^* to emerge as the winning candidate, ω^* must defeat every candidate put forward *later* in the voting order, *and* succeed against the eventual winner of the voting order formed by the *earlier* candidates. We will show in the next section that the probability of a candidate being a winner in any given voting tree can be computed in time $O(n^3)$; for linear orders, we can improve this to $O(n^2)$; we omit the proof.

What else can we say about voting with linear orders? First, we can make precise, and prove correct, the intuition that there is no benefit to going early; a candidate can only benefit by going late in a voting order. While this seems intuitive, the proof of the following lemma, which we omit, is surprisingly involved.

Lemma 1 *Given $\langle M, \Omega \rangle$ let ω_i, ω_j be distinct members of Ω. For every voting order $\pi_1\pi_2$ of $\Omega \setminus \{\omega_i, \omega_j\}$, it holds that $Pr[w(\pi_1\omega_j\omega_i\pi_2) = \omega_i] \geq Pr[w(\pi_1\omega_i\omega_j\pi_2) = \omega_i]$.*

The immediate corollary is as follows.

Corollary 1 *For any candidate $\omega^* \in \Omega$, if there is a voting order, π such that $Pr[w(\pi) = \omega^*] \geq p]$, then there is a voting order π' such that $Pr[w(\pi') = \omega^*] \geq p]$ and in which ω^* is the final candidate to run.*

Proof: Given π with $Pr[w(\pi) = \omega] \geq p$, apply Lemma 1 repeatedly to move ω later in the voting order until it is the final candidate. ▫

Note that the equivalent result *does not* hold for trees. It is possible to construct an example in which a candidate wins with probability 1 in a fair tree, but loses with probability 1 if we convert the tree to a linear order with the same candidates order and place the candidate last. Even though it faces fewer ballots the candidate has a strictly smaller probability of winning.

It is also natural to ask what the probability is that a particular candidate is a *possible winner*, that is, if there is *some* permutation of Ω which would make ω^* the winner with non-zero probability. The decision problem POSS-WIN takes as its instance a triple $\langle M, \Omega, \omega^* \rangle$, which is accepted iff ω^* is a possible winner of $\langle M, \Omega \rangle$.

Theorem 1 POSS-WIN *is polynomial time decidable.*

Proof: (Outline) Convert all the non-zero probabilities to one, and apply the techniques of [8] to check whether ω^* is a possible winner. ▫

The general problem of determining whether there exists any agenda which gives a named candidate at least a certain probability of winning is hard to classify, and so we will analyse a restricted version of the problem, as follows. When we think informally about rigging an agenda, we tend not to think just in terms of the agenda, but also in terms of the *specific outcomes* that we want the agenda to lead to. So, we might think in terms of "if I put A up against B, then B wins and goes up against C, and C wins..." and so on. Here, we have not just the agenda (ABC) but also the outcomes of the ballots (B wins the first; C the second; ...). Here, we call these structures – which include the agenda for the ballots together with the intended outcomes – a *run*. A run has the form $r : \omega_1, \omega_2 \xrightarrow{\omega_2} \omega_3 \xrightarrow{\omega_3} \cdots \xrightarrow{\omega_{k-1}} \omega^* \xrightarrow{\omega^*}$ where ω_1 and ω_2 are the candidates up against each other in the first ballot, ω_2 is the intended winner of this ballot, and so on, until the final ballot is between ω_{k-1} and ω^*, in which we intend the winner – and hence overall winner – to be ω^*. Computing the probability that this run will result in our desired candidate ω^* winning is simple – it is the value: $Pr[w(\omega_1, \omega_2) = \omega_2] \times Pr[w(\omega_2, \omega_3) = \omega_3] \times \cdots \times Pr[w(\omega^{k-1}, \omega^*) = \omega^*]$. We denote this value for a run r by $Pr[r \mid M]$. So, in the *weak imperfect information rigged agenda* (WIIRA) problem, we are given a set of candidates Ω, an imperfect information ballot matrix M, a favored candidate $\omega^* \in \Omega$, and a probability p. We are asked whether there exists a run r, in which the overall winner is ω^*, such that $Pr[r \mid M] \geq p$.

Theorem 2 WIIRA *is* NP*-complete.*

Proof: (Summary) A standard "guess and check" algorithm gives membership in NP. For hardness, we reduce the k-HCA problem on tournaments [2, p.46]: we are given a tournament $G = (V, E)$ (i.e., a complete digraph such that $(\omega, \omega') \in E$ iff $(\omega', \omega) \notin E$) and a subset $E' \subseteq E$, and we are asked whether G contains a Hamiltonian cycle containing all edges E'. We create an instance of WIIRA as follows. The outcomes will be the vertices of G together with a new vertex, $v_\perp$. Given a tournament $G = (V, E)$ and required edge set E', we create a probability matrix so that G contains a cycle with E' iff we can create an ordering of vertices satisfying the property given above. For each $(u, v) \in E'$ we set $M[u, v] = 1$. For each $(u, v) \in E \setminus E'$ we set $M[u, v] = 1 - \frac{1}{10^{|V|}}$. We then set the target probability to be $(1 - \frac{1}{10^{|V|}})^{|V|-|E'|}$. We are after a cycle, so we need to select a vertex (call it $v_\top$) to act as the source of the cycle and our new vertex, $v_\perp$, will act as the sink in the cycle; if we have an arc $(u, v_\top) \in E'$ then we define $M[u, v_\perp] = 1$, while if $(u, v_\top) \in E \setminus E'$ then we define $M[u, v_\perp] = 1 - \frac{1}{10^{|V|}}$; if for any vertex $u \in V$ we have not yet defined a value for $M[u, v_\perp]$ then define $M[u, v_\perp] = 0.5$. We then ask whether we can rig the agenda for $v^\top$ to win with probability greater than the target. ▫

4 Voting with a Tree of Ballots

We now focus on a more realistic setting, in which ballots are organised into a binary tree. There are several obvious questions to ask here. For example, the most obvious decision problem with respect to rigging an election as follows: Given a set of outcomes Ω, an imperfect information ballot matrix M, a favored candidate $\omega^* \in \Omega$ and a probability p, does there exist a voting tree T with labeling $\alpha : \Omega \leftrightarrow \mathit{leaves}(T)$ such that ω^* wins in this setting with probability at least p? It is not obvious even that this problem is in NP, since to compute the probability that a given candidate wins overall, we must consider every possible ordering of wins arising from a given tree structure: in any given ballot, there are two outcomes, in contrast to the perfect information case. However, the following result implies the problem is in NP.

Theorem 3 *Let $T(V, E)$ be any binary ballot tree, α a labeling of the leaves of T by candidates Ω, and M a ballot matrix for Ω. The outcome evaluation of T with respect to α and M is computable in $O(n^3)$ arithmetic operations.*

Proof: Consider the following algorithm.

```
1. $\eta(x) = \mathit{unlabelled}$ for each $x \in V(T)$
2. For each leaf, $x$, of $T$, $\eta(x) = \langle x_1, \dots, x_n \rangle$ with $x_i = 1$ if $\alpha(x) = \omega_i$ and $x_i = 0$
   otherwise.
3. repeat
      a. Let $z$ be any node of $T$ with children $x$ and $y$ such that $\eta(x) \neq \mathit{unlabelled}$
         and $\eta(y) \neq \mathit{unlabelled}$. Let $\langle x_1, \dots, x_n \rangle$ denote $\eta(x)$ and, similarly, $\langle y_1, \dots, y_n \rangle$
         denote $\eta(y)$. Compute $\eta(z) = \langle z_1, \dots, z_n \rangle$ using
4. until every $v$ has $\eta(v) \neq \mathit{unlabelled}$
```

$$z_i \quad := \quad \begin{cases} x_i \sum_{j=1}^{n} y_j \; M[\omega_i, \omega_j] & \text{if} \quad x_i > 0 \\ y_i \sum_{j=1}^{n} x_j \; M[\omega_i, \omega_j] & \text{if} \quad y_i > 0 \\ 0 & \text{if} \quad x_i = y_i = 0 \end{cases}$$

This algorithm can be improved in some cases; e.g., if we know that the graph is *fair*, then we can use an optimised version to take account of this, reducing the overall time complexity to $O(n^2)$. ▫

Whether this result is matched by NP-hardness remains open, but we can obtain a related hardness result, as follows. Given a set of candidates Ω and an imperfect information ballot matrix M, a *fair tree run* r_T is a balanced tree, T, a labeling α of the leaves of T by candidates Ω, and a labeling of every other node of T by the candidate that has the higher probability to win the ballot between the node's children (If a node's children have equal probability, we select one arbitrarily). The motivation for this is just as in the linear order case; we tend to think on the agenda in terms of the *specific outcomes* that we want the agenda to lead to. The probability that a fair tree run will result in our favored candidate ω^* winning is simply the multiplication of the winning probabilities of the candidates in all the ballots induced by T. We denote this value by $Pr[r_T \mid M]$. So, in the *weak imperfect information rigged agenda for balanced trees* (WIIRA-BT) problem, we are given a set of candidates Ω, an imperfect information ballot matrix M, a favored candidate $\omega^* \in \Omega$, and a probability p. We are asked whether there exists a fair tree run r_T, in which the overall winner is ω^*, such that $Pr[r_T \mid M] \geq p$.

Theorem 4 WIIRA-BT *is* NP*-complete.*

Proof: (Summary) Membership of NP is immediate. As proved by [8], given a complete and weighted majority graph (also called a weighted tournament) G, every balanced voting tree with candidate ω^* at its root has a corresponding binomial tree with root ω^* which covers all the nodes in G – this binomial tree describes all the ballots between the candidates. We will prove our hardness result with respect to the following problem, a variation of the problem from [8, Th. 4], which is easily shown to be NP-complete: *Given a weighted tournament $G = (V, E)$, a candidate ω^* and a cost bound k, check whether there exist a balanced voting tree in which ω^* wins, such that the sum of the edges of the corresponding binomial tree is at least k.* Let denote $\min_E(\max_E)$ as the minimum (maximum) weight of the edges in E. Given an instance of the previous problem we create an instance of WIIRA-BT with a graph $G' = (V, E')$ such that for every edge $(u, v) \in E$ with a weight $w(u, v)$, we create the same edge in E' with the weight

$$2^{\frac{w(uv)-|\max_E|}{|\min_E - |\max_E||}}.$$

This conversion ensures that $0.5 \leq w(u, v) \leq 1$ in E'. For every edge $(u, v) \in E'$, we create an opposite edge (v, u) with the weight $1 - w(u, v)$. The candidate for our problem is also ω^*, and the winning probability p is

$$2^{\frac{k-(n-1)\times|\max_E|}{|\min_E - |\max_E||}}.$$

The overall number of ballots in a balanced voting tree is $n - 1$ so $k \leq (n - 1) \times |\max_E|$ and $0 \leq p \leq 1$ as required. If we denote the edges of a binomial tree for ω^* in G with a cost bound k as $x_1, x_2, \ldots, x_{n-1}$, then:

$$x_1 + x_2 + \cdots + x_{n-1} \geq k$$

$$(x_1 - |\max_E|) + \cdots + (x_{n-1} - |\max_E|) \geq k - (n - 1) \times |\max_E|$$

$$\frac{x_1-|\max_E|}{|\min_E - |\max_E||} + \cdots + \frac{x_{n-1}-|\max_E|}{|\min_E - |\max_E||} \geq \frac{k-(n-1)\times|\max_E|}{|\min_E - |\max_E||}$$

$$2^{\frac{x_1-|\max_E|}{|\min_E - |\max_E||}} \times \ldots \times 2^{\frac{x_{n-1}-|\max_E|}{|\min_E - |\max_E||}} \geq 2^{\frac{k-(n-1)\times|\max_E|}{|\min_E - |\max_E||}}$$

that is the wining probability in a fair tree run for ω^* in G' □

5 Heuristics and Experimental Evaluation

Our hardness results lead us to conjuncture that the general problem of rigging an election agenda with incomplete information is NP-Hard to compute; but we also think that a worst-case analysis is not enough. The situation is similar to that in cryptography, where a secure protocol is not one that is hard to break in the worst case, but one that can be broken only with negligible probability. Bartholdi et al., who were in many ways pioneers of the complexity-theoretic approach to understanding election manipulation [3] first voiced the concern that NP-Hardness results are not enough:

Concern: It might be that there are effective heuristics to manipulate an election even though manipulation is NP-complete.
Discussion: True. The existence of effective heuristics would weaken any practical import of our idea. It would be very interesting to find such heuristics.

This motivates us to consider heuristics for this problem, and to test their performance in different scenarios. We used a simulation program (written in C) to evaluate the heuristics. Our experiments are in two categories: first, randomly generated data, and second, public domain form data from sports competitions. For each of these settings, we evaluated the heuristics for both fair trees and linear orders. For the randomly generated data sets, we first generated random values for the probability matrix from a uniform distribution in the range $[0, 1]$, and completed the matrix to preserve probability constraints. We then ran 100 iterations, and during each iteration a winner candidate, ω^*, was randomly chosen and each heuristic was used in an attempt to generate an optimal order for this candidate. The second scenario was the same, except that the random values for the matrix were taken from a normal distribution with an average of 0.5 and a standard deviation of 0.2. (If a value of more than 1 (less than 0) was generated it was changed to 1 (0) to preserve probability constraints.) For the real-world data sets, we based our experiments on data from basketball and tennis competitions. For the basketball experiments, we took the 29 teams in the NBA, and computed the ballot matrix from public domain form data. Here, there were no iterations, but for every team we used each heuristic to generate a playing order that would give this team the best chance of winning. For the tennis experiments, we used the 13 players at the top of the ATP ranking, again computing the ballot matrix from public domain form tables.

5.1 Heuristics for Linear Voting Orders

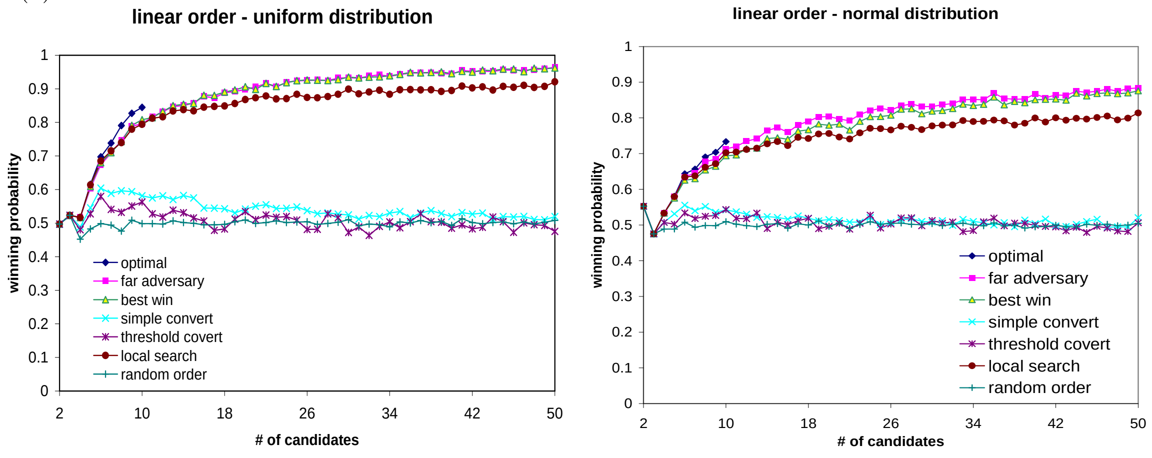
The heuristics we developed are as follows.

- *Optimal*: In those cases where it was computationally feasible to do so, we exhaustively evaluated every permutation in order to find the real optimal solution as a comparison.
- *Far adversary*: The idea here is to put candidates who are likely to beat you as far away from you as possible. Thus, the candidate who has the highest probability to beat ω^* was selected to be the first, and so on; ω^* was chosen to be the last (cf. Corollary 1).
- *Best win*: ω^* was chosen to be the last, and the candidate that ω^* has the best chance of beating was chosen to be before it, the next being the one that this candidate was most likely to beat, and so on.
- *Simple convert*: The idea here is to convert the imperfect information ballot matrix into one of perfect information, and then simply apply an algorithm which is known to work in polynomial time for such cases [8]. To create the perfect information matrix, every probability which was greater than or equal to 0.5 was converted to 1, and others were converted to 0. If no order could be found, (which is sometimes the case where the number of candidates is small) a random order was generated.
- *Threshold convert*: This is a more sophisticated attempt to make a perfect information ballot matrix. We searched for the maximum threshold above which, if we convert all the probabilities above this to 1 and below it to 0, we still have an order that enable ω^* to win (on the converted tournament matrix). We used a binary search, stopping when the difference between the low/high limit and the threshold was less than 0.005. As before, if no order could be found a random order was generated.

- *Local search*: ω^* was chosen to be last. For the other places, in every iteration a random permutation was chosen. Then $0.5 * \textit{num-of-candidates}$ random swaps were tested to find an order with maximum winning probability. ($\textit{num-of-candidates}$ iterations were done.)
- *Random order*: As a control, a random order was also generated.

The results for heuristics on linear orders are shown in Figure 2.

(a)



(b)

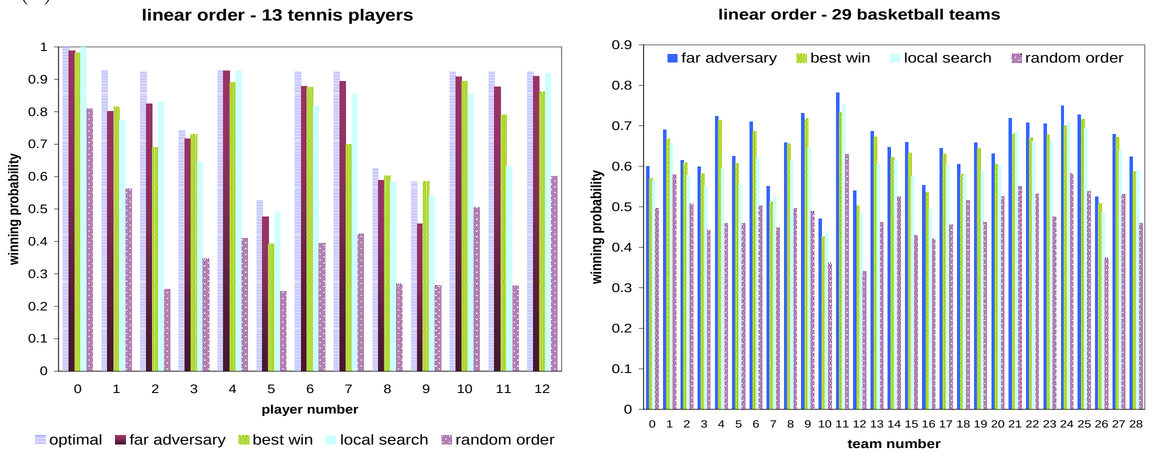


Figure 2: Performance of heuristics for linear order ballots. (a) Shows the performance of the heuristics for randomly generated ballot matrices using a uniform and normal probability distributions; and (b) Shows performance for real-world data from the domain of professional tennis and basketball.

First, note that the overall performance of the heuristics does not vary significantly between uniform and normal distributions (Figure 2(a) left and right columns, respectively). In these graphs, the x-axis is the number of candidates and the y-axis is the winning probability that was found using our heuristics. Every point in the graph represents the winning probability that was averaged over 100 iterations.

In the uniform distribution experiments, it seems that best-win and far adversary seem to perform similarly well, (marginally better than local search), while for a normal distribution they are slightly differentiated, but again perform better than local search. In the graph for linear order with normal distribution (Figure 2(a) left column), again, we find the first two heuristics – far adversary and best win – gave about the same results, while local search lies a little behind; all reach a very high winning probability, almost 0.9, and they performed better as the number of candidates increased. These heuristics also perform well when comparing them to the optimal solution – they gave a winning probability which is on average only 98% from the optimal solution. Note that the "convert" heuristics perform very poorly, both for uniform and normal distributions, and so we omitted them in subsequent experiments.

In the graph for linear order with 13 tennis players (Figure 2(b), left column), the x-axis is the player number that was chosen to be the winning candidate and the y-axis as before. Here, there was no heuristic that performed significantly better than the others in general, but when choosing from the heuristics, the best solution for each candidate performs very well when compared to the optimal solution. They gave a winning probability which is only 96% from the optimal solution on average. The winning probability is on average more than twice as high as the random order. Player number 0, (Roger Federer, currently the world's number one player), even succeeded in obtaining a winning probability of 1 from local search.

We conclude that there is no one heuristic that performs significantly better than the others for all cases. We suggest the best thing to do here is to run all the heuristics and order the candidates according the heuristic which gives the best results for this candidate because they all run quite fast. Note that local search takes much more time than other heuristics, but still has acceptable time performance.

5.2 Heuristics for Fair Voting Trees

For this voting agenda type we investigated the following heuristics.

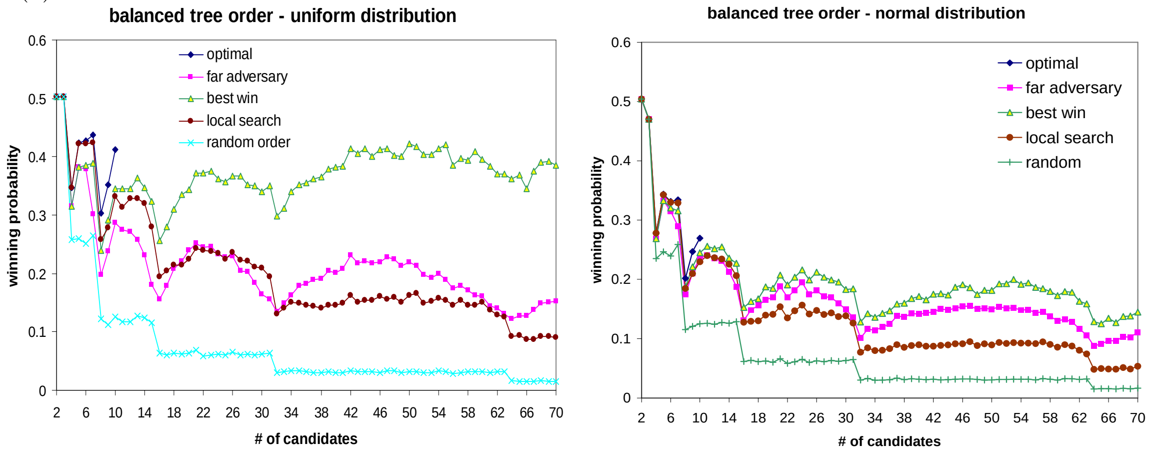
- *Optimal*, *Far adversary*, *Local search*: We organized the leaves of the binary tree as a linear order from left to right and applied these heuristics as above. (Note that when the number of candidates is not a power of 2, some of the rightmost candidates may face one less ballot than other candidates.)

- *Best win*: Because of the tree structure, we had to use a modified version of the best win heuristic, but the principle remains the same. We try to maximize the probability that ω^* will face candidates that he has a high probability of beating, and to maximize the probability that they will reach the point when they compete against him. So we first assign ω^* at the rightmost leaf of the voting tree, and for each ballot along its path to the root we assign candidates that ω^* has a high probability to beat. In this way we define for each one of them a sub-tree that we want him to be its overall winner (unless this candidate has been assigned to a leaf) so we can repeat this assignment procedure recursively.

The results of our experiments with balanced tree ballots are shown in Figure 3.

In the tree order with random uniform and normal distribution (Figure 3(a) left and right columns, respectively) the axes are as in the linear order case. Generally, the wining probability is much lower than the linear order voting protocol, which seems a direct consequence of the relative fairness of the procedure. Nevertheless, if we compare the best heuristic for each case to the random order, we get a winning probability which is on average 4.31 times higher than the random order winning probability and which is on average only 96% from

(a)



(b)

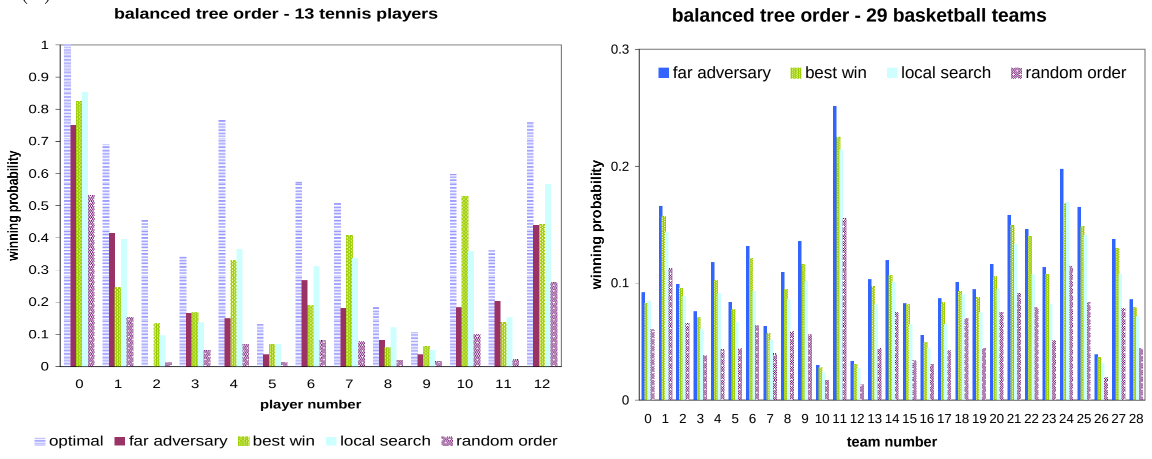


Figure 3: Performance of heuristics for balanced tree ballots. (a) Shows the performance of the heuristics for randomly generated ballot matrices using a uniform and normal probability distributions; and (b) Shows performance for real-world data from the domain of professional tennis and basketball.

the optimal solution. We also note that the graphs have a wave-like shape: the lowest points in these waves are when the number of candidates is exactly a power of 2. This is simply because when the number of candidates is a power of 2, every candidate must compete in exactly the same number of ballots. In all other cases, there are some of the candidates (including our favorite, ω^*) which face one less ballot. This effect can help ω^*, by forcing strong competitors to undergo one more ballot. It can also be seen that in contrast to the linear order with random normal distribution, the best win heuristic performs better than the others: 1.25 times better than far adversary, and 1.66 times better than local search on average.

In the tree order with 29 NBA basketball teams graph, (Figure 3(b), right column), the far adversary method performed better than others with a winning probability that is on average 1.24 times better than local search and 1.08 times better than best win. The highest winning probability was generated for team 11, the LA Lakers. It was not computationally feasible to calculate the optimal solution for 29 teams, so we ran another scenario with only the first 13 teams to check the performance of our heuristics against the optimal solution. The best heuristic in each case gave a winning probability which in on average only 99% from the optimal solution!

In the tree order with 13 tennis players graph ((Figure 3(b), left column) there was no heuristic that performed significantly better than the others, just as in the linear order case. But here, when we choose from the heuristics the best solution for each case they gave a winning probability which is on average 61% from the optimal solution. Perhaps the performance difference between this case and the basketball case emerges from the type of the distribution. The basketball scenario has a distribution which is much closer to a normal distribution than in the tennis scenario, in which the probability matrix contains many high probabilities. Another indicator for this is the performance of our best heuristic in each case against the random order. In the tennis scenario it was almost 5 times better, while in the basketball scenario it was only about 1.5 times better.

6 Related Work and Conclusions

A closely related stream of work is the problem of optimal seeding for tournaments. This problem considers how to determine an agenda for a voting tree that will result in an "interesting" sporting competition. [10] for example, assumes an imperfect information ballot matrix as we do, and uses it to produce an interesting competition agenda that still satisfies some fairness criterions. [7] investigates a 4-player scenario with an auction-like analysis; instead of knowing the probability of winning, each player exerts some effort according to its private valuation. Early work by [11] analyzes voting trees for 8 players, among 3 other tournament types. It uses an imperfect information matrix and investigates the effect of the initial agenda on the probability that the best player will win the game. Note that none of these papers analyzed the complexity of finding the optimal agenda for a specific candidate, however.

By understanding when the manipulation of elections is possible, and those cases where it is computationally easy to manipulate an election, we can engineer voting protocols to avoid such cases. We have demonstrated that, while it seems hard to control an election under incomplete information in theory, there are heuristics that perform well on this problem in practice. This research can hence usefully inform the design of future voting protocols in multi-agent systems.

References

[1] K. J. Arrow, A. K. Sen, and K. Suzumura, eds. *Handbook of Social Choice and Welfare Volume 1.* Elsevier, 2002.

[2] J. Bang-Jensen and G. Gutin. On the complexity of hamiltonian path and cycle problems in certain classes of digraphs. *Discrete App. Math.*, 95:41–60, 1999.

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

[4] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. How hard is it to control an election. *Math. and Comp. Modelling*, 16:27–40, 1992.

[5] S. J. Brams and P. C. Fishburn. Voting procedures. In K. J. Arrow, A. K. Sen, and K. Suzumura, eds, *Handbook of Social Choice and Welfare Volume 1*, Elsevier, 2002.

[6] U. Endriss and J. Lang, editors. *Proc. COMSOC-2006.* Amsterdam, 2006.

[7] Christian Groh, Benny Moldovanu, Aner Sela, and Uwe Sunde. Optimal seedlings in elimination tournaments. *Jnl of Economic Th.*, 107:140–150, 2008.

[8] J. Lang, M. S. Pini, K. B. Venable, and T. Walsh. Winner determination in sequential majority voting. In *Proc. IJCAI-07*, 2007.

[9] T. Sandholm. Distributed rational decision making. In G. Weiß, ed., *Multiagent Systems*, pages 201–258. MIT Press 1999.

[10] Allen J. Schwenk. What is the correct way to seed a knockout tournament. *American Math. Monthly*, 107(2):140–150, 2000.

[11] Donald T. Searls. On the probability of winning with different tournament procedures. *Jnl. of the American Statistical Assn*, 58(304):1064–1081, 1963.

Noam Hazon
Department of Computer Science, Bar Ilan University
Ramat Gat 52900, Israel
Email: `hazonn@cs.biu.ac.il`

Paul E. Dunne
Department of Computer Science, University of Liverpool
Liverpool L69 7ZF, United Kingdom
Email: `ped@liverpool.ac.uk`

Sarit Kraus
Department of Computer Science, Bar Ilan University
Ramat Gat 52900, Israel
Email: `sarit@cs.biu.ac.il`

Michael Wooldridge
Department of Computer Science, University of Liverpool
Liverpool L69 3BX, United Kingdom
Email: `mjw@liverpool.ac.uk`