Use Complexity to Avoid Vote Manipulation

Minghao Lu

Computer Science Department
University of British Columbia
squallmh@cs.ubc.ca

December 27, 2006

Abstract

Aggregating the preferences of self-interested agents is a key problem for multiagent systems. A common method for doing so is to vote over the candidates. Unfortunately, the Gibbard-Satterthwaite's theorem shows that when there are more than three candidates, every nondictorial voting scheme for choosing the winning candidate is manipulable (in other words, there exist situations so that some voter would benefit from reporting his/her preference untruthfully). Recent research has had some limited success on making the voting schemes computationally hard to manipulate. However, after some simple modifications, most common voting protocols can make the manipulation NP-hard, #P-hard or even PSPACE-hard in worst-case analysis. In the paper, we will examine the computational complexity of some basic protocols and their modifications on vote manipulation. Also we will have some average-case complexity analysis.

1. Introduction

In multiagent system, it is of crucial importance to be able to aggregate the preferences of multiple agents in order to achieve their joint goal even though their individual options may vary over the alternatives (candidates). Voting is a general method of reconciling these differences. One fundamental problem encountered by all voting schemes is that of manipulation by the voters. An agent is said to vote strategically if he does not report his/her true preference in order to obtain a more beneficial outcome. For example, an agent has a preference order a>b>c, but somehow he knows that a could not win and b and c are close. The agent is better off to choose b as his favorite candidate. Thus, we can see from the example that a socially undesirable outcome may arise if some agents report insincerely.

In fact, a negative result has already been shown by Gibbard-Satterthwaite (Gibbard 1973; Satterthwaite 1975) that if there are three or more candidates, any nondictatorial voting scheme

for choosing the winning candidate is vulnerable to manipulation.
In the context of computer software agents, it makes sense to use computational complexity to scale the manipulation. Such complexity can be used as a guideline for voters to examine if manipulation is feasible or not. Hence, designing a protocol where manipulation is computationally difficult is highly desired in multiagent settings.
The rest part of the paper is organized as follows. Section 2 is notations and definitions. Section 3 introduces some common voting protocols' complexities in both constructive and destructive manipulations. In section 4 we will examine some modified version of protocols and their complexities. Section 5 is about average-case analysis. Section 6 is the conclusion.

2. Preliminaries and Definitions

Before we move on to the main part, let us make some definitions and reviews of protocols that we will investigate.

2.1 Voting protocols

Definition 1: Election

An *election* consists m candidates and n voters (possibly weighted). Denote the set of all candidates $C = \{c_1, \dots, c_m\}$ and the set of all voters $V = \{v_1, \dots, v_n\}$. The set of all permutations of C is denoted by $\prod(C)$. Most of our complexity results are based on m and n, i.e. "polynomial time" means "polynomial in m and n".

<u>*Remark1:*</u> In the case that a vote are weighted (say k), such vote is defined by the social welfare functions W as k identical unweighted votes. Whenever points are defined, the candidate with the most points wins.

- **Plurality**: A candidate receives 1 point for every voter who ranks it the first;
- **Borda**: For each voter, a candidate receives m-1 points if the voter ranks it the first, m-2 if the voter ranks it the second, and so on,…, 0 if the voter ranks it the last;
- **Copeland** (aka. Tournament): Simulate a pairwise election for each pair of candidates in turn (a candidate wins in a pairwise election if it is preferred over the other one by more than half of the voters). A candidate obtains 1 point if it wins over an opponent, 0 points if it draws, and -1 point if it loses.
- **Maxmin:** A candidate's score in a pairwise election is the number of voters that prefer it over the other. A candidate's number of points is the lowest score it gets in any pairwise election.
- **Single Transferable Vote (STV):** Winner determination process proceeds in rounds. In each round, a candidate's score is the number of voters that rank it highest among the remaining candidates; the candidate with the lowest score drops out. The last remaining candidate wins. (A vote transfers from its top remaining candidate to the next highest remaining candidate when the former drops out).
- **Cup** (aka. Binary Protocol): The winner determination process consists of $\lceil \log m \rceil$ rounds. In each round, the candidates are paired. If there is an odd number of candidates, one of them gets a bye. The candidate that wins the pairwise election over the opponent (or got a

bye) advances to the next round.

The Cup protocol requires a method for scheduling candidates. For example, such assignment can be given ex-ante (prior to votes are elicited) or we can randomize over the assignment after the votes have been submitted (Randomized Cup).

2.2 Manipulation

The computational problem of manipulation has been defined in many ways. In general, the typical definition follows that given the nonmanipulator's votes, can the manipulator(s) submit their votes in a way so that one candidate from a given set of preferred candidates wins?

Definition 2: Manipulation

We say that a voter v_i can *manipulate* a protocol P if there $\exists \pi_i' \in \prod(C)$ such that for some value of $\pi_j \in \prod(C)$, $i = 1, \dots, n$, we have

- $P(\pi_1, \dots, \pi_n) = c$;
- $P(\pi_1, \dots, \pi_{i-1}, \pi_i', \pi_{i+1}, \dots, \pi_n) = c' \neq c$
- v_i ranks c' above c.

We say that v_i manipulates P *constructively* if v_i ranks c' first (causing a candidate to win) and *destructively* (causing a candidate to not win) otherwise.

3. Complexity of Manipulating Existing Protocols

Results (Bartholdi, Tovey, & Trick 1989; Bartholdi & Orlin 1991) have already shown that high complexity depends on both the number of candidates and the number of voters being unbounded. However, Conitzer and Sandholm have shown that even with small number (constant) of candidates, high complexity can also be obtained.

3.1 Unweighted voters

Proposition (Conitzer and Sandholm): Suppose there is a constant number of candidates and evaluating the result of a particular combination of votes by a coalition is in P(polynomial time), if there is only one voter in the coalition, or if the voters are unweighted, the manipulation problem is in P. (This holds for all the different variants of the manipulation problem)

This negative result leaves us two choices:

- We may examine the problem when voters have different weights.
- We may ask whether there are reasonable settings where evaluating a manipulation is NP-hard. (uncertainty about other voters)

3.2 Voters with different weights

Definition 3: Constructive-Coalitional-Weighted-Manipulation (CCWM)

We are given a set of weighted votes S, the weights for a set of votes T which are still open, and a preferred candidate p. We are asked whether there is a way to cast the votes in T so that p wins the election.

Table 1 shows that most basic protocols (except Cup with ex-ante schedule) are with high complexity to manipulate under constructive manipulation. This is because it is easy to manipulate if the Cup protocol's schedule is known in advance. However, if we add randomization into the Cup protocol, when number of candidates is greater than 7, the Cup protocol becomes NP-complete.

Protocols	**Computational Complexity**	**Number of Candidates**
Plurality	P	Any
Borda	NP-complete	≥3
Copeland	NP-complete	≥4
Maxmin	NP-complete	≥4
STV	NP-complete	≥3
Cup(ex-ante)	P	Any
Cup(randomized)	NP-complete	≥7

Table 1: Complexity under Constructive Manipulation

Definition 4: Destructive-Coalitional-Weighted-Manipulation (DCWM)
We are given a set of weighted votes S, the weights for a set of votes T which are still open, and a preferred candidate h. We are asked whether there is a way to cast the votes in T so that h does not win the election.

From table 2, we can see that unlike constructive case, the basic protocols (except STV) under destructive manipulation are extremely easier to manipulate. Thus, STV is preferable to all other protocols under destructive manipulation.

Protocols	**Computational Complexity**	**Number of Candidates**
Plurality	P	Any
Borda	P	Any
Copeland	P	Any
Maxmin	P	Any
STV	NP-complete	≥4

Table 2: Complexity under Destructive Manipulation

3.3 Uncertainty about others' votes

Conitzer and Sandholm (2002) have shown that if the protocol is NP-hard under constructive manipulation, it is also NP-hard under uncertain votes for both weighted and unweighted voters. In fact, even evaluating a candidate's winning probability is hard when there is uncertainty about the votes (even when there is no manipulator).

4. Constructing New Protocols with High Complexity to Manipulate

We have studied the complexity of manipulating existing voting protocols. However, a further step has been made by many researchers to design new protocols that are especially hard to manipulate. We will discuss two main approaches:

- ✧ Conitzer and Sandholm (2003) proposed that adding a preround before applying the actual voting protocol can make typical protocols hard to manipulate.
- ✧ Elkind and Lipmaa (2005) described a general technique for obtaining a new manipulation-resistant protocol by combining two or more base protocols (hybrid).

4.1 Adding a preround

Rather than designing new protocol from scratch, Conitzer and Sandholm showed how to tweak existing protocols to make manipulation computationally much harder while leaving much of the original nature of the protocol intact. The type of tweak they used is to add a preround.

Definition 5: Preround

Given a protocol P, the new protocol obtained by adding a preround to it proceeds as follows:

1. The candidates are paired. If there is an odd number of candidates, one candidate gets a bye.
2. In each pairing of two candidates, the candidate losing the pairwise election between the two is eliminated. A candidate with a bye is never eliminated.
3. On the remaining candidates, P is executed to produce a winner. For this, the implicit votes over the remaining candidates are used. (For example, if a voter voted a>b>c, and b was eliminated, the voter's implicit vote is a>c.)

The preround pairing is also known as schedule and we have three different scheduling ways according to when we publish the schedule.

- Deterministic preround (DPRE): the schedule is decided and known to all voters before the votes are collected. (ex-ante)
- Randomized preround (RPRE): the schedule is drawn completely randomly after the votes are collected. (ex-post)
- Interleaved preround (IPRE): the votes are elicited incrementally and the elicitation process is interleaved with the scheduling-and-publishing process which is also done randomly (ex-interim)

Preround Type	Computational Complexity
DPRE	NP-hard
RPRE	#P-hard
IPRE	PAPACE-hard

Table 3: Manipulation Complexity after Adding a Preround

Table 3 shows that even with a very simple tweak (adding a preround), the original protocol becomes NP-hard, #P-hard or even PSPACE-hard to manipulate depending on different scheduling methods.

The advantage of this method is that it preserves some of the properties of the original protocol. (e.g.: If the original protocol is Condorcet, the tweak is also Condorcet.)
However, some other desirable features are lost during the elimination process (preround). Dropping half of the candidates is generally not a good idea and is likely to alter the outcome by the original protocol.

4.2 Hybrid voting protocols

It can be observed that the preround phase can be viewed as Binary Cup (BC) protocol. While BC itself is not hard to manipulate, the result shows that combining BC with other protocol ends up with manipulation-free schemes. Elkind and Lipmaa generalized this idea by showing that this combination is not unique to BC.

Definition 6: Hybrid Protocol

A hybrid protocol Hyb(X_k, Y) consists of two phases. Suppose that the voters' preference lists are described by the n-tuple $(\pi_1, \dots, \pi_n)$. In the first phase, the protocol executes k steps of $X(\pi_1, \dots, \pi_n)$; suppose that S is the set of candidates not eliminated in the first phase. In the second phase, the protocol applies Y to $(\pi_1|S, \dots, \pi_n|S)$, i.e., the preference lists restricted to the remaining set S of candidates.

It is obvious that we can extend this definition to more than two protocols (phases) Hyb$\left(X_{k_1}^{(1)}, X_{k_2}^{(2)}, \dots, X_{k_t}^{(t)}, Y\right)$. Also hybrid of a protocol with itself is allowed.

Hybrid Protocols	**X**	**Y**	**Complexity**
Hyb(STV_k, Y)	/	{Plurality,Borda,Maxmin,Cup,STV}	NP-hard
Hyb(X_k, STV)	{Plurality,Borda,Maxmin,Cup}	/	NP-hard
Hyb$(X_k, Plurality)$	{Borda,Maxmin}	/	NP-hard
Hyb$(X_k, Borda)$	{Borda,Maxmin}	/	NP-hard
Hyb$(Plurality_k, Y)$	/	{Borda,Maxmin,Cup,Plurality}	P

Table 4: Manipulation Complexity of Hybrid Protocols

The advantages of this method are quite obvious. Firstly, it works for a wide range of protocols. Secondly, a hybrid of two protocols preserves many of their own properties and sometime may have a best combination of both. Thirdly, some easy-to-manipulate protocols after hybridization become manipulation-resistance. This method is generally an extension of Conitzer and Sandholm's work.
On the other hand, as table 4 shows, hybridization does not always result in hard-to-manipulate protocols. (e.g. Plurality)

5. Average-Case Analysis

All of the above results have one flaw in common that they calculated computational complexity only in worst-case hardness. In other words, the results showed that it is unlikely to find an efficient algorithm that determines a beneficial manipulation in all instances where a beneficial manipulation exists. But this does not mean that such efficient manipulation does not exist. Moreover, if the beneficial manipulation does exist, it may still be the cast that most instances of the problem are easy to manipulate.

A solution to this problem would be to find a rule that is hard to manipulate in all instances. Sadly, it is very difficult to show that a certain problem is average-case complete. There have been many attempts to design a rule under which finding a beneficial manipulation is hard on average, but they have failed. However, there are several recent attempts.

5.1 New approaches on average-case analysis

One is by Procaccia and Rosenschein (2006). Basically, they use Junda distribution over the instances they analyze and show that Scoring protocols (e.g.: Borda) are *susceptible* to manipulation on average when number of candidates is constant. This is a new starting point of the area for average-case manipulation complexity analysis. But it is not complete, even the definition of Junda distribution is debatable. The authors also point out that defining natural criteria where a manipulation is hard in average-case is of great importance.

Remark2: A protocol is *susceptible* to manipulation when there is a polynomial time algorithm that can usually (average-case) manipulate it.

Another discouraging result which states that there does not exist a voting rule that is usually (average-case) hard to manipulate is by Conitzer and Sandholm (2006). Unlike Procaccia and Rosenschein's claim, this result does not require any restriction on the voting rule, number of candidates or distribution over instances. The result generally shows that a voting rule is not usually hard to manipulate under some properties (such as weak monotonicity). The authors also give approaches for potentially circumventing this impossibility result.

6. Conclusion

In multiagent systems, different agents have various preferences over the candidates. It is frequently desired to aggregate these preferences so as to achieve a socially beneficial result. Unfortunately, Gibbard-Satterthwaite theorem shows that all voting protocols are manipulable when there are more than two candidates. To circumvent this problem, recent research has been seeking whether there is a way to make beneficial manipulation computationally hard. As we have showed in the paper, many protocols can be NP-hard to manipulate with the properties such that the voters are differently weighted, adding randomness, or uncertainties about others' votes. Moreover, after some simple modifications (adding a preround or hybridization) to the original protocols, the resulting protocols can also achieve NP-hardness (or even PAPACE-hardness). However, these results are all in worst-case sense. In the last section, we analyzed the average-case complexity. Surprisingly, Conitzer and Sandholm's result shows that it is in fact impossible to design a protocol which is simultaneously required to satisfy another property: a large portion of the instances are both weakly monotone and allow the manipulators to make either of exactly two candidates win.

Some future work can be made on how to circumventing this impossibility result and how to define natural criteria where a manipulation is hard in average-case since the regular average-case analysis is not suitable here.

References

1) John J. Bartholdi, III and James B. Orlin, "*Single Transferable Vote Resists Strategic Voting*" Social Choice and Welfare p.341-354, Nov 1990, revised April 2003
2) Vincent Conitzer and Tumoas Sondheim, *"Complexity of Manipulating Elections with Few Candidates".* In Proceedings of the National Conference on Artificial Intelligence (AAAI), p314-319, 2002.
3) Vincent Conitzer and Tuomas Sondheim, *"Universal Voting Protocol Tweaks to Make Manipulation Hard"*. In Proceedings of the National Conference on Artificial Intelligence (AAAI), p.201-214, 2003.
4) Edith Elkind and Helger Lipmaa, *"Small Coalitions Cannot Manipulate Voting",* Financial Cryptography and Data Security 2005, volume 3570 of Lecture Notes in Computer Science, p.285-297, 2005.
5) Edith Elkind and Helger Lipmaa, *"Hybrid Voting Protocols and Hardness of Manipulation*" The 16th Annual International Symposium on Algorithms and Computation, ISAAC 2005, volume 3827 of Lecture Notes in Computer Science, p206-215, Sanya, Hainan, China, Dec 2005
6) Ariel D. Procaccia and Jeffrey S. Rosenschein, "*Junta Distributions and the Average-Case Complexity of Manipulating Elections*", The 5th International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, May 2006.
7) Vincent Contizer and Tuomas Sandholm, "*Nonexistence of Voting Rules That Are Usually Hard to Manipulate*", AAAI, 2006