Some Results on Approximating the Minimax Solution in Approval Voting

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

Evangelos Markakis*
Center for Mathematics and Computer Science (CWI)
Amsterdam, Netherlands
vangelis@cwi.nl

Aranyak Mehta
IBM Almaden Research Center
San Jose, California, U.S.A.
mehtaa@us.ibm.com

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents, Multiagent systems*

General Terms

Algorithms, Experimentation, Theory

Keywords

approval voting, minimax, approximation, heuristics

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 according to some voting protocol. Several decision making applications in AI have followed this approach including problems in collaborative filtering [10] and planning [3, 4].

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, among others, the American Mathematical Society (AMS), the Institute of Electrical and Electronics Engineers (IEEE) and the Game Theory Society (GTS).

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 [2].

Recently, a new solution concept, the minimax solution, was proposed by Brams, Kilgour and Sanver [1]. 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 [1, 2].

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

AAMAS '07 14–18 May 2007, Honolulu, Hawai'i, U.S.A.

978-81-904262-7-5 (RPS)

algorithms.

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 $\text{wt}(v)$ 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: $\text{maxscore}(v) \equiv \max_i H(v, v_i)$ 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 $k_1 \leq \text{wt}(v^*) \leq k_2$ so as to minimize $\text{maxscore}(v^*)$.

BSM includes as a special case the endogenous version, 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 variant of BSM with $k_1 = k_2 = k$, which we call Fixed-size Minimax (FSM(k)).

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.

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 [5], where the problem is stated in the context of coding theory. It follows that BSM in general is **NP**-hard. We have shown that FSM is also **NP**-hard (proofs omitted for space).

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

We will say that an algorithm for a minimization problem achieves an approximation ratio of α if for every instance of the problem the algorithm outputs a solution with cost at most α times the cost of an optimal solution. We will show that a very simple and fast algorithm achieves an approximation ratio of 3 for FSM(k), for every k, but 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 $\text{wt}(u) = k$, 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 $\text{wt}(v) < k$, then pick any $k - \text{wt}(v)$ coordinates in v that are 0 and set them to 1; if $\text{wt}(v) > k$ then pick any $\text{wt}(v) - k$ 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 this algorithm runs in time $O(n)$, independent of the number of voters.

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

We can also show that if at least one voter has weight k, then the algorithm achieves a ratio of 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$(0, n)$, the ratio is 2.

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

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 [9], in the context of computational biology. Before that, constant-factor approximations for BSM$(0, n)$ had been obtained in [6] and [7]. We believe that algorithms with such better factors may also be obtainable for FSM(k).

4. LOCAL SEARCH HEURISTICS FOR FIXED-SIZE MINIMAX

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

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. We keep making local moves until no improvement in maxscore is seen for n consecutive moves.

This heuristic framework finishes in polynomial time and has two parameters: the starting point for the binary vector c and the constant number p of candidates to replace in one local move. 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, (2) the FSM 3-approximation presented above, (3) a random set of k candidates and (4) a k-completion of a ballot with highest maxscore. (The endogenous minimax equivalent of each of these approaches was investigated by LeGrand [8].)

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

Our experimental approach was as follows: given n, m and k, some large number of simulated elections were run.

For each election, m ballots of n candidates were generated according to one of two simple distributions. The maxscores of the optimal minimax set and the winner sets found using each of the heuristics and our 3-approximation (with ballot and flipped bits chosen at random) were then calculated.

We ran 5000 simulated elections in each of seven different configurations, varying n, m, k and the ballot-generating distribution. We also ran the heuristics 5000 times each on the ballots from the 2003 Game Theory Society council election. For each heuristic in each configuration, we noted both the highest and the average realized approximation ratio (maxscore of committee found divided by optimal maxscore).

We found that the heuristics find good, if not optimal, winner sets on average. Also our 3-approximation in practice performs appreciably better than its guarantee—its ratio was less than 2 for every simulated election.

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 $p = 1$; *i.e.*, performance improved greatly with a larger local-search neighborhood. 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 and is never outperformed by them with any statistical significance.

5. MANIPULATION

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

DEFINITION 1. *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 $L_A^i(\mathbf{v})$ 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. $L_A^i(v_i, \mathbf{v^{-i}}) > L_A^i(v', \mathbf{v^{-i}})$.*

We have proved FSM's manipulability by giving an example of a voter gaining a superior outcome by voting insincerely (an analogous example for the endogenous version was provided by LeGrand [8]).

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

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.*

The above theorems give rise to the following question: What is the smallest value of α for which there exists a nonmanipulable polynomial-time approximation algorithm with ratio α?

6. FUTURE WORK

There are still many interesting directions for future research. First, we are planning to adjust our heuristics for the weighted version of the minimax solution [2]. We are also investigating variations of local search that may improve even more the performance, *e.g.*, can there be a better starting point or can we enrich the set of local moves? 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.

7. ACKNOWLEDGEMENTS

We would like to thank 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.

8. REFERENCES

[1] 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.

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

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

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

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

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

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

[8] 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.

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

[10] 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.