Analysis of the Minimax Procedure

Rob LeGrand
Washington University, St. Louis, Missouri
Department of Computer Science and Engineering
legrand@cse.wustl.edu

October 7, 2005

Abstract

Minimax is a recently proposed procedure for electing committees, but it cannot be applied without an understanding of its computational complexity and its susceptibility to manipulation. We derive upper and lower bounds on the "maxscore" of a minimax winner set for a given election. We examine two compelling variations of minimax and prove both to be NP-complete. We present a heuristic for calculating a minimax winner set and present experiments based on its use. Finally, we present a general strategy for manipulating a minimax election.

1 Introduction

Disagreement has long existed on the best way to elect a fixed-size committee, *i.e.*, to elect $m > 0$ winners from $k > m$ alternatives; most such systems, like the single transferablevote, use ranked ballots and have relatively complex counting procedures. However, the notion of electing a committee of endogenous size, so that the size of the winner set depends on the ballots and is allowed to range fully from 0 to k, is relatively new.

To motivate this new concept, imagine a political party's primary election that aims to move forward the set of candidates that would be most acceptable to the party as a whole.[1] In this situation, the party might have as their main goal to choose an outcome that would satisfy the maximum number of voters or, alternatively, make the least satisfied voter as satisfied as possible.

For single-winner elections, approval voting (AV) has been shown to satisfy many desirable criteria [1]. AV chooses as the winner the alternative that receives the single greatest number of votes from a set of approval ballots, which allow a vote for or against each alternative. AV aims to ensure that the winning alternative is acceptable to many voters.

In that single-winner context, it is presumed that a voter be satisfied if and only if one of his approved alternatives is elected. But if any number of winners can be chosen, the satisfaction of an individual voter must be measured differently. Simplest is to assume that each voter considers the presence or absence of each alternative in the winner set to be equally important; then, a voter's satisfaction with an outcome is inversely related to the Hamming distance between his ballot and the winner set. The Hamming distance between two sets A and B is $|A - B| + |B - A|$, the number of elements that are in one set and not the other. Similarly, the Hamming distance between two bit strings is the number of ones in the XOR of the strings.

[1] General elections that use the one-vote plurality voting system have necessitated primaries that select only one candidate. Through the well-known vote-splitting effect, plurality discourages the running of multiple candidates with similar views. Here we assume the general election uses a voting system immune to vote-splitting, such as approval voting, that would not discourage running multiple similar candidates.

1.1 Minisum

At first glance, it may seem that the best way to accomplish the above party goal using approval ballots would be to elect alternatives that are approved by the most voters. More prescisely, if there are k alternatives up for election and the number of winners is allowed to range between 0 and k inclusive—in other words, each alternative is either elected or not without restriction on the size of the set of winners—the most intuitive system lets the voters cast approval ballots and then elects those alternatives that earn more than 50% approval. Brams, Kilgour and Sanver [2] call this procedure minisum since it can be shown to minimize the sum of Hamming distances between the set of winners and each ballot.

Minisum, however intuitive and simple to count, could be seen as inadequate in this case. It may be too subject to the whims of a majority faction and end up alienating a subset of voters. For example, consider the ballots

1. **000000**
2. **000001**
3. **000010**
4. **000100**
5. **111111**

where **000001** means approval of the last alternative and disapproval of the first five. Since there is a majority disapproving each alternative, the minisum procedure would choose the set **000000**, electing none of the alternatives and likely completely alienating the last voter. This "tyranny of the majority" could be unacceptably divisive in a party primary context.

1.2 Minimax

Brams, Kilgour and Sanver [2] have introduced the minimax procedure, a new way to elect a group of alternatives of unconstrained size using approval ballots. In contrast to the minisum procedure, the minimax procedure minimizes the Hamming distance between the set of winners and the ballot farthest from that set in terms of Hamming distance.

The above example can be used to show that the two procedures are different. Minisum returned the winner set **000000**, electing no alternative, while minimax would return the winner set **000111**, only 3 away from all ballots, electing three of the six alternatives. The minimax outcome has a "sumscore" (total Hamming distance) of 12, greater than the minisum outcome, which has a sumscore of only 9. The minisum outcome, **000000**, has Hamming distance 6 from the ballot **111111**, greater than the minimax outcome, which has Hamming distance only 3 from all ballots. So these two evaluation measures of winner sets are sometimes at odds.

2 Bounds for minimax

A problem with using minimax in real-world applications is that, unlike minisum, there is no known algorithm for finding a solution set (an optimal winner set) that is substantially more efficient than exhaustive search of all sets of alternatives. Until and unless such an algorithm is found, it may be worth trying to reduce the search space.

One approach is to establish bounds on the "maxscore" of the solution set(s). First, construct a Hamming-distance matrix on the ballots. Say there are six alternatives and five ballots:

1. **000011**
2. **000011**
3. **000111**
4. **001111**
5. **011001**

The corresponding Hamming-distance matrix is

	1	2	3	4	5	max
1	0	0	1	2	3	3
2	0	0	1	2	3	3
3	1	1	0	1	4	4
4	2	2	1	0	3	3
5	3	3	4	3	0	4

So the Hamming distance between ballots 3 and 5 is 4, and so on. The "max" column is the maxscore of a ballot taken as the winner set; for example, ballot 4, **001111**, is no farther than 3 from any other ballot.

One way to find an upper bound on the maxscore of the solution(s) is by construction: Calculate the maxscore of several possible winner sets and take the lowest. Any collection of winner sets could provide an upper bound, but some collections tend to produce a tighter bound than others. One approach is to try each ballot as the winner set and take the lowest maxscore as the upper bound. Equivalently, find the smallest entry in the max column of the Hamming-distance matrix. The best solutions cannot have a higher maxscore. In the above example, ballots 1, 2 and 4 have a maxscore of 3, so no best solution can have a higher (worse) maxscore.

A lower bound can be found, but it is necessarily nonconstructive. Find the two ballots that are farthest from each other—this can be done by finding the largest entry in the Hamming-distance matrix. Now a best solution is not likely to be too far from either ballot; in particular, on the alternatives on which the two ballots disagree, it aims to agree with each of the two a roughly equal number of times to minimize its maxscore. Thus a lower bound can be set at half of the maximum entry (rounded up). In the above example, the maximum Hamming-distance-matrix entry is 4, corresponding to ballots 3 and 5, which disagree on four alternatives. So any best solution must either agree with ballot 3 on 2 alternatives and with ballot 5 on 2 alternatives or have a maxscore of 3 or more, so the lower bound would be 2. Finding this lower bound does not always give a winner set that realizes the bound; indeed, sometimes there is none. But the bound is useful: If a set that realizes it is found, it must necessarily be a minimax set.

The unique solution set is this example turns out to be **001011** with a maxscore of 2, realizing the lower bound of 2, while the minisum procedure would choose **000011** as the best solution set.

3 NP-completeness of minimax variants

If the minimax procedure were to be used in real-world elections and decisions, one would need an algorithm for calculating an optimal set of winners. A brute-force approach gives an algorithm that would calculate the maxscore for every possible solution and report one with the smallest maxscore; if there are k alternatives and n ballots, there are 2^k possible sets to try and calculating the maxscore of each possible winner set takes time $O(kn)$, so this algorithm runs in $O(2^k kn)$ time. Ideally, an algorithm substantially more efficient than trying all possible sets could be found. Unfortunately, it can be shown that the problem is NP-hard, so an efficient (polynomial-time) algorithm is unlikely to be found.

The calculation of the winner set according to the minimax procedure can be recast as a decision problem of the form used in [5], using sets instead of bit strings:

MINIMAX

INSTANCE: Set A with $|A| = k$; collection B of n subsets $B_1, B_2, \ldots B_n$ of A; nonnegative integer $d < k$.
QUESTION: Is there a subset W of A such that $|W - B_i| + |B_i - W| \leq d$ for all i?

Recasting the example from section 2 in the set-style form would give

$$\{\{E, F\}, \{E, F\}, \{D, E, F\}, \{C, D, E, F\}, \{B, C, F\}\}$$

as the collection of ballots if the alternatives are $\{A, B, C, D, E, F\}$. Then the minimax set is $\{C, E, F\}$ and the minisum set is $\{E, F\}$.

3.1 Fixed-size minimax

A similar decision problem, which may be seen as more useful in electing real-world committees:

FIXED-SIZE MINIMAX (FSM)

INSTANCE: Set A with $|A| = k$; collection B of n subsets $B_1, B_2, \ldots B_n$ of A; nonnegative integer $d < k$; nonnegative integer $m \leq k$.
QUESTION: Is there a subset W of A such that $|W| = m$ and $|W - B_i| + |B_i - W| \leq d$ for all i?

The fixed-size minimax problem can be shown to be NP-hard by using the vertex cover problem, which is known to be NP-complete [5]:

VERTEX COVER (VC)

INSTANCE: Graph $G = (V, E)$; positive integer $c \leq |V|$.
QUESTION: Is there a vertex cover of size c or less for G, i.e., a subset $V' \subseteq V$ with $|V'| \leq c$ such that for each edge $E_i = u, v \in E$ at least one of u and v belongs to V'?

Any vertex cover problem can be reduced to a fixed-size minimax problem by letting $A = V$, $B = E$, $d = c$ and $m = c$; the vertices of the graph become the FSM alternatives and the edges become the ballots. Note that each so-constructed ballot necessarily votes for exactly two alternatives since each edge is a set of exactly two vertices.

It can be seen that if the VC problem should return a "yes" answer, the new FSM problem will give a "yes": If there is a vertex cover V' of size c or less, then there must be some vertex cover $V'' \supseteq V'$ of size exactly c. So $|V''| = c$ and, by definition of vertex cover,

$$V'' \cap E_i \neq \emptyset$$

for each edge $E_i \in E$. Then for each edge E_i, since V'' overlaps E_i,

$$|V'' - E_i| < |V''|$$

and so

$$|V'' - E_i| \leq |V''| - 1$$

Also, for each edge E_i,

$$|E_i - V''| < |E_i| = 2$$

so

$$|E_i - V''| \leq 1$$

Adding the inequalities gives

$$|V'' - E_i| + |E_i - V''| \leq |V''| - 1 + 1 = |V''| = c$$

for each E_i. Now consider the FSM problem corresponding to this VC problem according to the above reduction; call the set of alternatives in the FSM problem corresponding to V'' W. $|W| = c = m$, and if

$$|V'' - E_i| + |E_i - V''| \leq c$$

then $|W - B_i| + |B_i - W| \leq c = d$ must be true for each ballot B_i. Therefore W is a subset of A that satisfies the requirements of FSM and this FSM problem must return a "yes" answer.

Conversely, if the new FSM problem returns a "yes", the VC problem should also give a "yes": If there is a winner set $|W| \subseteq A$ of size m that satisfies

$$|W - B_i| + |B_i - W| \leq d$$

for all i, the set of vertices $V^* \subseteq V$ that corresponds to W according to the reduction must satisfy

$$|V^* - E_i| + |E_i - V^*| \leq d = c$$

for all $E_i \in E$ and $|V^*| = m = c$. Therefore

$$|V^* - E_i| + |E_i - V^*| \leq |V^*|$$

for all E_i. Now if there were some $E^* \in E$ such that

$$V^* \cap E^* = \emptyset$$

then it would have to be true that

$$|V^* - E^*| = |V^*|$$

and

$$|E^* - V^*| = |E^*| = 2$$

It would follow that

$$|V^* - E^*| + |E^* - V^*| = |V^*| + 2 > |V^*|$$

But

$$|V^* - E_i| + |E_i - V^*| \leq |V^*|$$

for all E_i, so the assumed E^* cannot exist and $V^* \cap E_i \neq \emptyset$ must be true for all E_i. It follows, of course, that V^* is a vertex cover for G, thus a "yes" answer should be returned for the VC problem.

Consequently, since one problem provides a "yes" answer if and only if the other problem does, the two problems' answers always agree, so any VC problem can be solved by reducing it to a FSM problem. This reduction from VC to FSM can be constructed in polynomial time. Therefore, since VC is NP-complete, FSM must be NP-hard. Actually, checking whether a given W gives a "yes" answer to the question can be done in time $O(kn)$, so a subset of A can be guessed nondeterministically and checked in polynomial time. Therefore FSM is in NP and so is NP-complete.

3.2 Bounded-size minimax

Yet another NP-complete problem is a further generalization of minimax:

BOUNDED-SIZE MINIMAX (BSM)
INSTANCE: Set A with $|A| = k$; collection B of n subsets $B_1, B_2, \ldots B_n$ of A; nonnegative integer $d < k$; nonnegative integers $m_{min} \leq m_{max} \leq k$.
QUESTION: Is there a subset W of A such that $m_{min} \leq |W| \leq m_{max}$ and $|W - B_i| + |B_i - W| \leq d$ for all i?

BSM can be easily seen to be NP-complete by a trivial reduction of FSM: Any FSM problem can be solved by reducing it to a BSM problem simply by setting $m_{min} = m$ and $m_{max} = m$. This result is unfortunate, since BSM would seem to be the most generally useful problem to be able to solve for real-world committee elections.

3.3 Minimax

Unfortunately, the previous results are still not enough to prove that the original MINIMAX decision problem is NP-complete. Frances and Litman [4] showed that a decision problem based on the radius of a binary code is NP-complete by reducing the 3-SATISFIABILITY problem to it. The above MINIMAX problem is equivalent to their radius decision problem and is thus also NP-complete.

3.4 Minimax optimization problems

If the optimization problems corresponding to FSM and BSM (finding a best minimax solution W of desired cardinality) could be solved in time polynomial in k and n, the decision problem could then be solved polynomially by adding a simple test: If $|W - B_i| + |B_i - W| \leq d$ for all i, then the answer to the problem is "yes", otherwise "no". But the decision problems are NP-complete and unlikely to have a polynomial-time algorithm, so it is as least as unlikely that an efficient algorithm for the optimization problems will be found.

4 Heuristics for minimax

Since finding an optimal minimax winner set is NP-hard, it would be useful to have a polynomial-time heuristic that performs well on average, if not optimally, in practice.

4.1 A framework for minimax heuristics

One approach to creating such a heuristic is based on the lower bound on maxscores from section 2:

1. Construct the Hamming-distance matrix.
2. Find the largest entry in the matrix and, of the two corresponding ballots, call the one with the smaller sumscore A and the other B. (Note that ballots A and B are the two farthest away from each other and necessarily have the same maxscore.)
3. Start the current solution C at A.
4. Repeat until C does not change:
   (a) Initialize a collection L of sets to be empty.
   (b) For each alternative x on which A and B differ:
   - If $x \in C$, add $C - \{x\}$ to L.
   - If $x \notin C$, add $C \cup \{x\}$ to L.

   (c) Compare the sets in L and, of the ones with the smallest maxscore, call the one with the smallest minisum score D. If D has a smaller maxscore, or equal maxscore and smaller sumscore, than the current C, make it the new C.
5. Take C as the minimax "solution".

This heuristic can be seen to run in polynomial time by showing that a well-ordered metric, $(\text{maxscore}(C), \text{sumscore}(C))$ sorted in lexicographical order, decreases monotonically with each run of the loop. Each time C is changed, either its maxscore is decreased or its maxscore is unchanged and its sumscore is decreased. $\text{maxscore}(C)$ is bounded by 0 and k and $\text{sumscore}(C)$ is bounded by 0 and kn, so the loop could run no more than $k \cdot kn = k^2n$ times. The worst-case running time of the heuristic turns out to be in $O(k^3n^2)$.

Essentially the heuristic seeks to find the winner set that agrees with each of A and B on about half of the alternatives on which A and B differ. The approach could be called greedy in that it starts at one possible winner set (A) and tries to improve it as a solution by making small changes to it, never increasing its maxscore. It could also be called optimistic in that it attempts to realize the lower bound from section 2; if the optimal solution is worse than that lower bound, the heuristic may be less likely to find it.

As one example, consider the ballots from section 2:

1. **000011**
2. **000011**
3. **000111**
4. **001111**
5. **011001**

The Hamming-distance matrix reveals that ballots 3 and 5 are farthest from one another with maxscores of 4. Since the sumscore of ballot 3 is only 7 and that of ballot 5 is 13, ballot 3 is called ballot A and ballot 2 is ballot B. So initially C is **000111**. The collection L in the first run of the loop consists of all sets that disagree with C (while agreeing with B) on exactly one alternative:

000011, **000101**, **001111**, **010111**

All of these have a maxscore of 5 but **000011** has the uniquely smallest sumscore of 6, so it becomes the new C. In the next run of the loop, **001011** has the uniquely smallest maxscore, 2, of the options in L; **001011** cannot be improved as a solution by flipping on a single alternative, so it becomes the heuristic solution. In this example, the heuristic successfully finds the optimal winner set.

Another example shows that the heuristic can fail to find an optimal winner set:

1. **000100**
2. **101011**
3. **011011**
4. **100100**
5. **010001**
6. **100001**
7. **111000**

The corresponding Hamming-distance matrix is

	1	2	3	4	5	6	7	max
1	0	5	5	1	3	3	4	5
2	5	0	2	4	4	2	3	5
3	5	2	0	6	2	4	3	6
4	1	4	6	0	4	2	3	6
5	3	4	2	4	0	2	3	4
6	3	2	4	2	2	0	3	4
7	4	3	3	3	3	3	0	4

The Hamming-distance matrix reveals that ballots 3 and 4 are farthest from one another with maxscores of 6; they disagree on all alternatives. Since the sumscore of ballot 4 is only 20 it is called ballot A and ballot 3 is ballot B. So initially C is **100100**. The collection L in the first run of the loop consists of all sets that disagree with C on exactly one alternative:

000100, **100000**, **100101**, **100110**, **101100**, **110100**

All of these but **100100** have a maxscore of 5 but **100000** has the uniquely smallest sumscore of 17, so it becomes the new C. In the next run of the loop, five adjacent sets have a maxscore of 4 and, of those, **100001** has the uniquely smallest sumscore of 16. **100001** cannot be improved as a solution by flipping on a single alternative, so it becomes the heuristic solution. But the true optimal minimax winner set is **001000** with a maxscore of 3, so the heuristic fails in this case.

The same basic heuristic idea can be used with different starting sets. For example, instead of starting at ballot A, one could start at the optimal minisum set, which is easily computed, and move greedily in any direction from there. Some possibilities:

Heuristic 1 Start at ballot A; move toward ballot B.

Heuristic 2 Start at ballot A; move anywhere.

Heuristic 3 Start at the optimal minisum set; move anywhere.

Heuristic 4 Start at a randomly generated set; move anywhere.

Heuristic 5 Start at a randomly selected ballot; move anywhere.

4.2 Evaluating the heuristics

In this section we show that the heuristics find good, if not optimal, winner sets on average. The test approach is as follows: Given k and n, each test election randomly generates n ballots of k alternatives, where each ballot approves each alternative with uniform $\frac{1}{2}$ probability. The maxscores of the optimal minimax set, the optimal minisum set and the winner sets found using each of the five heuristics are calculated along with the upper bound described in section 2. These maxscores are compared with k, the worst possible maxscore of a winner set, and scaled so that the maxscore of the exact minimax set becomes 100% and k becomes 0%, giving a performance metric for heuristics. A heuristic can score no better than 100% and no worse than 0%.

As an example election, say $k = 7$ and $n = 7$ and the following ballots are randomly generated:

1. **0011110**
2. **1000010**
3. **1101111**
4. **0011010**
5. **1010110**
6. **0110011**
7. **1001101**

The unique optimal minimax set turns out to be **1011011** with a maxscore of 3. Any winner set that disagrees with any one ballot on every alternative, a sort of "maximax" set, will have a maxscore of $k = 7$, the worst score possible. So the best maxscore a heuristic could achieve is 3 and the worst is 7. The following table shows the results of running each heuristic and the minisum procedure on the above election and scaling the resulting maxscores into percentage scores according to the above. (Note that the winner set found is not necessarily unique; there are $n = 7$ maximax sets, for example, found by complementing each ballot, but all have a maxscore of 7.)

	winner set found	maxscore	scaled percentage score
minimax set	**1011011**	3	$(7-3)/(7-3) = 100\%$
heuristic 1	**1011011**	3	$(7-3)/(7-3) = 100\%$
heuristic 2	**1010110**	4	$(7-4)/(7-3) = 75\%$
heuristic 3	**1011011**	3	$(7-3)/(7-3) = 100\%$
heuristic 4	**1010110**	4	$(7-4)/(7-3) = 75\%$
heuristic 5	**1010110**	4	$(7-4)/(7-3) = 75\%$
minisum set	**1011110**	5	$(7-5)/(7-3) = 50\%$
upper bound	**0011110**	4	$(7-4)/(7-3) = 75\%$
maximax set	**1100001**	7	$(7-7)/(7-3) = 0\%$

The following table shows the average of the above scaled percentage scores over 10000 elections using each of several combinations of k and n.

k	6	6	6	12	12	12	18	18	18
n	5	25	125	5	25	125	5	25	125
minimax set	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
heuristic 1	99.67%	92.71%	99.08%	99.23%	92.07%	94.70%	98.69%	93.14%	93.34%
heuristic 2	99.67%	92.76%	99.08%	99.46%	93.37%	94.73%	99.24%	94.69%	94.21%
heuristic 3	99.10%	88.26%	95.38%	98.87%	92.23%	93.16%	98.80%	94.30%	91.46%
heuristic 4	99.09%	94.35%	99.69%	98.28%	93.69%	94.95%	97.86%	94.81%	94.98%
heuristic 5	99.35%	93.47%	99.52%	98.83%	93.50%	94.94%	98.42%	94.39%	94.68%
minisum set	83.19%	56.90%	48.68%	86.78%	71.29%	65.19%	88.46%	76.93%	70.47%
upper bound	79.86%	86.61%	99.99%	75.50%	81.84%	94.27%	74.13%	82.15%	86.91%
maximax set	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

To estimate how significant the differences among heuristics are, confidence intervals for their mean scaled percentage scores can be calculated. For example, consider heuristic 1 in the $k = 12$, $n = 25$ case: It scored 100% (by finding the optimal minimax set) in 3851 simulated elections, 83.33% in another 6050 and 80% or less in 99; its average scaled score is 92.07%. But how close to heuristic 1's true mean score[2] can we be sure 92.07% is? Assuming the data have a normally distributed mean, which is a safe assumption when there are 10000 data points, we can be approximately 95% certain it falls within the range

$$\left(\bar{x} - z_{\alpha/2} \cdot \frac{s}{\sqrt{n}}, \bar{x} + z_{\alpha/2} \cdot \frac{s}{\sqrt{n}}\right)$$

where $\bar{x}$ is the mean of the sample (92.07), α is the maximum probability that the mean will be excluded from the confidence interval (.05), z_p is the $100(1-p)$th percentile of the standard normal distribution (1.96 for $p = .025$), s is the sample standard deviation (9.878) and n is the sample size (10000). The 95% confidence interval is thus

$$\left(\bar{x} - z_{\alpha/2} \cdot \frac{s}{\sqrt{n}}, \bar{x} + z_{\alpha/2} \cdot \frac{s}{\sqrt{n}}\right) = \left(92.07 - z_{.025} \cdot \frac{9.878}{\sqrt{10000}}, 92.07 + z_{.025} \cdot \frac{9.878}{\sqrt{10000}}\right)$$

$$= \left(92.07 - 1.96 \cdot \frac{9.878}{100}, 92.07 + 1.96 \cdot \frac{9.878}{100}\right) \approx (91.87, 92.26)$$

Calculating the 95% confidence intervals for all five heuristics gives

	95% CI
heuristic 1	(91.87, 92.26)
heuristic 2	(93.18, 93.55)
heuristic 3	(92.04, 92.42)
heuristic 4	(93.51, 93.88)
heuristic 5	(93.31, 93.68)

We can conclude that, with more than 95% certainty, heuristics 2, 4 and 5 perform better in the $k = 12$, $n = 25$ case than heuristics 1 and 3. While heuristic 4 seems to perform best, we cannot conclude such at the 95% confidence level; a lower confidence level such as 90% (or more data points) would be required.

[2] A heuristic's true mean score could in principle be found by generating every possible combination of n ballots over k alternatives and running the heuristic on each of them (which would take strictly longer than finding the optimal minimax set for each of them).

The following table summarizes the results of similar analysis for the other k, n combinations. In each column, one per k, n combination, the heuristics (h1 ... h5) are ordered by their mean scaled score, best to worst. Heuristics that perform significantly worse than another are italicized; those that might might conceivably be best are in bold.

k	6	6	6	12	12	12	18	18	18
n	5	25	125	5	25	125	5	25	125
best	**h2**	**h4**	**h4**	**h2**	**h4**	**h4**	**h2**	**h4**	**h4**
	h1	*h5*	**h5**	*h1*	**h5**	**h5**	*h3*	**h2**	**h5**
	h5	*h2*	*h2*	*h3*	**h2**	**h2**	*h1*	*h5*	*h2*
	h3	*h1*	*h1*	*h5*	*h3*	**h1**	*h5*	*h3*	*h1*
worst	*h4*	*h3*	*h3*	*h4*	*h1*	*h3*	*h4*	*h1*	*h3*

Since heuristic 2 performs uniformly and often significantly better than heuristic 1, one obvious conclusion is that little if anything is gained by explicitly trying to realize the lower bound; a heuristic can do at least as well by considering flipping all alternatives at each step and not only those on which A and B differ. It seems the heuristics' efficacy comes solely from the greedy search for a locally optimal winner set. Also, the best place to start the greedy search seems to vary according to k and n. It seems clear that heuristic 2 is the best of the five when n is small and that heuristic 4 emerges as best as n increases. Heuristic 3 generally performs poorly, probably because the minisum set is often close to locally optimal but globally optimal relatively rarely.

Most importantly, it turns out that all five heuristics significantly outperform the minisum set; they significantly outperform the upper bound except when $n = 125$.[3] While this result shows that it is possible to approximate minimax relatively closely in practice with a polynomial-time procedure, a true approximation ratio remains to be proved.

5 Manipulating minimax

Gibbard [6] and Satterthwaite [8] proved independently that any election system that chooses exactly one winner from a slate of more than two alternatives and satisfies a few obviously desirable assumptions (such as an absence of bias for some alternatives over others) is sometimes manipulable in practice. 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 procedures 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. This is true because a minisum election with k alternatives is exactly equivalent to k elections of two alternatives each: approve or disapprove that alternative. Since a voter's decision to approve or disapprove one alternative has absolutely no effect on whether other alternatives 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 procedure is easily shown to be manipulable. Consider the following set of *sincere* ballots:

1. **000000**
2. **000001**
3. **000010**

[3] The upper bound from section 2 is found by testing the ballots as solution sets. When n is large compared to k, especially when $n \sim 2^k$, the optimal minimax set is likely to be included in the set of ballots and thus found as the upper bound.

4. **000100**
5. **000111**

The minimax winner sets are **000001**, **000010** and **000100**, all of which have a maxscore of 2. Voter 5, however, could manipulate the result by voting the insincere ballot **111111** (resulting in the example from section 1.1), making the unique minimax set **000111**, which matches his sincerely most desired outcome. For comparison, the minisum outcome for both the sincere set of ballots and the set in which voter 5 votes insincerely is **000000**; voter 5 cannot achieve a better outcome for himself by voting insincerely, nor indeed can any other voter.

This example illustrates one general guideline to manipulating a minimax election: If there are alternatives on which a voter is in the overwhelming majority, he may be able to vote safely against the majority on those alternatives 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. Since manipulating minimax is not generally infeasibly difficult, it is less reasonable to expect a set of minimax ballots to have been sincerely voted. Perhaps an even more general approach to manipulating a minimax election is possible.

6 Conclusion and future work

The minimax procedure is a new way to elect from a set of alternatives a set of winners whose size is endogenous on the ballots. Minimax, which essentially minimizes the maximum voter disagreement with the winner set, can be seen as fairer to all voters than the minisum procedure, which minimizes the total disagreement with the winner set. Unfortunately, unlike minisum, minimax suffers from two major problems that limit its usefulness: computational complexity and manipulability. There exist polynomial-time heuristics to calculate a winner set that approximates minimax well, vastly reducing computational complexity at the expense of precision. However, at this time, answers are less clear regarding ways to modify the minimax procedure to anticipate and resist manipulation.

Questions remaining for future research include:

- Is there a better starting point for the heuristic given in section 4.1?
- Is it possible to find a better deterministic (non-randomized) approach to a minimax heuristic? Can an approximation ratio be proved?
- Might a nondeterministic approach result in a better heuristic? How does the greedy search relate to the Metropolis algorithm (simulated annealing) [7]? Would a genetic-algorithm approach be fruitful?
- How would heuristics perform under non-uniform electorate models?
- How can the heuristics presented be modified to perform well on the more general bounded-size minimax problem?
- Is there a more general approach to manipulating minimax?
- Is there a way to change minimax to lessen the effects of manipulability while retaining minimax's coalition-forming properties?
- Would using minimax in a Declared-Strategy Voting [3] framework be feasible? Could any general properties of equilibria be found?

Despite its problems, the minimax procedure holds promise for collective decision-making.

References

[1] Brams, Steven J., and Peter C. Fishburn. *Approval Voting*. Birkhäuser, Boston, Massachusetts, 1983.

[2] Brams, Steven J., D. Mark Kilgour and M. Remzi Sanver. *A Minimax Procedure for Negotiating Multilateral Treaties*. October 2003.

[3] Cranor, Lorrie Faith. *Declared-Strategy Voting: An Instrument for Group Decision-Making*. D.Sc. dissertation, Department of Computer Science, Washington University, St. Louis, Missouri, December 1996.

[4] Frances, M., and A. Litman. On Covering Problems of Codes. *Theory of Computing Systems*, 30:113–119, March 1997.

[5] Garey, Michael R., and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, New York, 1979.

[6] Gibbard, Allan. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41:587–602, 1973.

[7] Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[8] Satterthwaite, Mark A. Strategyproofness and Arrow's Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions. *Journal of Economic Theory*, 10:187–217, 1975.