Controlling Elections by Replacing Candidates: Theoretical and Experimental Results

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

Nina Narodytska
University of Toronto and UNSW
email: ninan@cs.toronto.edu

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

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

Toby Walsh
NICTA and UNSW
email: 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, and Borda for which we prove that destructive control via replacing candidates is computationally easy. To get more insight into the practical complexity of this problem, we also perform an extensive experimental study. This study shows that the theoretical computational complexity results are often not reflecting the practical difficulty of controlling elections by replacing candidates.

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.

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, while Borda is vulnerable to it.

We then run an extensive experimental analysis to assess the practical complexity of this problem and check whether such voting rules are really difficult to control in practice. To do that, we use real datasets from the preflib repository (Mattei and Walsh 2013) and we consider k-approval and Borda. Our experimental study shows that Plurality is more resistant to this form of control than other versions of k-approval. Borda, instead, is rather vulnerable to control by replacing candidates. Moreover, we compare the control power of replacing candidates to the power of just adding of deleting them, showing that replacing candidates add significant power to the chair of the election, with respect to to the single control actions of adding or deleting candidates.

We also observe that, even if theoretically Veto is resistant and Borda is vulnerable, they both show a different behaviour in the empirical evaluation, due to the low likelihood of the conditions that make the system vulnerable or resistant to replacement control. These results suggest that the study of computational complexity in the worst case is not enough to ensure a significant protection to the system, as suggested in (Bartholdi, Tovey, and Trick 1992; Faliszewski and Procaccia 2010; Faliszewski et al. 2009; Faliszewski, Hemaspaandra, and Hemaspaandra 2011; 2010; Walsh 2010).

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 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), and Borda ($v =< m-1, m-2, \ldots, 0 >$).

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 deciding how to perform that action is NP-hard, otherwise we say that it is *vulnerable*.

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 (Faliszewski, Hemaspaandra, and Hemaspaandra 2011; Hemaspaandra, Hemaspaandra, and Rothe 2005; Erdélyi, Piras, and Rothe 2010b; 2010a; Menton 2010). Control actions can be *constructive* or *destructive*, depending on whether the chair aims at making a certain candidate win or lose. We will use the usual acronym DC for Destructive Control, AC (for Adding Candidates), DC (for Deleting Candidates) as well as their combinations.

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).

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 $r \in \mathbb{Z}_+$

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.

Relationship with Adding/Deleting Candidates

There is no connection between the computational complexity of replacement control and that of the single control actions of adding or deleting candidates.

Theorem 1. *There exist voting rules such that:*

1. *destructive control by adding candidates is in P, by deleting is NP-hard, and replacing is in P;*
2. *destructive control by adding candidates is in NP-hard, by deleting is NP-hard, and replacing is in P;*
3. *destructive control by adding candidates is in NP-hard, by deleting is in P, and replacing is in P;*
4. *destructive control by adding candidates is in P, by deleting is NP-hard, and replacing is NP-hard;*
5. *destructive control by adding candidates is in P, by deleting is in P, and replacing is NP-hard;*
6. *destructive control by adding candidates is in NP-hard, by deleting is in P, and replacing is NP-hard.*

Proof. The NP-hardness results of the single destructive control action of adding and deleting candidates are showed in (Faliszewski et al. 2009). Given an election $E = (C, V)$, where C is a set of m qualified alternatives and V is a collection of n voters, fix $r = \frac{m}{n}$. There are also a set A of candidates that can be added to the election. Anytime we can compute the value $r' = \frac{m'}{n}$, where m' is the number of candidates in the election the chair is trying to control. With respect to the value of r:

1. consider a voting rule that works like Borda when $r' > r$, it works like plurality when the value of $r' < r$ and works like Borda when $r' = r$. When $r' < r$ the chair is exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant to DCDC (Faliszewski et al. 2009), while Borda is vulnerable to DCAC from corollary 3. When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Corollary 1 that Borda is vulnerable to DCRC;
2. consider a voting rule that works like plurality when $r' > r$ and $r' < r$, it works like Borda when $r' = r$. When $r' < r$ the chair is exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant to DCDC and to DCAC (Faliszewski et al. 2009). When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Corollary 1 that Borda is vulnerable to DCRC;
3. consider a voting rule that works like plurality when $r' > r$, it works like Borda when the value of $r' < r$ and works like Borda when $r' = r$. When $r' < r$ the chair is exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant DCAC (Faliszewski et al. 2009), while Borda is vulnerable to DCDC from corollary 4. When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Corollary 1 that Borda is vulnerable to DCRC;
4. consider a voting rule that works like Borda when $r' > r$, it works like plurality when the value of $r' < r$ and works like plurality when $r' = r$. When $r' < r$ the chair is

exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant to DCDC (Faliszewski et al. 2009), while Borda is vulnerable to DCAC from Theorem 3. When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Theorem 2 that plurality is resistant to DCRC;

5. consider a voting rule that works like plurality when $r' > r$ and $r' < r$, it works like plurality when $r' = r$. When $r' < r$ the chair is exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant to DCDC and to DCAC (Faliszewski et al. 2009). When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Theorem 2 that plurality is resistant to DCRC;
6. consider a voting rule that works like plurality when $r' > r$, it works like Borda when the value of $r' < r$ and works like plurality when $r' = r$. When $r' < r$ the chair is exploiting some deleting control actions against the set of candidates, while she is using some adding control actions when $r' > r$. We know that plurality is resistant DCAC (Faliszewski et al. 2009), while Borda is vulnerable to DCDC from Theorem 4. When the value of $r' = r$ the chair is replacing some candidates or she is doing nothing. Then, we showed with Theorem 2 that plurality is resistant to DCRC.

□

Plurality, Veto and Borda Results

In this section we report some results that are useful for the empirical evaluation we do in this paper. In particular, we are interested in the theoretical results about the computational complexity of for DCAC, DCDC and DCRC for Plurality, Veto, and Borda. The results about the single control actions of DCAC and DCDC for Plurality and Veto can be found in (Faliszewski et al. 2009), while we formally report and analyze the two algorithms that prove the vulnerability of Borda to DCAC and DCDC.

Theorem 2. *Plurality and Veto are resistant to DCRC.*

Proof. (Sketch) We can prove the resistance of Plurality building a particular profile that uses the resistance to DCDC (Hemaspaandra, Hemaspaandra, and Rothe 2005). We can prove the resistance of Veto to DCRC by reduction from the hitting set problem. These proof are omitted due to lack of space. □

Surprisingly, DCRC is polynomial for Borda, because the differences between two consecutive scores in the Borda's scoring vector are identical. We prove it by firstly reporting and analyzing the algorithms proving that Borda is vulnerable to adding or deleting candidates for destructive control.

Theorem 3. *Borda is vulnerable to DCAC.*

Algorithm 1 Destructive Control Adding Candidates

```
1: procedure DCACBORDA(C, A, V, p, r)
2:     w ←Compute-Borda-winner(C, V)
3:     if p ≠ w then
4:         return ∅
5:     for all x ∈ C ∪ A do
6:         for all c_i ∈ ((C ∪ A) \ {x}) do
7:             dist(x, c_i) ← 0
8:             for all v_l ∈ V do
9:                 dist(x, c_i) ← sign(sign(pos(v_l, x) − pos(v_l, c_i)) −
   sign(pos(v_l, w) − pos(v_l, c_i))) + dist(x, c_i)
10:    for all x ∈ C ∪ A do
11:        A' = ∅; copyA ← A
12:        j ← 0
13:        if x ∈ A then
14:            A' ← A' ∪ {x}; j ← 1
15:        while j < r do
16:            a ← arg max_{a_i ∈ copyA} dist(x, a_i)
17:            A' ← A' ∪ {a};
18:            w ←Compute-Borda-winner(C ∪ A', V)
19:            copyA ← copyA \ {a}; j ← j + 1
20:            if w ≠ p then
21:                return A'
22:    return null
```

Proof. Algorithm 1 is polynomial and proves the statement of this theorem. The rough idea of the algorithm is that, given an instance DCAC(C,A,V,p,r), the algorithm fixes another candidate x and it verifies if it can defeat p. The chair can make p lose the election by adding candidates in C that are ranked between x and p. For instance, candidates $a_i \in A$ give 1 point to x and 0 point to p for each voter that has preferences like in the following example:

$$\ldots \succ x \succ \ldots \succ a_i \succ \ldots \succ p \succ \ldots$$

where dots means that the other candidates are ranked in any arbitrary way. Starting from this observation, the chair can fix a candidate $x \in (C \cup A)$ one at a time and it can add candidates that maximize the score of x with respect to the score of p and the budget r.

Given two disjoint sets C, A of alternatives, a collection V of preferences over $C \cup A$, a distinguished candidate $p \in C$ and $r \in \mathbb{Z}_+$, the algorithm answers to the question if it is possible to make p lose the election by adding at most r candidates. If the answer is "yes", it also returns a subset $A' \subseteq A$ which is a solution to the problem, otherwise it returns null. Let $m = |C \cup A|$ and $n = |V|$. First, it controls if p is the current winner. If it is not the case, the chair does not need to do anything to make p lose the election. Line 2 to 4 are computed in $O(nm)$, since winner determination is polynomial for Borda. If p is the current winner, then the algorithm computes the support that each candidate gives to the other ones. This is done in lines 5 to 9 in $O(nm^2)$. From line 10 to 21 the algorithm fixes a candidate x and tries to add the candidates that give maximal support to it. After each addition, it computes the winner. If the winner changes, then a solution to the problem is found and this solution is returned (line 21), otherwise the algorithm goes on. These operations are done for each candidate until the

budget is exhausted or the winner changes.

The algorithm is sound and complete. It always terminates and returns a solution of the problem instance, if it exists. In particular it always returns the solution that firstly change the winner and makes some other candidate $x \in C$ win the election. If the algorithm stops without returning a solution, then a solution to the instance of the problem does not exist. The computational complexity of the algorithm is $O(knm^2)$. □

Theorem 4. *Borda is vulnerable to DCDC.*

Proof. The polynomial Algorithm 2 that shows the statement of this theorem is very similar to algorithm 1, except that for the DCDC problem it fixes a candidate x and tries to maximize its score with respect to the score of p and the budget r by the deletion of candidates in C. □

Algorithm 2 Destructive Control Deleting Candidates

```
procedure DCDCBORDA(C, A, V, p, r)
    w ←Compute-Borda-winner(C, V)
    if p ≠ w then
        return (∅, ∅)
    for all x ∈ C do
        for all c_i ∈ (C \ {x}) do
            dist(x, c_i) ← 0
            for all v_l ∈ V do
                dist(x, c_i) ← sign(sign(pos(v_l, x) − pos(v_l, c_i)) −
sign(pos(v_l, w) − pos(v_l, c_i))) + dist(x, c_i)
    for all x ∈ C do
        D = ∅; copyC ← C;
        j ← 0
        while j < r do
            c ← arg min_{c_i ∈ copyC} dist(x, c_i);
            D ← D ∪ {c}
            w ←Compute-Borda-winner((C \ D), V)
            copyC ← copyC \ {c}; j ← j + 1
            if w ≠ p then
                return D
    return null
```

Corollary 1. *Borda is vulnerable to DCRC.*

Proof. The algorithm can be easily derive as a combination of the two algorithms used in the Theorem 3,4. □

Empirical Evaluation

To better understand the theoretical results showing the hardness or easiness 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 $m - 1$ as this naturally interpolates between Plurality (1-approval) and Veto (m-1-approval). We also run some experiments using Borda, to check whether in practice it is easy, as the theoretical results say.

We also compare DCRC with single control actions which just add or delete candidates (DCAC and DCDC).

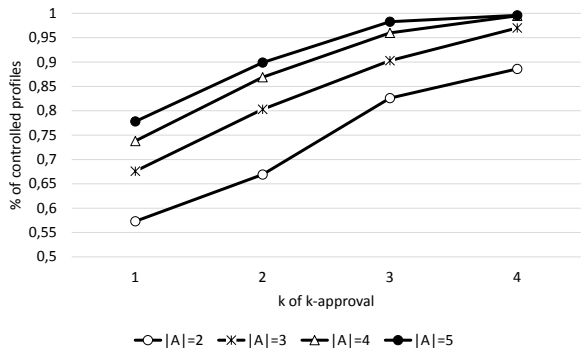


Figure 1: Percentage of profiles (over 1000) with successful DCRC.

We consider profiles coming from real world data sets. In particular, we use three datasets from the prelib repository (www.preflib.org) (Mattei and Walsh 2013):

- the AGH Course Selection ED00009, which contains the preferences of some university students over a set of courses (Skowron, Faliszewski, and Slinko 2013);
- 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 (Kamishima, Kazawa, and Akaho 2010).

For each data set, we generate profiles of 1000 votes by randomly selecting preference rankings from the dataset.

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 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 decided which delete/add combinations to try first with the same budget size.

Figure 2 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. We are interested in the

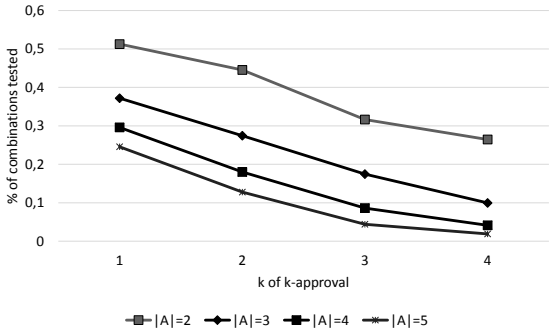


Figure 2: Deterministic algorithm: Average percentage of tried add/delete combinations.

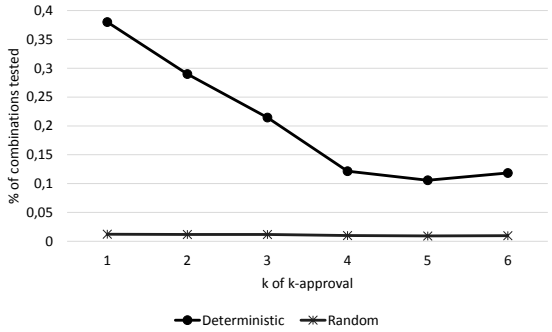
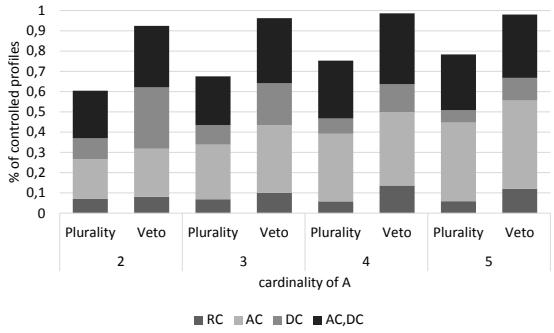


Figure 3: Deterministic and non-deterministic algorithm: comparison.

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 3 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.



t]

Figure 4: Deterministic algorithm: RC compared to AC, DC, and AC+DC.

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 4 shows the percentage of profiles where the winner changes using RC. 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.

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.

Data from experiment over t-shirt dataset show 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.

We also run some experiments using Borda. Theorem 1 shows that Borda is vulnerable to DCRC. Surprisingly, the deterministic algorithm for checking whether the winner can

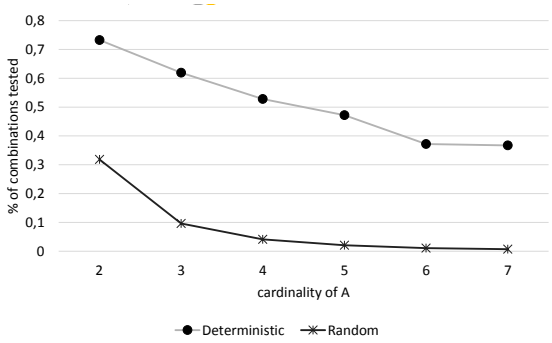


Figure 5: Borda deterministic and non-deterministic algorithm: comparison.

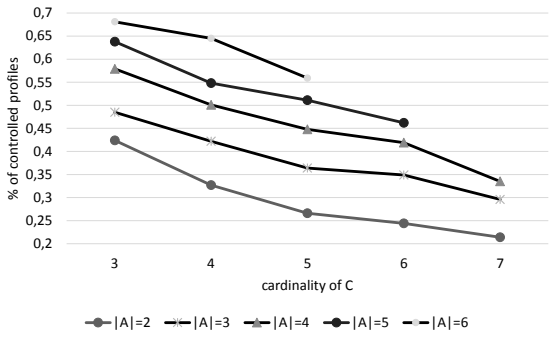


Figure 6: Borda: percentage of profiles (over 1000) with successful DCRC.

be changed by replacing candidates needs to test many combinations, as shown in Figure 5. Also, the number of profiles where the control succeeds decreases when the cardinality of C increases, as shown in Figure 6.

These results suggest that, even if the worst case theoretical analysis tells us that a voting rule is vulnerable to a certain control action, or resistant, the conditions that make it susceptible to the control action could be difficult to find in real-world scenarios, and this could sometimes lead to a reverse situation in practice. Veto, for instance, is resistant to DCRC, but in practice it is very easy to control and this can be done in almost all the profiles. On the other hand, Borda is vulnerable to DCRC, but in practice, when the size of the profile grows, it is unlikely to find a combination that changes the winner.

We also performed experiments with the data collected using the AGH course selection dataset. However, we do not report them here since they show the same trends as the ones of the other datasets.

Conclusions

After reporting theoretical results that show that Plurality and Veto are difficult to control, while Borda is easy, with respect to DCRC, we also performed an extensive experimental work, using real-world data sets, to test if k-approval and Borda are 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. Furthermore, the results show that Borda becomes more resistant to replacement control when the size of the profile grows. 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. These results suggest that the study of computational complexity in the worst case is not enough to ensure a significant protection to the system, as reported in many works such as (Bartholdi, Tovey, and Trick 1992; Faliszewski and Procaccia 2010; Faliszewski et al. 2009; Faliszewski, Hemaspaandra, and Hemaspaandra 2011; Walsh 2010). Experimental analysis is needed to get a deep comprehension of the likelihood of the conditions that make the system vulnerable/resistant to the control action.

References

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

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

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

Faliszewski, and Procaccia. 2010. Ai's war on manipulation: are we winning? *AI Magazine* pp. 53–64.

Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L. A.; and Rothe, J. 2009. Llull and copeland voting computationally resist bribery and control. *JAIR* 275–341.

Faliszewski, P.; Hemaspaandra, E.; and Hemaspaandra, L. A. 2010. Using complexity to protect elections. *Commun. ACM* 53(11):74–82.

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

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

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

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

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

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

Walsh, T. 2010. Is computational complexity a barrier to manipulation? *CoRR*.