Controlling elections by replacing candidates for plurality and veto: theoretical and experimental results

Andrea Loreggia
University of Padova
loreggia@math.unipd.it

Nina Narodytska
University of Toronto and
University of New South Wales
ninan@cs.toronto.edu

Francesca Rossi
University of Padova
frossi@math.unipd.it

K. Brent Venable
Tulane University and IHMC
kvenabl@tulane.edu

Toby Walsh
NICTA and UNSW
toby.walsh@nicta.com.au

ABSTRACT

We consider elections where the chair may attempt to influence the result by replacing candidates with the intention to make a specific candidate lose (destructive control). We call this form of control "replacement control" and we study its computational complexity. We focus in particular on plurality and veto, for which we prove that destructive control via replacing candidates is computationally difficult. To get more insight into the practical complexity of this problem, we also perform an extensive experimental study.

Categories and Subject Descriptors

I.2.11 [[**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Economics

Keywords

Social Choice, Voting, Control

1. INTRODUCTION

The result of an election can be influenced in many ways. For instance, voters may submit insincere preferences or the chair may introduce new candidates or choose the voting rule. We focus here on control by the chair.

Control may be constructive when the chair's goal is for a certain candidate to win, or destructive when the chair's goal is to prevent a candidate winning. One action that the chair can take is adding or deleting candidates or votes. We consider a specific form of combining the basic control actions, called *replacement control*, where we replace some candidates (or votes) with the same number of other candidates (or votes). This can be seen as a combination of deletion and addition of candidates (or votes) in the same quantity.

Replacement of candidates or voters in order to make some candidate win or lose can be useful in many election settings, for example when the chair feels that certain candidates or voters are impeding the desired result. A real-life example can be found in the current campaign for the Indian elections of the House of the People (Lok Sabha): the socialist (Samajwadi) party replaced almost half of its candidates after an internal feedback that suggested that the party would otherwise have won very few seats in the Lok Sabha. [1]

We consider a voting rule to be vulnerable (resp. resistant) to a form of control if it is polynomial (resp. NP-hard) to check whether that control can be achieved. We prove that plurality and veto are both resistant to destructive control via replacing candidates.

We then run an extensive experimental analysis of the practical complexity of this problem, to check whether such voting rules are really difficult to control in practice. To do that, we use real datasets from the preflib repository [8] and we consider k-approval. Our experimental study shows that plurality is more resistant to this form of control than other versions of k-approval. Moreover, we compare the control power of replacing candidates to the power of just adding or deleting them, showing that replacing candidates add significant power to the chair of the election, with respect to to the single control action of adding or deleting candidates.

2. BACKGROUND

An election E is a pair (C, V) where C is a set of m candidates and V is a collection of n votes (linear orders over C). A voting rule R takes an election and returns the winning candidate from C. Besides the voters and the candidates, there is also another agent, the chair who can influence, for example, which candidates and voters participate.

Positional scoring rules give to each candidate points based on their ranked position in each vote. The sum of the points gives the total score of the candidate. The candidate with

[1] http://archive.indianexpress.com/news/samajwadi-party-to-replace-24-ls-candidates/1184010/, http://indiatoday.intoday.in/story/mulayam-singh-yadav-changes-a-dozen-samajwadi-party-candidates-for-2014-polls/1/326837.html

Appears at: 1st Workshop on Exploring Beyond the Worst Case in Computational Social Choice. Held as part of the 13th International Conference on Autonomous Agents and Multiagent Systems. May 6th, 2014. Paris, France.

the highest score is the winner. Scoring rules differ in the way points are allocated to candidates. This difference is expressed by the scoring vector, which denotes the score given by each vote to each candidate according to its position. We consider plurality (where the scoring vector is $v =< 1, 0, \ldots, 0 >$), veto ($v =< 1, 1, \ldots, 1, 0 >$), and k-approval ($v =< 1, 1, \ldots, 1, 0, 0, \ldots, 0 >$, where there are k 1's).

There are many ways that the outcome of an election can be influenced. Voters may submit insincere preferences, whilst the chair may add or delete candidates or votes. While there are few situations in which voting rule cannot be manipulated or controlled, it could be computationally complex to understand whether a form of manipulation or control is possible, and how to do it. This may protect the election against such strategic actions. In this paper we focus on control actions, and we say that a voting rule is *immune* to a type of control if the result cannot be affected by that type of control, otherwise we say that it is *susceptible* to that type of control. If it is susceptible, we say that it is *resistant* to a control action if the problem that the chair has to deal with is NP-hard, otherwise if the problem is in P we say that it is *vulnerable*. This line of thought originated by the seminal work of Bartholdi, Tovey and Trick [1], which was the first one to consider computational complexity as a shield that the system can use against control actions.

Control actions can be *constructive* or *destructive*, depending on whether the chair aims at making a certain candidate win or lose. The forms of control considered here are the addition or deletion of candidates and/or votes. The computational complexity of these forms of control have been studied in the literature, with results for several voting rules [4, 5, 3, 2, 9]. We will use the usual acronym DC for Destructive Control, AC (for Adding Candidates), DC (for Deleting Candidates) as well as their combinations.

3. REPLACEMENT CONTROL

Replacement control can be seen as the combination of the addition and deletion of either votes or candidates in equal amount. That is, the chair can *replace* some candidates. We use RC for Replacing Candidates. These will be combined with destructive control (DC). Formally, we will study the following problem.

Name: DCRC (Destructive Control via Replacing Candidates)

Given: a collection V of votes over $C_1 \cup C_2$ (with C_1 and C_2 disjoint), a distinguished candidate $p \in C_1$, and $r \in \mathbb{Z}_{\geq 0}$

Question (DCRC): is there a subset $A \subseteq C_2$ and a subset $D \subseteq C_1$ such that $|A| = |D| \leq r$ and $p \in (C_1 \setminus D)$ is not the winner of the election $E = ((C_1 \setminus D) \cup A, V)$?

We write $DCY(C, A, V, p, r)$ to denote an instance of the problem with $Y \in \{AC, DC, RC\}$, where C is a set of candidates. A is another set of candidates and V is the collection of votes over $C \cup A$. Moreover, $p \in C$ is a distinguished candidate and r is the budget. Informally, A is the set of candidates or votes that the chair may add to the election, while candidates or votes to be deleted comes from C or V.

4. PLURALITY AND VETO RESISTANCE TO DCRC

For the following result, we use the notion of *Insensitive to Bottom-ranked Candidates (IBC)* defined in [7]. A voting rule is IBC if it elects the same winner after unanimously adding at the bottom of the profile some candidates. In other words, we say that a voting rule is IBC if, given a profile P over a set of alternatives $C = \{c_1, \ldots, c_m\}$, and P' another profile over $C \cup \{c_{m+1}\}$ that is obtained by adding c_{m+1} as the least preferred candidate of every vote in P, then the winner in P is the same as in P'. This property allows us to prove the resistance to DCRC of a voting rule, if it is resistant to DCDC, by adding useless candidates at the bottom.

THEOREM 1. *Every voting rule that is IBC and resistant to DCAC or DCDC is also resistant to DCRC.*

PROOF. From any instance $I = DCDC(C, \emptyset, V, p, r)$ we can define an instance $I' = DCRC(C, A, V', p, r)$, where $|A| = r$, and the preferences over C in V' are the same as in V, while preferences in V' rank all candidates in A unanimously at the bottom, all in the same order. We claim that there exists a solution to I if and only if there exists a solution to I'. Suppose that there exists a solution $D \subseteq C$ to I. This means that $|D| = r$ and p is not a winner in the election $E = (C \setminus D, V)$. Then p is not a winner in the election $E' = ((C \setminus D) \cup A, V')$, thus giving a solution (D, A) to I'. The addition of candidates A does not change the winner because of IBC. For the reverse, suppose that there exists a solution (D, A') to I'. This means that $A' \subseteq A$ and p is not a winner in the election $E' = ((C \cup A') \setminus D, V')$. Then, p is not a winner also in the election $E = (C \setminus D, V')$, since the elimination of A does not influence the winner by IBC. This gives us a solution D to the instance I. We can give a similar proof reducing from any instance $I = DCAC(C, A, V, p, r)$. □

We are now ready to prove that plurality and veto are resistant to replacing candidates.

THEOREM 2. *Plurality is resistant to DCRC.*

PROOF. It follows from the fact that plurality is resistant to DCDC [5] and from the application of Theorem 1. □

The situation for veto is the same as for plurality: resistant to the replacement of candidates.

THEOREM 3. *Veto is resistant to DCRC.*

PROOF. We prove the NP-hardness of DCRC by reduction from the hitting set problem. Let (B, S, r) be an instance of the hitting set problem: $B = \{b_1, \ldots, b_n\}$, $r \leq n$ and $S = \{S_1, \ldots, S_m\}$ subsets of B. The question is if there exists a subset $B' \subseteq B$ with $|B'| \leq r$ such that B' contains at least one element from each subset of S. Given an instance $I = (B, S, r)$, we show how to define an instance $I' = DCRC(C, A, V, p, r)$ of the DCRC problem such that I has a solution B' if and only if I' has a solution (D, A'), which means that p loses the election $((C \setminus D) \cup A', V)$. In I', C contains the following candidates: w, p, and d_j, for $j = 1, \ldots, r$. We call D the set of candidates d_j. $A = \{a_1, \ldots, a_n\}$ is a set of candidates the chair could use to replace some candidates in C, each candidate $a_i \in A$ correspond to an element $b_i \in B$.

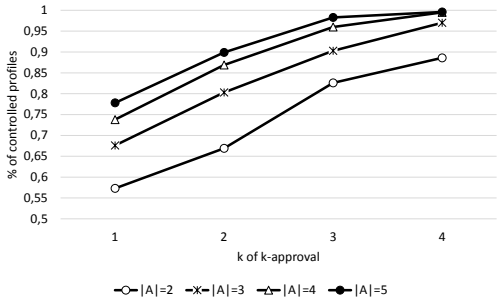

Figure 1: Sushi dataset: fraction of profiles (over 1000) with successful DCRC.

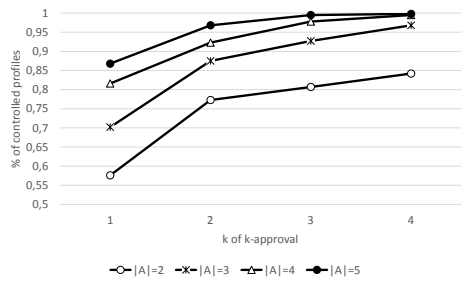

Figure 2: T-shirt dataset: percentage of profiles (over 1000) with successful DCRC.

The collection V of votes is as follows:

$$\begin{aligned}
\text{1 vote} &: D \succ A \succ w \succ p \\
\text{2 votes for each } S_i \in S &: p \succ D \succ A \setminus S_i \succ w \succ S_i \\
\text{2 votes for each } b_i \in B &: p \succ D \succ w \succ A \setminus \{a_i\} \succ a_i \\
\text{2 votes for each } d_i \in D &: p \succ w \succ A \succ D \setminus \{d_i\} \succ d_i \\
\text{2 votes for each } a_j \in A &: p \succ w \succ D \succ A \setminus \{a_j\} \succ a_j
\end{aligned}$$

We use the notation D to mean the linear order of the candidates in D, that is, $d_1, \ldots, d_k$. The same is for A. Before the replacement the number of vetoes per candidate are: $veto(p) = 1$, $veto(w) = 2m + 2n$, $veto(d_j) \geq 2$, so p is the winner of the election (C, V). We claim that a solution to I exists if and only if there exists a solution to I'. Suppose that B' is a solution to I. Then, in I', we add the candidates A corresponding to the elements in B' and we delete the candidates in D. This gets the following number of vetoes per candidate: $veto(p) = 1$, $veto(w) = 0$, $veto(a_j) \geq 4$, so p loses the election. On the other hand, suppose p loses the election by replacing at most r candidates and there exists a solution (D, A') to I'. None of the candidates added/deleted changes the vetoes for p, because of the structure of the profile, while the number of vetoes of w is decreased: for each candidate $a_i \in A$ added to the election, the number of vetoes of w decreases by 2 if $b_i \in S_i$ and no previously added candidate corresponded to another $b_j \in S_i$. Let us consider the set B' of all b_i corresponding to added candidates a_i. We have that $|B'| \leq r$ and contains at least one element from each subset of S. Thus it is a hitting set and therefore B' is a solution to I. The candidate w is the only one which can dethrone p and this is possible if in the second group of voters there is always at least 1 candidate ranked lower than w. This again is only possible if there is a hitting set. □

5. EMPIRICAL EVALUATION

To understand better the theoretical results showing the hardness of the DCRC control problem, we performed an empirical evaluation on real-world datasets. Besides plurality and veto, in this experimental analysis we also consider k-approval for values of k that are different from 1 and $m-1$. We also compare DCRC with single control actions which just add or delete candidates (DCAC and DCDC).

We consider profiles coming from real world data sets. In particular, we use three datasets from the prelib repository (www.preflib.org) [8]:

- the AGH Course Selection ED00009, which contains the preferences of some university students over a set of courses [10];
- the T-Shirt ED00012 dataset, which contains the preferences of some NICTA employees over some tshirt templates;
- the sushi dataset ED00014, which contains the preferences of 5000 people on various kinds of sushi [6].

For each data set, we generate profiles of 1000 votes by randomly selecting preference rankings from the dataset. We also assume that $r = |A|$.

The first thing we show is the percentage of profiles where DCRC is able to change the winner. Figure 1 reports the test run using the sushi data set, with 10 voters, $|C| = 5$ and $2 \leq |A| \leq 5$. The x axis has the value of k in k-approval, which varies from 1 to 4. The four curves correspond to different sizes of set A. Clearly, the larger k and A, the more controllable the profile is, because there could be more harmful combinations of candidate replacements. The same behaviour can be observed in figure 2 that reports the test run using the t-shirt data set, with 25 voters, $|C| = 5$ and $2 \leq |A| \leq 5$. The x axis has the value of k in k-approval, which varies from 1 to 4. Even if the data is different we can observe the same trend in both chart: while veto seems to be controllable most of the times, plurality shows some resistance to it.

We then consider the actual difficulty for changing the winner, or for discovering that it cannot be changed, by considering a deterministic algorithm that checks all possible combinations of candidates to be added, and an equal number of candidates to be deleted, starting from combinations with budget (number of replacements) 1 and going up to the maximum size. A lexicographic ordering over candidates is used to decide which delete/add combinations to try first with the same budget size.

Figure 3 shows the average percentage of combinations tested, over all possible add/delete combinations, when using 1000 profiles from the sushi dataset, with 10 voters, $|C| = 5$ and $2 \leq |A| \leq 5$. The x axis has the value of k in k-approval, which varies from 1 to 4. Figure 4 shows the same information but against the t-shirt dataset with 25

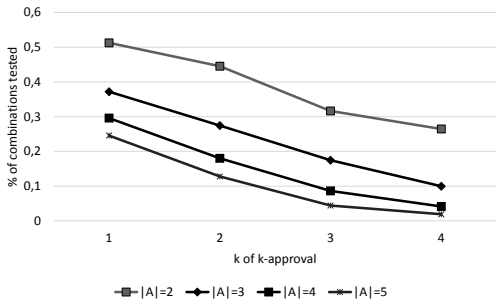

Figure 3: Deterministic algorithm on sushi dataset: Average percentage of tried add/delete combinations.

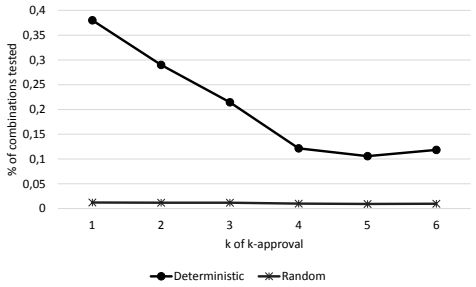

Figure 5: Deterministic and non-deterministic algorithm on sushi dataset: comparison.

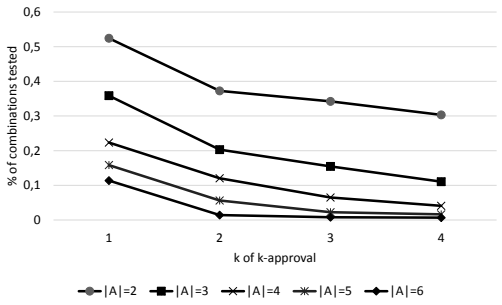

Figure 4: Deterministic algorithm on T-shirt dataset: Average percentage of tried add/delete combinations.

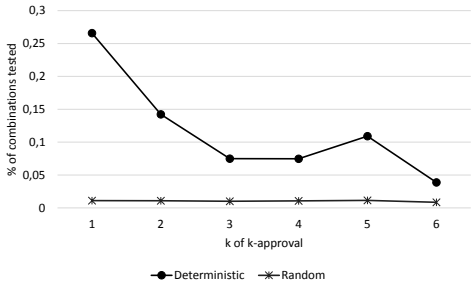

Figure 6: Deterministic and non-deterministic algorithm on t-shirt dataset: comparison.

votes. We are interested in the main trend of these charts and not in the small differences that they can report, because these small differences are connected to the structure of the preferences in the dataset. What is really interesting is once again that the larger are k and $|A|$, the smaller is the computational effort of this algorithm.

We also considered a non-deterministic algorithm which the chair of the election could use to change the winner by replacing candidates. Such an algorithm consists of picking an add/delete combination randomly (over all possible combinations), and checking whether the winner changes. From the experimental data, we count the percentage of profiles where the winner changes (see Fig. 1) and we use this as the probability of success of this approach. If p is the probability that picking one profile is enough to change the winner, it is easy to see that $1/p$ is the expected number of profiles to be picked up before changing the winner. We therefore show this $1/p$ number as a measure of how many combinations should be tested by this non-deterministic algorithm before changing the result (or discovering that it cannot change).

Figure 5 compares the difficulty of the DCRC problem as measured in Fig.2 to this measure of the difficult of DCRC via the non-deterministic algorithm. We used the sushi dataset, with 10 voters, $|C| = 7$ and $|A| = 3$. The x axis has the value of k in k-approval, which varies from 1 to 6, while the y axis shows the percentage of add/delete combinations that the algorithm tries before stopping. Figure 6 shows the data collected using the t-shirt data set, with 25 voters, $|C| = 7$ and $|A| = 3$. It can be seen that the non-deterministic algorithm appears to be more efficient, since it always needs to try a smaller number of combinations.

We also compared the power of replacing candidates with respect to just adding or deleting candidates. We consider the profiles where the winner changes using RC, and we count in how many of these profiles

- the winner changes using AC but does not change using DC (denoted by "AC only");
- the winner changes using DC but does not change using AC (denoted by "DC only");
- the winner changes using either DC or AC (denoted by "AC only and DC only");
- the winner changes only using RC (denoted by "RC only").

Notice that "AC only" and "DC only" do not add up to "AC only and DC only" because all these categories represent disjoint sets of profiles.

Figure 7 shows the percentage of profiles where the winner changes using RC on the sushi dataset. We use a stacked

bar histogram that report the percentage of profiles where the winner change using RC only, AC only, DC only, or AC only and DC only, for plurality and veto. We used the sushi dataset, with 10 voters, $|C| = 5$ and $|A|$ varies over the x axis from 1 to 4.

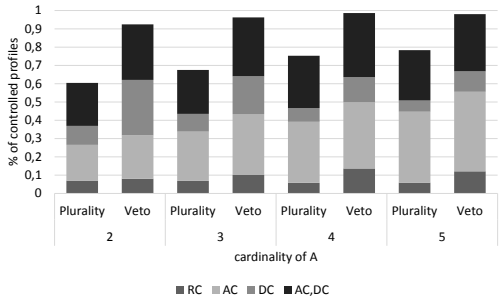

Figure 7: Deterministic algorithm on sushi dataset: RC compared to AC, DC, and AC+DC.

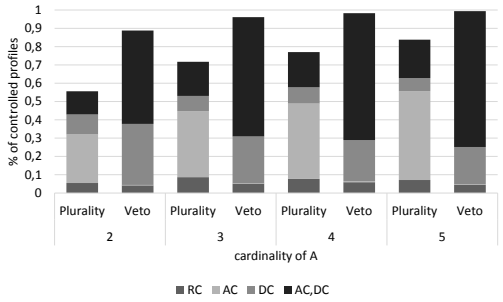

Figure 8: Deterministic algorithm on t-shirt dataset: RC compared to AC, DC, and AC+DC.

The number of profiles where the winner changes using AC but does not change using replacement control is on average 0.3%. Thus, it can be seen that RC improves the vulnerability of the voting rule since the number of controllable profiles increases by about 9%, this is a significant increase in controllability compared to AC or DC alone that is not reported in this chart and which is around 0.3%, thus making the voting rule much more vulnerable to this kind of control action.

Figure 8 shows data about the same experiment but using the t-shirt dataset. What is interesting in this chart is that the structure of the preferences made veto almost resistant to AC only but the voting rule shows the same trend about the vulnerability to RC. Once again RC improves the vulnerability of the voting rule since the number of controllable profiles increases by about 7%, this is a significant increase in controllability compared to AC or DC alone that is not reported in this chart and which is around 0.2%, thus making the voting rule much more vulnerable to this kind of control action.

Furthermore, we do not report the data collected using the AGH course selection dataset, because they show the same trends as the ones of the other datasets.

6. CONCLUSIONS

We have studied the computational complexity of replacement control, where the chair tries to influence an election by replacing candidates or votes. In particular, in this paper we focused on destructive control via replacing voters, showing that this problem is NP-complete in both plurality and veto.

We also performed an extensive experimental work, using real-world data sets, to test if k-approval is really difficult in practice to control via replacing candidates. Our experiments show that plurality is more resistant to DCRC than other versions of k-approval. Also, a non-deterministic algorithm seems to be the most convenient for the chair to control the election. Finally, RC is significantly more powerful than just AC or DC alone in terms of giving the chair control over the election.

7. REFERENCES

[1] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. How hard is it to control an election. *Mathematical and Computer Modeling*, pages 27–40, 1992.

[2] G. Erdélyi, L. Piras, and J. Rothe. Bucklin voting is broadly resistant to control. *CoRR*, abs/1005.4115, 2010.

[3] G. Erdélyi, L. Piras, and J. Rothe. Control complexity in fallback voting. *CoRR*, abs/1004.3398, 2010.

[4] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Multimode control attacks on elections. *JAIR*, pages 305–351, 2011.

[5] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171:255–285, 2005.

[6] T. Kamishima, H. Kazawa, and S. Akaho. A survey and empirical comparison of object ranking methods. In J. Fürnkranz and E. Hüllermeier, editors, *Preference Learning*, pages 181–201. Springer-Verlag, 2010.

[7] J. Lang, N. Maudet, and M. Polukarov. New results on equilibria in strategic candidacy. *Proceedings of the 6th International Symposium on Algorithmic Game Theory (2013)*, 2013.

[8] N. Mattei and T. Walsh. Preflib: A library for preferences http: //www.preflib.org. In *ADT*, pages 259–270, 2013.

[9] C. Menton. Normalized range voting broadly resists control. *CoRR*, abs/1005.5698, 2010.

[10] P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation is easy in practice. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 399–406, Richland, SC, 2013. International Foundation for Autonomous Agents and Multiagent Systems.