Approval Voting: Local Search Heuristics and Approximation Algorithms for the Minimax Solution

Rob LeGrand Evangelos Markakis Aranyak Mehta

Abstract

Voting has been the most general scheme for preference aggregation in multi-agent settings involving agents of diverse preferences. Here, we study a specific type of voting protocols for multi-winner elections, namely *approval voting*, and we investigate the complexity of computing or approximating the *minimax solution* in approval voting, concentrating on elections for committees of fixed size. Given an approval voting election, where voters can vote for as many candidates as they like, a minimax outcome is a committee that minimizes the maximum Hamming distance to the voters' ballots. We first show that the problem is **NP**-hard and give a simple 3-approximation algorithm. We then introduce and evaluate various heuristics based on local search. Our heuristics have low running times (and provably polynomial) and our experimental results show that they perform very well on average, computing solutions that are very close to the optimal minimax solutions. Finally, we address the issue of manipulating minimax outcomes. We show that even though exact algorithms for the minimax solution are manipulable, we can have approximation algorithms that are non-manipulable.

1 Introduction

Voting has been a very popular method for preference aggregation in multi-agent environments. It is often the case that a set of agents with different preferences need to make a choice among a set of alternatives, where the alternatives could be various entities such as potential committee members, or joint plans of action. A standard methodology for this scenario is to have each agent express his preferences and then select an alternative (or more than one alternative in multi-winner elections) according to some voting protocol. Several decision making applications in AI have followed this approach including problems in collaborative filtering [19] and planning [9, 10].

In this work we focus on solution concepts for approval voting, which is a voting scheme for committee elections (multi-winner elections). In such a protocol, the voters are allowed to vote for, or approve of, as many candidates as they like. In the last three decades, many scientific societies and organizations have adopted approval voting, including the American Mathematical Society (AMS), the Institute of Electrical and Electronics Engineers (IEEE), the Game Theory Society (GTS) and the European Association for Logic, Language and Information.

A ballot in an approval voting protocol can be seen as a binary vector that indicates the candidates approved of by the voter. Given the ballots, the obvious question is: what should the outcome of the election be? The solution concept that has been used in almost all such elections is the minisum solution, *i.e.*, output the committee which, when seen as a 0/1-vector, minimizes the sum of the Hamming distances to the ballots. If there is no restriction on the size of the elected committee this is equivalent to a majority vote on each candidate. If there is a restriction, *e.g.*, if the elected committee should be of size exactly k, then the minisum solution consists of the k candidates with the highest number of approvals [4].

Recently, a new solution concept, the minimax solution, was proposed by Brams, Kilgour and Sanver [3]. The minimax solution chooses a committee which, when seen as a 0/1-vector, minimizes the *maximum* Hamming distance to all ballots. When there is a restriction that the size of the committee should be exactly k, then the minimax solution picks, among all committees of size k, the one that minimizes the maximum Hamming distance to the ballots.

The main motivation behind the minimax solution is to address the issues of fairness and compromise. Since minimax minimizes the disagreement with the least satisfied voter, it tends to result in outcomes that are more widely acceptable than the minisum solution. Also, majority tyranny is avoided: a majority of voters cannot guarantee a specific outcome, unlike under minisum. On the other hand, advantages of the minisum approach include simplicity, ease of computation and nonmanipulability. A further discussion on the properties and the pros and cons of the minisum and the minimax solutions can be found in [3, 4].

In this work we address computational aspects of the minimax solution, with a focus on elections for committees of fixed size. In contrast to the minisum solution, which is easy to compute in polynomial time, we show that finding a minimax solution is **NP**-hard. We therefore resort to polynomial-time heuristics and approximation algorithms.

We first exhibit a simple algorithm that achieves an approximation factor of 3. We then propose a variety of local search heuristics, some of which use the solution of our approximation algorithm as an initial point. All our heuristics run relatively fast and we evaluated the quality of their output both on randomly generated data as well as on the 2003 Game Theory Society election. Our simulations show that the heuristics perform very well, finding a solution very close to optimal on average. In fact for some heuristics the average error in the approximation can be as low as 0.05%.

Finally, in Section 5, we focus on the question of manipulating the minimax solution. We show that any algorithm that computes an optimal minimax solution is manipulable. However, the same may not be true for approximation algorithms. As an example, we show that our 3-approximation algorithm is nonmanipulable.

1.1 Related Work

The minimax solution concept that we study here was introduced by Brams, Kilgour and Sanver [3]. In subsequent work by the same authors [4, 14], a weighted version of the minimax solution is studied, which takes into account the number of voters who voted for each distinct ballot and the proximity of each ballot to the other voters' ballots. The algorithms that are proposed in [3, 4, 14] are all exponential, and this is not surprising since the problem is **NP**-hard, as we exhibit in Section 3. Approximation algorithms have previously been established only for the version in which there is no restriction on the size of the committee (which includes as a possibility that no candidate is elected). This variant is referred to as the endogenous minimax solution and it also arises in coding theory under the name of the Minimum Radius Problem or the Hamming Center Problem and in computational biology, where it is known as the Closest String Problem. In the context of computational biology, it was shown by Li, Ma and Wang [17] that the endogenous version admits a Polynomial Time Approximation Scheme (PTAS), *i.e.*, a $(1+\epsilon)$-approximation for any constant ϵ (with the running time depending exponentially in $1/\epsilon$). Other constant-factor approximations for the endogenous version had been obtained before [12, 15]. We are not aware of any polynomial-time approximation algorithms or any heuristic approaches for the non-endogenous versions, *i.e.*, in the presence of any upper or lower bounds on the size of the committee. Complexity considerations for winner determination in multi-winner elections have also been addressed recently [21] but not for the minimax solution.

2 Definitions and Notation

We now formally define our problem. We have an election with m ballots and n candidates. Each ballot is a binary vector $v \in \{0,1\}^n$, with the meaning that the ith coordinate of v is 1 if the voter approves of candidate i. For two binary vectors v_i, v_j of the same length, let $H(v_i, v_j)$ denote their Hamming distance, which is the number of coordinates in which they differ. For a vector $v \in \{0,1\}^n$, we will denote by $\mathrm{wt}(v)$ the number of coordinates that are set to 1 in v. The maxscore of a binary vector is defined as the Hamming distance between it and the ballot farthest from it: $\mathrm{maxscore}(v) \equiv \max_i H(v, v_i)$ where v_i is the ith ballot. We first define the problem in its generality.

> **Problem [Bounded-size Minimax (BSM(k_1, k_2))]** Given m ballots, $v_1, \ldots, v_m \in \{0,1\}^n$, and 2 integers k_1, k_2, with $0 \leq k_1, k_2 \leq n$, find a vector v^* such that $k_1 \leq \mathrm{wt}(v^*) \leq k_2$ so as to minimize $\mathrm{maxscore}(v^*)$.

Clearly BSM includes as a special case the endogenous version, which is BSM$(0, n)$, *i.e.*, no restrictions on the size of the committee. Also, since in some committee elections, the size of the committee to be elected is fixed (*e.g.*, the Game Theory Society elections), we are interested in the following variant of BSM with $k_1 = k_2 = k$:

> **Problem [Fixed-size Minimax (FSM(k))]** Given m ballots, $v_1, \ldots, v_m \in \{0,1\}^n$, and an integer k with $1 \leq k \leq n$, find a vector v^* of weight k so as to minimize maxscore(v^*).

In this preliminary version, we focus on elections with committees of fixed size and report our findings for FSM. We briefly mention in the relevant sections throughout the paper as well as in Section 6 which of our results extend to the general BSM problem.

As we show in the next section, BSM and FSM are **NP**-hard. Therefore, a natural approach is to focus on polynomial-time approximation algorithms. We use the standard notion of approximation algorithms, defined below:

Definition 1. *An algorithm for a minimization problem achieves an approximation ratio (or factor) of α ($\alpha \geq 1$), if for every instance of the problem the algorithm outputs a solution with cost at most α times the cost of an optimal solution.*

3 NP-hardness and Approximation Algorithms

We first show that it is unlikely to have a polynomial-time algorithm for the minimax solution. In fact for the endogenous version of BSM, BSM$(0,n)$, **NP**-hardness has already been established by Frances and Litman in [11], where the problem is stated in the context of coding theory. It follows that BSM in general is **NP**-hard. We next show that FSM is also **NP**-hard.

Theorem 1. *FSM is* **NP***-hard.*

Proof. Suppose we had a polynomial-time algorithm for FSM. Then we could run such an algorithm first with $k = 0$, then with $k = 1$ and so on up to $k = n$ and output the best solution. That would give an optimal solution for BSM$(0,n)$. Hence FSM is also **NP**-hard. An alternative proof for the **NP**-hardness of FSM (and consequently of BSM as well) via a reduction from Vertex Cover was also obtained by LeGrand [16]. □

FSM(k) can be solved in polynomial time if k is an absolute constant, since then we can just go through all the $\binom{n}{k}$ different committees and output the best one. Also, if m is an absolute constant then we can express the problem as an integer program with a constant number of constraints, which by a result of Papadimitriou [18] can be solved in polynomial time.

The standard approach in dealing with **NP**-hard problems is to search for approximation algorithms. We will now show that a very simple and fast algorithm achieves an approximation ratio of 3 for FSM(k), for every k. In fact, we will see that the algorithm has a factor of 3 for approval voting problems with much more general constraints.

Before stating the algorithm we need to introduce some more notation. Given a vector v, we will say that u is a *k-completion* of v, if $\mathrm{wt}(u) = k$, and $H(u, v)$ is the minimum possible Hamming distance between v and any vector of weight k. It is very easy to obtain a k-completion for any vector v: if $\mathrm{wt}(v) < k$, then pick any $k - \mathrm{wt}(v)$ coordinates in v that are 0 and set them to 1; if $\mathrm{wt}(v) > k$ then pick any $\mathrm{wt}(v) - k$ coordinates that are set to 1 and set them to 0.

The algorithm is now very simple to state: Pick arbitrarily one of the m ballots, say v_j. Output a k-completion of v_j, say u.

Obviously the algorithm runs in time $O(n)$, independent of the number of voters.

Theorem 2. *The above algorithm achieves an approximation ratio of* 3.

Proof. Let v^* be an optimal solution ($\mathrm{wt}(v^*) = k$) and let $\mathrm{OPT} = \mathrm{maxscore}(v^*) = \max_i H(v^*, v_i)$ be the maximum distance of a ballot from the optimal solution. Let v_j be the ballot picked by the algorithm and let u be the k-completion of v_j that is output by the algorithm. We need to show that for every i, $H(u, v_i) \leq 3\,\mathrm{OPT}$. By the triangle inequality, we know that for every $1 \leq i \leq m$, $H(u, v_i) \leq H(u, v_j) + H(v_j, v_i)$. By applying the triangle inequality again we have:

$$H(u, v_i) \leq H(u, v_j) + H(v_j, v^*) + H(v^*, v_i)$$

Since v^* is an optimal solution, we have that $H(v^*, v_i) \leq \mathrm{OPT}$ and $H(v^*, v_j) \leq \mathrm{OPT}$. Also since u is a k-completion of v_j, by definition $H(u, v_j) \leq H(v^*, v_j) \leq \mathrm{OPT}$. Hence in total we obtain that $H(u, v_i) \leq 3\,\mathrm{OPT}$, as desired. □

Remark 1. *Note that if we know that there is at least one voter of weight k, say* $\mathrm{wt}(v_j) = k$*, then we can prove that the algorithm achieves a ratio of* 2*, since then $u = v_j$ and we need to apply triangle inequality only once.*

Remark 2. *The algorithm can be easily adapted to give a ratio of* 3 *for the BSM version too. We only need to modify the notion of a k-completion accordingly. In fact, for BSM*$(0, n)$*, we can show that the ratio will be* 2.

Note also that the analysis shows that there can be many different solutions that constitute a 3-approximation, since every ballot can potentially have many different k-completions.

Remark 3. Generalized Constraints: *One may define an approval voting problem with constraints that are more general than simply those on the size of the committee (as in BSM). For example, one may have constraints on the number of members elected from a particular subgroup of candidates (quotas), or constraints which require exactly one out of two particular candidates to be in the committee (XOR constraints). Suppose, for any vote vector v, we can compute in polynomial time a* feasible-completion *of v, which is a committee* *that satisfies the constraints, and is closest to v in Hamming distance. Then, we can extend our algorithm to this setting in a natural manner, and prove that it provides a factor 3 approximation.*

We are not aware of any better approximation algorithm for FSM. The endogenous version BSM$(0, n)$, admits a Polynomial Time Approximation Scheme (PTAS), *i.e.*, for every constant ϵ, there exists a $(1 + \epsilon)$-approximation, which is polynomial in n and m and exponential in $1/\epsilon$. The PTAS was obtained in [17], in the context of computational biology. Before that, constant-factor approximations for BSM$(0, n)$ had been obtained in [12] and [15]. We believe that algorithms with such better factors may also be obtainable for FSM(k).

4 Local Search Heuristics for Fixed-size Minimax

Even though the algorithm of Section 3 gives us a theoretical worst-case guarantee (in fact, we may even have a better performance in practice for some instances), a factor 3-approximation may still be far away from acceptably good outcomes. In this section we focus on polynomial-time heuristics, which turn out to perform very well in practice, if not optimally, even though we cannot obtain an improved theoretical worst-case guarantee. The heuristics that we will investigate are based on local search; some of them use the 3-approximation as a starting point and retain its ratio guarantee.

4.1 A Framework for FSM Heuristics

Our overall heuristic approach is as follows. We start from a binary vector (picked according to some rule) and then we investigate if neighboring solutions to the current one improve the current maxscore. The local moves that we allow are removing some candidates from the current committee and adding the same number of candidates in, from the set of candidates who do not belong to the current committee:

1. Start with some $c \in \{0, 1\}^n$.
2. Repeat until maxscore(c) does not change for n loop iterations:
   (a) Let A be the set of all binary vectors reachable from c by flipping up to p number of 0-bits of c to 1 and p 1-bits to 0, where p is an integer constant. (Note that c will necessarily be a member of A.)
   (b) Let $A^\star$ be the set that includes all members of A with smallest maxscore.
   (c) Choose at random one member of $A^\star$ and make it the new c.
3. Take c as the solution.

It is obviously important that the heuristic find a solution in time polynomial in the size of the input. In the worst case, the loop in the heuristic could run for n iterations for each step down in maxscore, so even if the maxscore of the initial c is the largest possible, n, no more than $O(n^2)$ iterations of the loop will be made. Each loop iteration runs in $O(mn^{2p+1})$ time, since the number of swaps to be considered is $O(n^{2p})$ and calculating the maxscore of each takes $O(mn)$ time, so the worst-case running time for the heuristic is $O(mn^{2p+3})$, which is of course polynomial in m and n as long as p is constant.

This heuristic framework has two parameters: the starting point for the binary vector c and the constant p. While many combinations are possible, we will investigate using four different approaches to determining the c starting point and two values of p—1 and 2—resulting in eight specific heuristics. The c starting points are

1. A fixed-size-minisum solution: the set of the k candidates most approved on the ballots.
2. The FSM 3-approximation presented above: a k-completion of a ballot.
3. A random set of k candidates.
4. A k-completion of a ballot with highest maxscore.

For approach 2, the ballot and k-completion are not chosen randomly: Of the ballots with Hamming weight nearest to k, the v^* minimizing $\text{sumscore}(v) \equiv \sum_i H(v^*, v_i)$ is chosen, and bits flipped are chosen to minimize resulting sumscore. The endogenous minimax equivalent of each of these approaches was investigated by LeGrand [16].

We will use the notation $h_{i,j}$ to refer to the heuristic with starting point i and $p = j$. For example, $h_{3,1}$ is the heuristic that starts with a random set of k candidates and swaps at most one 0-bit with one 1-bit at a time.

4.2 Evaluating the Heuristics

We show that the heuristics find good, if not optimal, winner sets on average. The approach is as follows. Given n, m and k, some large number of simulated elections are run. For each election, m ballots of n candidates are generated according to some distribution. The maxscores of the optimal minimax set and the winner sets found using each of the heuristics are then calculated.

We used two ballot-generating distributions: “unbiased” and “biased”. The unbiased distribution simply sets each bit on each ballot to 0 or 1 with equal probability, like flipping an unbiased coin. The biased distribution generates for each candidate two approval probabilities, π_1 and π_2, between 0 and 1 with uniform randomness. The ballots are then divided into three groups. 40% of the ballots are generated according to the π_1 values; that is, each ballot approves each candidate with probability equal to its π_1 value. Another 40%

of the ballots are generated according to the π_2 values, and the remaining 20% are generated as in the unbiased distribution.

We ran 5000 simulated elections in each of seven different configurations, varying n, m, k and the ballot-generating distribution. In the tables of results below, the last column gives the results of running the heuristics 5000 times each on the ballots from the 2003 Game Theory Society council election.

Table 1 gives the highest realized approximation ratio (maxscore found divided by optimal maxscore) found over all 5000 elections for each heuristic, our 3-approximation (with ballot and flipped bits chosen at random), the minisum set (for comparison), and a maximax set. A maximax set is a set of size k that has the highest possible maxscore; it can be found by choosing a ballot with Hamming weight nearest to $n - k$ and performing a $(n - k)$-completion on it.

It can be seen that our 3-approximation in practice performs appreciably better than its guarantee—its ratio was less than 2 for every simulated election. (We were able to find instances of ratio-3 performance for smaller values of n, *e.g.*, 6.) As Table 1 shows, the heuristics reliably find solutions with ratios well below 2, but the average ratios found, given in Table 2, show that the average performance of the heuristics is more impressive still.

Finally, we compared the maxscores found by the heuristics with the worst possible maxscore of a winner set, and scaled them so that the maxscore of the exact minimax set becomes 100% and that of a maximax set becomes 0%, giving a more intuitive performance metric for heuristics. For example, if the minimax set has a maxscore of 12, a maximax set has a maxscore of 20 and a heuristic finds a solution with maxscore 13, the heuristic's scaled performance for that election will be $(20 - 13)/(20 - 12) = 87.5\%$. The averages of these scaled performances can be found in Table 3.

We draw the following conclusions from our experiments.

- The heuristics perform well. Given the ballot distributions we used, very rarely would a heuristic find a solution that is unacceptably poorer than the optimal minimax solution. In particular, $h_{2,1}$ and $h_{2,2}$ vastly outperform the plain 3-approximation (while retaining its ratio-3 guarantee) with only a modest increase in running time.

- The heuristics perform significantly better on average when $p = 2$ than when $p = 1$. Increasing p further can be expected to improve performance further, at the expense of increased running time.

- Comparing the performance of the heuristics with equal p, all four perform similarly overall, but the best c-starting-point approach on average seems to be the first (a fixed-size-minisum solution); it significantly outperforms the other three sometimes (*e.g.*, when $p = 1$ in the unbiased-coin cases with 50 ballots) and is never outperformed by them with any statistical significance.

Table 1: Largest approximation ratios found for local search heuristics

n	20	20	24	20	20	24
k	10	10	12	10	10	12
m	50	200	50	50	200	161
ballots	unbiased	unbiased	unbiased	biased	biased	GTS 2003
minimax	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
$h_{1,1}$	1.1818	1.0769	1.1538	1.2000	1.0909	1.0714
$h_{2,1}$	1.1818	1.0769	1.1538	1.2000	1.1818	1.0714
$h_{3,1}$	1.1818	1.0769	1.1538	1.2000	1.1818	1.0714
$h_{4,1}$	1.1818	1.0769	1.1538	1.2000	1.1818	1.0714
$h_{1,2}$	1.0909	1.0769	1.0769	1.1000	1.0833	1.0000
$h_{2,2}$	1.0909	1.0769	1.0769	1.1000	1.0833	1.0000
$h_{3,2}$	1.0909	1.0769	1.0769	1.1000	1.0833	1.0000
$h_{4,2}$	1.0909	1.0769	1.0769	1.1000	1.0833	1.0000
3-approx.	1.6667	1.4615	1.6154	1.8182	1.5833	1.3571
minisum	1.5455	1.4615	1.6923	1.6364	1.5833	1.2143
maximax	1.8182	1.5385	1.8462	2.2222	1.8182	1.7143

Table 2: Average approximation ratios found for local search heuristics

n	20	20	24	20	20	24
k	10	10	12	10	10	12
m	50	200	50	50	200	161
ballots	unbiased	unbiased	unbiased	biased	biased	GTS 2003
minimax	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
$h_{1,1}$	1.0058	1.0320	1.0093	1.0083	1.0210	1.0012
$h_{2,1}$	1.0118	1.0365	1.0147	1.0112	1.0251	1.0017
$h_{3,1}$	1.0122	1.0370	1.0151	1.0122	1.0262	1.0057
$h_{4,1}$	1.0117	1.0364	1.0149	1.0116	1.0262	1.0059
$h_{1,2}$	1.0004	1.0129	1.0011	1.0004	1.0025	1.0000
$h_{2,2}$	1.0004	1.0164	1.0014	1.0005	1.0029	1.0000
$h_{3,2}$	1.0004	1.0164	1.0018	1.0005	1.0031	1.0000
$h_{4,2}$	1.0003	1.0167	1.0014	1.0006	1.0029	1.0000
3-approx.	1.2477	1.1871	1.2567	1.3121	1.2424	1.3571
minisum	1.1650	1.1521	1.1665	1.2119	1.1932	1.2143
maximax	1.6746	1.4895	1.7320	1.8509	1.6302	1.7143

Table 3: Average scaled performance of local search heuristics

n	20	20	24	20	20	24
k	10	10	12	10	10	12
m	50	200	50	50	200	161
ballots	unbiased	unbiased	unbiased	biased	biased	GTS '03
$h_{1,1}$	99.18%	94.05%	98.83%	99.07%	96.86%	99.82%
$h_{2,1}$	98.33%	93.23%	98.11%	98.74%	96.24%	99.77%
$h_{3,1}$	98.27%	93.13%	98.06%	98.62%	96.06%	99.20%
$h_{4,1}$	98.33%	93.24%	98.08%	98.68%	96.08%	99.18%
$h_{1,2}$	99.95%	97.60%	99.87%	99.95%	99.63%	100.00%
$h_{2,2}$	99.95%	96.96%	99.83%	99.94%	99.57%	100.00%
$h_{3,2}$	99.95%	96.95%	99.79%	99.94%	99.54%	100.00%
$h_{4,2}$	99.96%	96.89%	99.83%	99.94%	99.57%	100.00%
3-approx.	63.36%	62.31%	65.04%	63.36%	61.73%	50.00%
minisum	75.57%	69.40%	77.29%	75.04%	69.49%	70.00%

5 Manipulation

Gibbard [13] and Satterthwaite [22] proved independently that any election system that chooses exactly one winner from a slate of more than two candidates and satisfies a few obviously desirable assumptions (such as an absence of bias for some candidates over others) is sometimes manipulable. In other words, there exist situations under any reasonable single-winner system in which some voters can gain better outcomes for themselves by voting insincerely.

Happily, the Gibbard–Satterthwaite theorem does not apply to the minimax and minisum solutions since they are free to choose winner sets of any size. In fact, the minisum procedure is completely nonmanipulable when any set of winners is allowed, as shown by Brams *et al.* [4]. This is true because a minisum election with n candidates is exactly equivalent to n elections of two "candidates" each: approve or disapprove that candidate. Since a voter's decision to approve or disapprove one candidate has absolutely no effect on whether other candidates are chosen as winners, there is no more effective strategy than voting sincerely. Consequently, it is reasonable to expect a set of minisum ballots to have been sincerely voted.

Unfortunately, in addition to being possibly hard to compute exactly, the minimax solution is easily shown to be manipulable for the FSM version.

Definition 2. *Fix an approval voting algorithm A and a set of ballots $\mathbf{v} = (v_1, v_2, ..., v_m)$. Fix a voter i, and let $\mathbf{v^{-i}}$ denote the ballots of the rest of the voters. The loss $L^i_A(\mathbf{v})$ of voter i is defined as $H(v_i, A(\mathbf{v}))$. Algorithm A is said to be manipulable if there exist ballots $\mathbf{v}$, a voter i, and a ballot $v' \neq v_i$, s.t. $L^i_A(v_i, \mathbf{v^{-i}}) > L^i_A(v', \mathbf{v^{-i}})$.*

Theorem 3. *Any algorithm that computes an optimal solution for FSM is manipulable.*

Proof. Consider the following set of *sincere* ballots:

00110, **00011**, **00111**, **00001**, **10111**, **01111**

The minimax winner sets of size 2 are **00011** and **00101** with a maxscore of 2. The first voter, however, could manipulate the result by voting the insincere ballot **11110**. In that case, it can be checked that the optimal solution of size 2 is **00110**, which is exactly the most preferred outcome of the first voter. □

An analogous example for the endogenous version was provided by LeGrand [16]. These examples illustrate a general guideline to manipulating a minimax election: If there are candidates of which the majority disapproves, a voter may be able to vote safely in favor of those candidates to force more agreement with his relatively controversial choices. Put another way, if the minimax set can be seen as a kind of average of all ballots, a voter can move his ballot farther away from the current consensus to drag it closer to his ideal outcome. The minimax solution is extremely sensitive to "outliers" compared to the minisum solution, in much the same way that the average of a sample of data is more sensitive to outliers than the median.

Although algorithms that always compute an optimal minimax solution are manipulable, the same may not be true if we allow approximation algorithms. The following theorem shows that we can have nonmanipulable algorithms if we are willing to settle for approximate solutions.

Theorem 4. *The voting procedure that results from using the* 3*-approximation algorithm described in Section 3 is nonmanipulable.*

Proof. The algorithm picks a ballot v_j at random and outputs a k-completion of v_j. For a voter i, if the algorithm did not pick v_i, then the voter cannot change the output of the algorithm by lying. Furthermore, if the algorithm did pick v_i, then the best outcomes of size k for v_i are precisely all the k-completions of v_i. Therefore, by lying, the voter cannot possibly alter the outcome to his benefit. □

We conjecture that the heuristics of Section 4 are also hard to manipulate. Although we do not have a proof for this, our intuition is the following. The heuristics use a lot of randomization—in all of them, either the starting point or the local move is based on a random choice. It therefore seems unlikely for a voter to be able to change his vote in such a way that the random choices of the algorithms will (even in expectation) work towards his benefit.

The above theorems give rise to the following question:

Question 1. *What is the smallest value of* α *for which there exists a nonmanipulable polynomial-time approximation algorithm with ratio* α*?*

Another interesting question is whether there exist algorithms (either exact or approximate) which are **NP**-hard to manipulate (i.e., although they are manipulable, the voter would have to solve an **NP**-hard problem in order to cheat). See Bartholdi *et al.* [1, 2] as well as more recent work [5, 6, 7, 8] along this line of research. In another recent work [20], average-case complexity is introduced as a complexity measure for manipulation instead of worst-case complexity (**NP**-hardness).

6 Discussion and Future Work

We have initiated a study of the computational issues involved in committee elections. Our results, along with the analysis of the endogenous version by LeGrand [16], show that local search heuristics perform very well in approximating the minimax solution in polynomial time.

There are still many interesting directions for future research. In terms of heuristic approaches, we are planning to adjust our heuristics for the weighted version of the minimax solution, as introduced by Brams *et al.* [4]. This version takes into account both the number of voters that vote each distinct ballot and the proximity of each ballot to the other voters' ballots. We are also investigating variations of local search that may improve even more the performance, *e.g.*, can there be a better starting point in our heuristics, or can we enrich the set of local moves without increasing too much the running time? Another interesting topic would be to compare local search with other heuristic approaches that could be adapted for our problem, like simulated annealing or genetic algorithms.

In terms of theoretical results, the most compelling question is to determine the best approximation ratio that can be achieved in polynomial time for the minimax solution. The questions stated in Section 5 regarding manipulation would also be interesting to pursue.

7 Acknowledgements

We would like to than Eric van Damme, secretary-treasurer of the Game Theory Society, for letting us use the ballot data of the 2003 Game Theory Society council election in our experiments. We would also like to thank Steven Brams for introducing us to the problem and for his valuable comments and pointers to the literature.

References

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

[2] J. J. Bartholdi III, C. A. Tovey, and M. A. Trick. How hard is it to control an election? *Mathematical Computational Modeling*, 16(8/9):27–40, 1992.

[3] S. J. Brams, D. M. Kilgour, and M. R. Sanver. A minimax procedure for negotiating multilateral treaties. In M. Wiberg, editor, *Reasoned Choices: Essays in Honor of Hannu Nurmi*. Finnish Political Science Association, 2004.

[4] S. J. Brams, D. M. Kilgour, and M. R. Sanver. A minimax procedure for electing committees. manuscript, 2006.

[5] V. Conitzer, J. Lang, and T. Sandholm. How many candidates are needed to make elections hard to manipulate? In *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-03)*, pages 201–214, Bloomington, Indiana, 2003.

[6] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 314–319, Edmonton, Canada, 2002.

[7] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 781–788, Acapulco, Mexico, 2003.

[8] E. Elkind and H. Lipmaa. Hybrid voting protocols and hardness of manipulation. In *The 16th Annual International Symposium on Algorithms and Computation (ISAAC 2005)*, pages 206–215, Sanya, Hainan, China, Dec. 2005.

[9] E. Ephrati and J. Rosenschein. The clarke tax as a consensus mechanism among automated agents. In *AAAI*, pages 173–178, 1991.

[10] E. Ephrati and J. Rosenschein. Multi-agent planning as a dynamic search for social consensus. In *IJCAI*, pages 423–429, 1993.

[11] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30:113–119, Mar. 1997.

[12] L. Gasieniec, J. Jansson, and A. Lingas. Efficient approximation algorithms for the Hamming center problem. In *SODA*, 1999.

[13] A. Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(3):587–601, May 1973.

[14] D. M. Kilgour, S. J. Brams, and M. R. Sanver. How to elect a representative committee using approval balloting. In B. Simeone and F. Pukelsheim, editors, *Mathematics and Democracy: Recent Advances in Voting Systems and Collective Choice*. Springer, forthcoming, 2007.

[15] J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. *Information and Computation*, 185:41–55, 2003.

[16] R. LeGrand. Analysis of the minimax procedure. Technical Report WUCSE-2004-67, Department of Computer Science and Engineering, Washington University, St. Louis, Missouri, Nov. 2004.

[17] M. Li, B. Ma, and S. Wang. Finding similar regions in many strings. In *STOC*, pages 473–482, 1999.

[18] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM*, 28(4):765–768, 1981.

[19] D. Pennock, E. Horvitz, and C. L. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI*, pages 729–734, 2000.

[20] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. In *AAMAS*, pages 497–504, Hakodate, Japan, 2006.

[21] A. D. Procaccia, J. S. Rosenschein, and A. Zohar. Multi-winner elections: Complexity of manipulation, control and winner-determination. In *IJCAI*, Hyderabad, India, 2007.

[22] M. A. Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, Apr. 1975.

Rob LeGrand
Washington University in St. Louis
St. Louis, Missouri, U.S.A.
Email: `legrand@cse.wustl.edu`

Evangelos Markakis
University of Toronto
Toronto ON M5S3G4, Canada
Email: `vangelis@cs.toronto.edu`

Aranyak Mehta
IBM Almaden Research Center
San Jose, CA 95120, USA
Email: `mehtaa@us.ibm.com`