Journal of Computer and System Sciences 81 (2015) 661–670

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Control complexity in Bucklin and fallback voting: An experimental analysis[☆]

Gábor Erdélyi[a], Michael R. Fellows[b], Jörg Rothe[c,*], Lena Schend[c]

[a] School of Economic Disciplines, University of Siegen, 57076 Siegen, Germany
[b] Parameterized Complexity Research Unit, Charles Darwin University, Darwin, NT 0909, Australia
[c] Institut für Informatik, Heinrich-Heine-Universität Düsseldorf, 40225 Düsseldorf, Germany

ARTICLE INFO

Article history:
Received 14 December 2012
Received in revised form 30 April 2014
Accepted 5 September 2014
Available online 18 November 2014

Keywords:
Computational social choice
Control complexity
Bucklin voting
Fallback voting
Experimental analysis
Election systems

ABSTRACT

Control in elections models situations in which an external actor tries to change the outcome of an election by restructuring the election itself. The corresponding decision problems have been shown NP-hard for a variety of voting systems. In particular, in our companion paper [16], we have shown that fallback and Bucklin voting are resistant (in terms of NP-hardness) to almost all of the common types of control. While NP-hardness results for manipulation (another way of tampering with the outcomes of elections) have been challenged experimentally (see, e.g., the work of Walsh [38,37]), such an experimental approach is sorely missing for control. We for the first time tackle NP-hard control problems in an experimental setting. Our experiments allow a more fine-grained analysis and comparison—across various control scenarios, vote distribution models, and voting systems—than merely stating NP-hardness for all these control problems.

1. Introduction

The notion of *control* in elections was introduced by Bartholdi et al. [4] to model situations where an external actor, "the chair," tries to change the outcome of an election by restructuring the election itself. In the *constructive control* scenario, the chair aims at making a designated candidate the winner of the changed election, while in the *destructive control* scenario, introduced by Hemaspaandra et al. [23], the chair tries to prevent the current winner from winning. To achieve his goal, the chair might add or delete candidates or voters, or partition them in the course of a control action. Another way of influencing the outcome of an election is *manipulation*, where a coalition of voters (consisting of one or more voters) participating in the election tries to change the outcome by casting insincere votes, see [3,2,10].

Both scenarios have been studied in terms of their worst-case complexity for various voting systems (see, e.g., the surveys of Faliszewski et al. [20,18], Conitzer [9], and Faliszewski and Procaccia [17]). One of the main issues of this line of research is to determine a natural voting system with a deterministic polynomial-time winner determination procedure that is resistant to as many types of control as possible, where resistance is defined as NP-hardness of the corresponding control

[☆] Preliminary versions of parts of this paper appear in the proceedings of the *11th International Symposium on Experimental Algorithms* [31] and the COMSOC special session at the *12th International Symposium on Artificial Intelligence and Mathematics* [32], and were presented at the *Tenth International Meeting of the Society for Social Choice and Welfare* and at the *Dagstuhl Seminar "Computation and Incentives in Social Choice"* [34].

[*] Corresponding author.
E-mail address: rothe@cs.uni-duesseldorf.de (J. Rothe).

http://dx.doi.org/10.1016/j.jcss.2014.11.003
0022-0000/© 2014 Elsevier Inc. All rights reserved.

problem. As shown in our companion paper [16], fallback voting (a hybrid voting system introduced by Brams and Sanver [6] that combines Bucklin and approval voting) has the broadest resistance to electoral control currently known to hold.

To complement these theoretical results, we conducted an experimental analysis of the control complexity in fallback and Bucklin elections, following the approach proposed by Walsh [37,38] that he, Davies et al. [12], and Narodytska et al. [28] (see also [8]) applied to manipulation problems for voting systems such as single transferable vote (STV), veto, Borda's, Nanson's, and Baldwin's rules. They showed that these voting systems can often be manipulated effectively, even though their manipulation problems are NP-hard. Such an experimental approach has been sorely missing for NP-hard control problems; this paper makes the first such attempt. Since both our classical and our parameterized complexity results on control in Bucklin and fallback voting [16] refer to a *worst-case* measure of complexity, they leave open the possibility that many elections can still be controlled in a reasonable amount of time. While the above-mentioned papers [38,37,12,28] focus on *constructive* manipulation problems only, we study both *constructive* and *destructive* control problems experimentally.

When generating random elections in our experiments, we consider two probability distributions: the *Impartial Culture model* where votes are distributed uniformly and are drawn independently, and the *Two Mainstreams model*, introduced here to model two mainstreams in society by adapting the *Pólya–Eggenberger urn model* [5]. In general, our findings indicate that some of the investigated NP-hard control problems can often be solved effectively in practice, whereas for other types of control our experimental results suggest that their problems may indeed be hard to solve even on random instances. Our experiments also allow a more fine-grained analysis than merely stating NP-hardness for all the corresponding control problems. Specifically, we can quantitatively compare constructive with destructive control, control across various voting systems in various control scenarios, and our two particular models of vote distribution.

Related work Many of the recent results on the hardness of manipulation problems are concerned with either *typical-case analyses* and *frequency of manipulability*[1] or *quantitative versions of the Gibbard–Satterthwaite theorem*; see, in particular, the papers by Mossel et al. [27], Procaccia and Rosenschein [30], Dobzinski and Procaccia [13], Friedgut et al. [21], Zuckerman et al. [42], Xia and Conitzer [39,40], Xia et al. [41], Isaksson et al. [25], Peleg [29], Baharad and Neeman [1], and Slinko [35, 36]. These papers provide theoretical insights into why NP-hard manipulation problems can still be easy to solve in practice. They are complemented by the previously mentioned empirical and experimental studies regarding manipulation problems due to Walsh [38,37], Davies et al. [12,11], and Narodytska et al. [28]. Some of these theoretical and experimental results have been surveyed by Rothe and Schend [32]. To the best of our knowledge, the experimental investigation of NP-hard control problems is new to this paper.

Organization We start by giving a brief summary of the theoretical results of control in fallback and Bucklin elections in Section 2, which have been shown in the companion paper [16]. In Section 3 we describe the setup of our experiments including the sampling of randomly generated elections and we describe the implemented algorithms. Section 4 provides a summary of our findings and a discussion of the results before we conclude in Section 5 with open questions and suggestions for future work.

2. Notions and theoretical results from the companion paper [16]

Table 1 gives an overview of the theoretical results for control complexity in fallback and Bucklin elections that were shown in the companion paper [16]; we refer to that paper also for the formal definitions of notions mentioned in the table, such as the studied control problems (and the tie-handling rules TE and TP), the notions of immunity, susceptibility, (parameterized) resistance, and vulnerability.

Note that, based on the work of Bartholdi et al. [4], Hemaspaandra et al. [23], and Faliszewski et al. [19], 22 types of electoral control have been defined originally and we conducted our experiments for all those 18 types of electoral control Bucklin and fallback voting are not vulnerable to, i.e., the corresponding decision problems are not known to be solvable in deterministic polynomial time. Recently, Hemaspaandra et al. [24] discovered that, depending on the winner model, some of the destructive cases of control by partition of candidates collapse, so there are in fact fewer than 22 distinct types of electoral control. Table 1 indicates these collapses (for the co-winner model) by having only one R entry for two (in fact, identical) control types.

3. Experimental setup

In this section we describe the framework of our experimental analysis beginning with the general setup followed by an introduction of the distribution models used to sample the randomly generated elections. We will conclude this section by giving a high-level description of the algorithms.

As stated in [16, Section 2] the instances of the problems modeling control by adding or by deleting either candidates or voters contain a parameter k bounding the number of candidates or voters that can be added or deleted, which is crucial

[1] Erdélyi et al. [14,15] noted that such approaches—which indeed are very valuable to pursue—are not to be mistaken for work in average-case complexity theory in the sense of Levin [26].

Table 1
Overview of classical and parameterized complexity results for control in Bucklin and fallback voting shown in [16]. All results hold in both the co-winner and the unique-winner model. Key: I = immune, S = susceptible, R = resistant, R* = parameterizedly resistant, V = vulnerable, TE = ties eliminate, and TP = ties promote.

Control by	Fallback voting		Bucklin voting	
	Constructive	Destructive	Constructive	Destructive
Adding Candidates	R*	R*	R*	R*
Adding Candidates unlimited	R	R	R	R
Deleting Candidates	R*	R*	R*	R*
Partition of Candidates – TE	R	R	R	R
Run-off Partition of Candidates – TE	R		R	
Partition of Candidates – TP	R	R	R	R
Run-off Partition of Candidates – TP	R		R	
Adding Voters	R*	V	R*	V
Deleting Voters	R*	V	R*	V
Partition of Voters – TE	R	R	R	R
Partition of Voters – TP	R	R	R	S

for the running time of the implemented algorithms. To realize our experiments in an acceptable time frame, we confine ourselves to the cases of $k = \lfloor m/3 \rfloor$ and $k = \lfloor n/3 \rfloor$, respectively, where m is the number of candidates and n is the number of voters. This restriction of the parameter k has a purely practical purpose needed to realize our experimental approach. As we will see later in the high-level description of our algorithms, we tackle the task of solving instances of NP-hard problems by exhaustively testing every subset up to size k on whether adding/deleting it is a successful control action. On the positive side, we have that a yes-instance for a given k is also a yes-instance for each $k' \geq k$, so the number of yes-instances found in our experiments for smaller k directly transfers to instances with bigger values of k. On the negative side, if no successful control actions could be found for a given k, we cannot make conclusions for the same election with a bigger value of k.

For similar reasons, as we have to cope with NP-hard problems, a time limit of ten minutes has been implemented such that the algorithm stops when exceeding this limit, indicating by the output "timeout" that the search process is aborted unsuccessfully. Again, this allows us to handle the worst-case scenarios in a reasonable amount of time. In our experiments we implemented the same timeout value for all investigated types of control. As our results will show, the different control types react differently to this constant timeout threshold, so a tuning of the timeout-parameter would be an interesting issue for further experiments. Also, varying the timeout value depending on the election size at hand might be an interesting approach.

We randomly generated elections (C, V) with $m = \|C\|$ and $n = \|V\|$ for all combinations of n and m chosen from $\{4, 8, 16, 32, 64, 128\}$. In the adding-candidates and adding-voters scenarios, the spoiler sets D and V' have the same size as the set of registered candidates and voters, respectively; i.e., $\|D\| = \|C\|$ and $\|V'\| = \|V\|$. Each such combination of n and m is one data point for which we evaluated 500 of these elections, trying to determine for each given election whether or not control is possible, and if it is possible, we say that this election is *controllable*. This restriction to 500 elections per data point, again, results from practical issues balancing out manageability and informative value of the experiments conducted.

The algorithms and data-generation programs are implemented in *Octave 3.2* and the experiments were run on a 2.67 GHz Core-I5 750 with 8 GB RAM.

3.1. Election generation and distributions of votes

Before we specify the different distribution models underlying our election generation, we explain how random votes can be cast in the considered voting systems and how many different votes can exist. Assuming that the generated election has m candidates, in Bucklin voting a random vote can be obtained by generating a random permutation over the m different candidates, so the overall number of different votes in Bucklin elections is $m!$. In fallback voting, random votes can be generated as follows:

- randomly draw a preference p from all $m!$ possible preferences with m candidates;
- randomly draw a number, say $\ell \in \{0, 1, \ldots, m\}$, of approved candidates;
- the generated vote consists of the first ℓ candidates in p.

Thus, there can be $\sum_{\ell=0}^{m} \binom{m}{\ell} \ell!$ different votes in fallback elections with m candidates.

We now turn to the distribution models and we start with the *general Pólya–Eggenberger urn model* (*PE model*), see [5], in which a set of votes is sampled in the following way: Assume that we have an urn containing all possible votes that can be cast given a certain voting system and let the number of different votes be denoted by t. For Bucklin voting, for example, $t = m!$, while for fallback voting we have that $t = \sum_{\ell=0}^{m} \binom{m}{\ell} \ell!$, as explained above. To sample an electorate consisting of n votes, we proceed in the following way for a fixed parameter b:

- randomly draw one preference from the given urn—this is the first of the n votes that shall be sampled,
- put the preference back into the urn along with b additional copies of it,
- randomly draw the second vote from the new urn,
- put the second vote back into the urn along with b additional copies of it,

  ⋮
- randomly draw the $(n-1)$st vote from the new urn,
- put the $(n-1)$st vote back into the urn along with b additional copies of it,
- randomly draw the nth vote from the new urn.

The correlation of the sampled votes strongly depends on the parameter b. By setting $b = 0$, we obtain the *Impartial Culture model* (*IC model*) which samples uniformly distributed votes out of all possible preferences since in each step the just drawn preference is put back into the urn without adding any more preferences.

To sample correlated votes, the usual approach (also employed by, e.g., Walsh [38]) is to use the above model with the parameter $b = t$. This means that when the first preference is drawn from the urn, it is put back into the urn along with t additional copies, leading to a probability of 0.5 that the second preference will be the same as the first one and, depending on the preferences that are drawn in each step, this effect can be intensified during the sampling process. Thus, there is a relatively high probability that many (or even all) sampled votes can be identical. In the setting of manipulation, where the preferences of the manipulators can be freely set independently of the nonmanipulators' preferences, this effect has less impact while in the control scenarios considered in this work, identical preferences of the voters (including e.g., unregistered votes that may be added), would artificially make control impossible or easy to find and thus would trivialize the problem.

Therefore, we introduce the *Two Mainstreams model* (*TM model*), which is the following adaption of the PE model when sampling correlated votes in our experiments:

- depending on the voting system, randomly draw two preferences out of an urn containing all possible, say t, preferences—recall that either $t = m!$ (for Bucklin) or $t = \sum_{\ell=0}^{m} \binom{m}{\ell} \ell!$ (for fallback);
- put each preference back into the urn with t additional copies;
- draw the votes out of this urn independently at random with replacement.

Each of the two preferences drawn in the first step can be interpreted as a representative of one "mainstream" in society (e.g., liberal and conservative).

3.2. A high-level description of the algorithms

Our algorithms are greedy heuristics, designed so as to test the most "promising" cases (depending on the control type at hand) first, by using appropriate preorderings. We only provide a high-level description. All algorithms for the different types of control share the same essential method of testing various subsets, and they differ only in the type of preordering and internal testing. Before actually searching for a successful sublist of voters or subset of candidates, the algorithms check conditions that, if true, indicate that the given instance is a no-instance. Let c be the designated candidate in the control problems defined in the companion paper [16]. Depending on the control type, some of the following conditions are tested:

Condition 1 (applied to all constructive cases): The designated candidate is ranked last (for Bucklin), or is ranked last or disapproved (for fallback), in every vote.

Condition 2 (applied to control by deleting voters): For each $k' \leq k$, determine the smallest i and j such that

$$score^i_{(C,V)}(c') \geq \lfloor (\|V\|-k')/2 \rfloor + 1 + k' \quad \text{and} \quad score^j_{(C,V)}(c) \geq \lfloor (\|V\|-k')/2 \rfloor + 1$$

hold for $c' \in C - \{c\}$. Note that $i \leq j - 1$ for all $k' \leq k$.

Condition 3 (applied to control by adding voters): For each $k' \leq k$ determine the smallest i and j such that

$$score^i_{(C,V)}(c') \geq \lfloor (\|V\| + k')/2 \rfloor + 1 \quad \text{and} \quad score^j_{(C,V)}(c) \geq \lfloor (\|V\| + k')/2 \rfloor + 1 - k'$$

hold for $c' \in C - \{c\}$. Note that $i \leq j - 1$ for all $k' \leq k$.

Condition 4 (applied to all destructive cases): In the given election, the winner has a strict majority on the first level already.

Condition 1 (respectively, Condition 4) is tested for every constructive (respectively, destructive) control type investigated here. Note that these conditions are checked in the election for both the registered and the unregistered voters for control by adding voters, and both for the original and the spoiler candidates for control by adding candidates. Condition 2 is additionally tested in the deleting-voters cases and tries to find the smallest level i on which another candidate than the designated one would still have a strict majority in the election with the reduced voter list even if he or she lost points

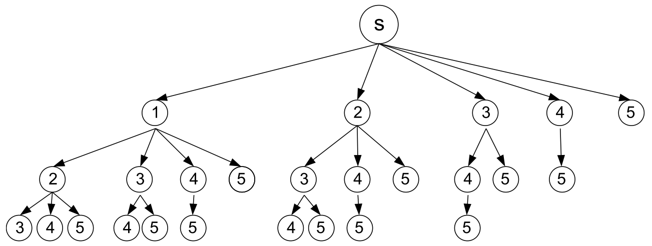


Fig. 1. Tree for $n = 5$ voters where up to $k = 3$ voters may be deleted. A node i corresponds to voter v_i after the preordering.

on this level from every voter that has been deleted. Moreover, we search for the smallest level j on which the designated candidate c reaches a strict majority in the election after the voters have been deleted if he or she is not harmed at all by the deleted voters. This is done for all possible numbers of deleted voters. If $i < j$ for all possible numbers of deleted voters, c is hopeless since there will always be a candidate reaching a strict majority before c. Condition 3 checks whether an analogous situation occurs when voters are added and is thus tested for the adding-voters cases only.

After having excluded these trivial cases, each of the algorithms searches for a successful sublist/subset of preordered versions of V or C. Let us describe this procedure only for constructive control by deleting voters in detail. In this case, the voters are preordered ascendingly for c; that is, after the preordering v_1 is a voter ranking c worst and v_n is a voter ranking c best among all voters. (In fallback voting, the "worst" position for a candidate is to be not approved at all.) The algorithm now starts with deleting those votes c benefits least of. It follows the procedure of a depth-first search on a tree of height k that is structured as shown in Fig. 1. In each node, it is tested whether deleting the votes on the path is a successful control action. For example, on path $s \to 1 \to 2 \to 3$ the algorithm tests the sublists (v_1), (v_1, v_2), (v_1, v_2, v_3) and then tracks back testing the sublists (v_1, v_2, v_4), (v_1, v_2, v_5), (v_1, v_3), (v_1, v_3, v_4), and so on. The branches on the left side are visited first and, due to the preordering of the votes, these are the votes c benefits least of.

For the adding-voters cases, the unregistered voters are ordered in a descending order for the designated candidate, and the algorithm proceeds similarly as the algorithm for the deleting-voters cases. With this preordering, the algorithm first tests those voters the designated candidate can benefit most from when these are added to the voter list.

For the partition-of-voters cases, the algorithm considers every possible sublist of the voter list up to size $k = \lfloor n/2 \rfloor$ as V_1, sets $V_2 = V - V_1$, and tests whether this is a successful control action or not. For the constructive cases, the voters are preordered descendingly with respect to the designated candidate, whereas for the destructive control cases no preordering is implemented.

In the candidate control scenarios, the candidates are also ordered with respect to the designated candidate, where a *descending order* here means that the first candidate has the most voters ranking him or her before the designated candidate and the last candidate has the fewest voters doing so. An *ascending order* is defined analogously. Again, in the adding-candidates case, the votes over all candidates (including the spoiler candidates) are considered. A descending ordering is used for finding control actions for constructive control by deleting candidates and for destructive control by adding candidates, whereas for the remaining candidate control cases an ascending order is used.

In the worst case, our algorithms check all possible subsets of size k, so they have a worst-case running time of $\sum_{\ell=1}^{k} \binom{n}{\ell}$ for voter control and $\sum_{\ell=1}^{k} \binom{m}{\ell}$ for candidate control, where n is the number of voters and m is the number of candidates. Finally, note that for each yes-answer, our algorithms also provide the corresponding successful control action.

4. Summary of experimental results

Table 2 summarizes our experimental results on control in Bucklin and fallback voting. We investigated the two voting systems only for those control types they are not known to be vulnerable to, which is indicated by an R*-, R-, or an S-entry in Table 1. That is, destructive control by adding and by deleting voters (DCAV and DCDV) are omitted in Table 2. Also, since our algorithms use the parameter k bounding the number of candidates to be added, constructive and destructive control by adding an unlimited number of candidates (CCAUC and DCAUC) are not considered either. For each combination of any of the remaining 18 control types, any of the two voting systems (Bucklin and fallback voting), and any of the two distribution models (IC and TM), we tested a total of $18,000 = 36 \cdot 500$ elections, varying over the 36 data points with different values for m and n, as explained above. This gives a total of $1,296,000 = 18 \cdot 4 \cdot 18,000$ generated and tested elections.

Table 2 gives an overview of the percentage of timeouts for each such combination of control type/voting system/distribution model, and also the minimal and maximal percentage of yes-instances observed. We do not discuss the results for all these cases in detail here. Rather, we will focus on adding/deleting/partitioning of voters to very briefly discuss some observations from our experiments, to exemplify some of the numbers in Table 2. For those cases that we discuss in detail, we provide plots giving the percentage of yes-instances, timeouts, and average computational costs for all different election sizes that were tested. Note that a comprehensive presentation of all results obtained so far containing the above information for all cases (showing 168 plots of experiments in total) can be found in the appendix of the 370-page technical report by Rothe and Schend [33], who in addition to Bucklin and fallback voting conducted experiments for plurality voting.

Table 2
Overview of experimental results on control in Bucklin and fallback voting. Key: The “min” and “max” columns give the minimal and maximal percentage of yes-instances observed in all tested instances for the given control type, including those elections where timeouts occurred; “timeout” gives the percentage of timeouts that occurred for the total of 18,000 elections tested in this control case.

	Fallback voting						Bucklin voting					
	min		max		timeout		min		max		timeout	
	IC	TM	IC	TM	IC	TM	IC	TM	IC	TM	IC	TM
CCAC	1	0	11	7	51	50	0	0	23	11	50	49
DCAC	53	39	92	71	11	14	71	42	99	77	6	12
CCDC	13	15	33	36	37	37	13	17	58	45	34	37
DCDC	8	12	78	63	15	22	48	25	99	77	7	18
CCPC-TE	0	0	19	18	62	64	1	0	57	37	57	62
DCPC-TE	8	16	88	65	18	29	49	29	100	78	10	23
CCPC-TP	1	0	17	17	62	64	1	0	60	38	57	61
DCPC-TP	8	16	87	61	18	29	49	29	100	82	9	23
CCRPC-TE	1	1	18	14	62	63	1	0	60	45	57	62
DCRPC-TE	8	16	86	68	20	29	46	29	100	84	9	23
CCRPC-TP	1	0	19	14	62	63	1	1	56	25	53	61
DCRPC-TP	8	16	85	68	21	29	45	27	100	81	10	23
CCAV	4	1	99	41	13	13	2	1	99	41	11	6
CCDV	2	1	97	39	16	12	2	1	100	42	11	7
CCPV-TE	2	0	97	34	9	45	2	0	98	32	8	44
DCPV-TE	50	34	100	88	4	16	64	40	100	89	4	10
CCPV-TP	1	1	53	20	40	50	1	0	72	23	31	48
DCPV-TP	37	27	100	87	6	17	60	39	100	88	3	10

4.1. Adding and deleting voters

We here briefly discuss some results for control by deleting voters only, since those for control by adding voters are very similar, in both Bucklin and fallback voting. Fig. 2 shows the results for control by deleting voters for Bucklin voting in the IC model and in detail we have the percentage of yes-instances in Fig. 2a, where the highest percentage of 100 % and the lowest percentage of 2 % can also be seen in the “max” and “min” column in Table 2. Fig. 2b gives the detailed occurrence of timeouts for the different election sizes and Fig. 2c shows the average time needed to determine whether a given Bucklin election generated under the IC model can be controlled by deleting voters or not. Remember that in the latter figure the average values do not consider those elections where the algorithm exceeded the time limit of 600 seconds.

In the IC model, increasing the number of candidates decreases the number of yes-instances in the generated Bucklin elections. On the other hand, the number of yes-instances increases as the number of voters grows. In the TM model, the same correlations can be observed but here, again, the total number and percentage of yes-instances is smaller than in the IC model.

Fallback voting behaves very similarly, so for both distributions and both voting systems increasing the number of candidates makes successful control actions by deleting voters less likely.

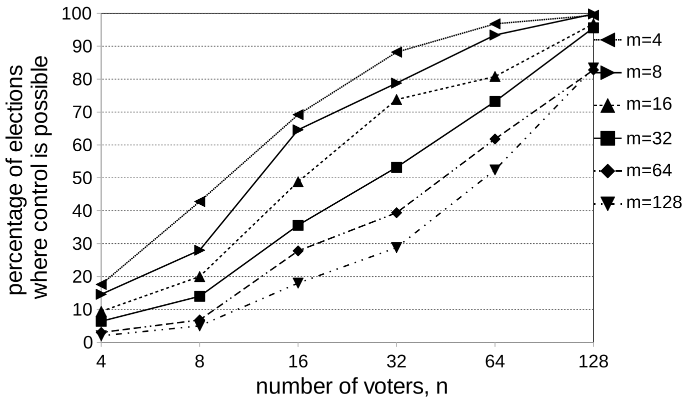
In both voting systems and in both distribution models, timeouts occur whenever the number of voters exceeds 32. If the number of candidates is 128, we have timeouts already with 16 voters. This can also be seen in the development of the computational costs shown in Fig. 2c after the peak for $n = 16$. For larger electorates, the average computational costs drop, since the number of timeouts increases as the number of no-instances diminishes.

4.2. Partition of voters

As mentioned in [16, Section 2], control by partition of voters comes in four problem variants, where each case must be investigated separately. We very briefly discuss some observations made for these control types.

For constructive control by partition of voters in model TP we made the following observations: Similarly to control by deleting or by adding voters, the number of controllable elections increases as the number of voters increases. This was observed for both voting systems investigated. Using the tie-handling model TE instead of TP, in both Bucklin and fallback voting an increase of yes-instances in the constructive cases is evident. By contrast, in the destructive counterparts no significant difference can be observed with respect to the tie-handling rule used.

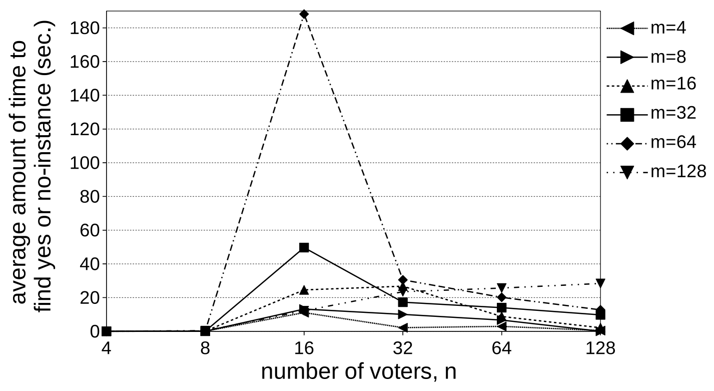
The most striking results are those obtained for the destructive cases. Here we have that, for both tie-handling models in the TM model, the average number of controllable elections is very high; and in the IC model, control is almost always possible, see Fig. 3. In light of the fact that for these cases the resistance proofs of Theorems 3.19, 3.21, and 3.25 in [16] tend to be the most involved ones (yielding the most complex instances for showing NP-hardness), these results might be surprising at first glance. However, one explanation for the observed results can be found in exactly this fact: The elections constructed in these reductions have a very complex structure which seems to be unlikely to occur in randomly generated elections (at least in elections generated under the distribution models discussed in this paper). Another explanation is that the problems used to reduce from in these proofs tend to be easy to solve for small input sizes, but due to the complexity



(a) Percentage of yes-instances.

$m\backslash n$	4	8	16	32	64	128
4	0	0	0	10	3	1
8	0	0	0	20	7	0
16	0	0	0	23	19	3
32	0	0	0	39	27	4
64	0	0	1	49	38	17
128	0	0	31	53	47	17

(b) Percentage of timeouts.



(c) Average time the algorithm needs to give a definite output, instances where timeouts occur are excluded.

Fig. 2. Bucklin voting in the IC model for CCDV.

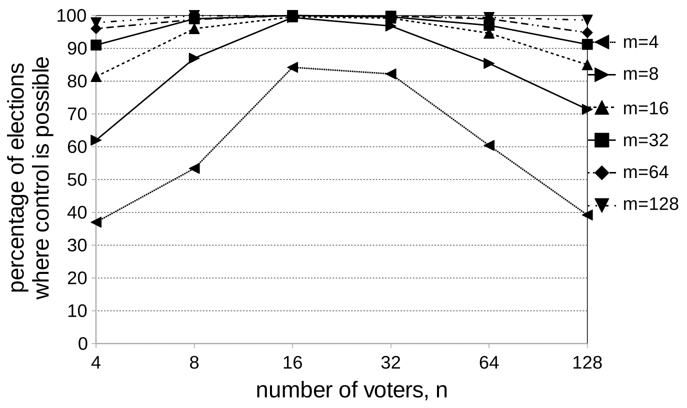
of the reduction, the resulting elections have many voters/candidates compared to the elections generated for the conducted experiments.

In the destructive cases, the number of timeouts is for both voting systems the lowest among all control types investigated. In Bucklin elections with uniformly distributed votes and for destructive control by partition of voters in model TP, for only 3.32 % of the elections no decision can be made within the time limit. As can be seen in the table, timeouts begin to occur for those elections where the number of voters exceeds 16. But, again, we have to emphasize that these values are very low compared to other types of control. This explains the plateaus all graphs show. On the one hand, increasing the number of voters increases the number of yes-instances. But on the other hand, for more than 16 voters timeouts begin to diminish the fraction of observed yes-instances. Also, the average running time of the algorithm for those instances where the time limit is not exceeded is rather low, compared to other types of control, see Fig. 3c. The highest computational costs occur for those election sizes where the most no-instances were observed. As expected, in the corresponding constructive cases the number of timeouts is significantly higher and so are the average computational costs.

4.3. Discussion and comparison of voting systems, control types, and distribution models

Finally, we summarize the main findings of our experiments, which allow a more fine-grained analysis and comparison—across various control scenarios, vote distribution models, and voting systems—than merely stating NP-hardness for all these problems. Obviously, our findings are limited by the experimental setup and, of course, the fact that exponential time seems unavoidable for these problems unless P = NP; thus, our conclusions cannot be generalized unconditionally.

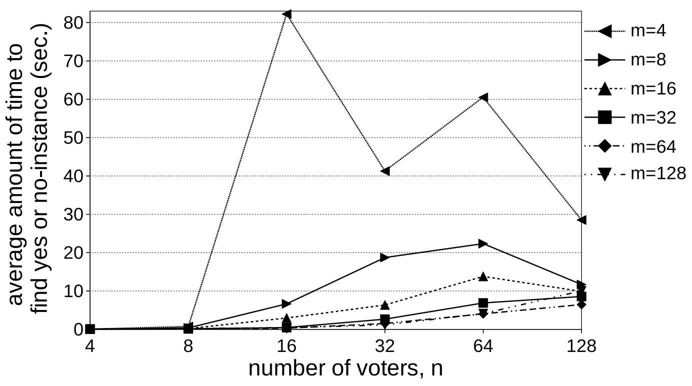
Distribution models: IC versus TM Comparing the results for the different distribution models, we see that in every voting system for all control types studied (except fallback voting in constructive control by deleting candidates) the overall number of yes-instances is higher in the IC than in the TM model. This may result from the fact that in elections with uniformly distributed votes, all candidates are likely to be approximately equally preferred by the voters. So both constructive and destructive control actions are easier to find by our greedy algorithms. This also explains the observation that the IC model produces fewer timeouts.



(a) Percentage of yes-instances.

$m\backslash n$	4	8	16	32	64	128
4	0	0	0	18	40	61
8	0	0	0	3	15	29
16	0	0	0	1	5	15
32	0	0	0	0	3	9
64	0	0	0	0	1	5
128	0	0	0	0	1	1

(b) Percentage of timeouts.



(c) Average time the algorithm needs to give a definite output, instances where timeouts occur are excluded.

Fig. 3. Fallback voting in the IC model for DCPV-TP.

Constructive versus destructive control For all investigated types of control where both constructive and destructive control was investigated, we found that the destructive control types are experimentally much easier than their constructive counterparts, culminating in almost 100 % of controllable elections for certain control types in the IC model. Compare this with the theoretical insight of Hemaspaandra et al. [23] that (unique-winner) destructive control problems disjunctively truth-table-reduce to their (co-winner) constructive counterparts and thus are never harder to solve, up to a polynomial factor (see also the corresponding observation of Conitzer et al. [10] regarding manipulation): In fact, destructive control tends to be even *much* easier than constructive control in our experiments.

Comparison across voting systems For constructive control, we have seen that fallback and Bucklin voting show similar tendencies and numbers of yes-instances regarding voter control. Bucklin voting tends to have more controllable elections in candidate control. In both voting systems, constructive control by partition of candidates seems to be the hardest control problem investigated, at least for our algorithms, as the most timeouts have occurred in these cases.

Adding candidates/voters versus deleting candidates/voters For fallback and Bucklin voting, we have seen that the results for control by adding voters do not differ significantly from those observed for control by deleting voters, suggesting that both types of control are roughly equally hard. By contrast, comparing control by adding candidates to control by deleting candidates in the constructive case leads to different findings. In both voting systems and both control types, we have small numbers of yes-instances. In the constructive case, however, we observed that the number of yes-instances for control by deleting candidates is significantly higher. These findings are perhaps not overly surprising, since in the voting systems considered here adding candidates to an election can only worsen the position of the designated candidate in the votes. That is, constructive control can be exerted successfully only if by adding candidates rivals of the designated candidate lose enough points so as to get defeated by him or her. This, in turn, can happen only if the designated candidate was already a highly preferred candidate in the original election.

Constructive voter versus candidate control For fallback and Bucklin voting, we can also compare constructive candidate and voter control directly. In both voting systems and both distribution models, the number of yes-instances for constructive control by adding voters is around four times higher than the number of yes-instances in the corresponding candidate control type, which confirms the argument above, saying that adding candidates cannot push the designated candidate directly. Constructive control by deleting voters can be successfully exerted more frequently when votes are less correlated, whereas the proportion of successful control actions for deleting candidates is about the same for both considered distribution models. The observed differences between these types of voter and candidate control may result from the fact that adding or deleting candidates only shifts the position of the designated candidate, which may not influence the outcome of the election as directly as increasing or decreasing the candidates' scores by adding or deleting voters does. This may explain why voter control can be tackled more easily than candidate control by greedy approaches such as ours.

5. Conclusions and open questions

Reviewing the results obtained from our experiments, we can roughly group the investigated control types in three different categories:

1. For all destructive control cases, we could show that for instances randomly generated with either of the voter distribution models considered here, the control problems are easily solvable by our greedy approach. This suggests that the NP- and W[2]-hardness results from [16] for these cases describe solely the worst-case behavior and do not give information about the complexity for typical instances, assuming the used voter distribution models do give "typical" instances.
   For constructive voter control by adding, deleting, or partitioning in model TE, we have to distinguish between the two types of input instances. For uniformly distributed electorates, we have seen that these control actions can also be easily computed by our greedy approach, whereas for instances with correlated votes the problems become harder to solve. So the complexity of these problems in practice depends immensely on the given instance's structure. These problems cannot be grouped into some specific category, as they fall somewhere between the first and the second category.
2. The second category classifies those problems that are at least for small election sizes efficiently solvable in our setting. This category contains the constructive cases of control by deleting candidates and partition of voters in model TP. For these problems our experiments show that for very small instances the problems are in practice easy to solve, but the worst case that is reflected by the theoretical hardness results is likely to occur even for random instances (according to IC and TM) when their size increases.
3. The remaining cases of constructive candidate control (namely, adding candidates and all variants of partition of candidates) form the third and last category in which we collect those control problems that are hard to solve by our algorithms for all considered instance sizes and structures. For these problems, our experiments may allow the conclusion that these problems seem to be hard to solve even in practice and on random instances (according to IC and TM).

Summarizing our results, we have seen that the natural parameterization by the number of deleted/added voters might not be fine-grained or expressive enough to give information about the behavior of instances actually occurring in practice. Even though parameterized complexity offers a more differentiated worst-case analysis with respect to the considered parameter than NP-hardness, we have seen that a further experimental analysis can provide further insights.

From an experimental perspective, this paper has done the first step. The next step would be to do empirical studies based on real-world election data instead of mere simulations of randomly generated elections.

Just as Walsh [37,38] observes for manipulation in the veto rule and in STV, for all types of control investigated in our experiments, the curves do not show the typical phase transition known for "really hard" computational problems such as the satisfiability problem (see [22,7] for a detailed discussion of this issue). These observations raise the question of how other distribution models influence the outcome of such experiments. Furthermore, the algorithms implemented could be improved in terms of considering a higher number of elections per data point, increasing the election sizes, or allowing a higher number of voters or candidates to be deleted or added in the corresponding control scenarios. For a more detailed analysis of the behavior of the parameterized problem variants, the parameter considered here, namely the number of voters or candidates that can be added or deleted, could be varied in future experiments. Besides this, other voting systems can be analyzed easily using modifications of our algorithms, since only their winner determination has to be implemented in addition to a few minor adjustments such as trivial-case checks for the investigated control scenarios tailored to the voting system at hand.

Acknowledgments

We thank the anonymous JCSS and SEA-2012 referees for their very helpful comments and suggestions. We thank Toby Walsh for interesting discussions on experimental studies of voting systems, Volker Aurich for giving us access to his computer lab, and Guido Königstein for his help in setting up our experiments. This work was supported in part by DFG grants ER 738/2-1, RO 1202/11-1, RO 1202/12-1, and RO 1202/15-1, by the European Science Foundation's EUROCORES program LogICCC, by the National Research Foundation (Singapore) under grant NRF-RF 2009-08, and by SFF grant "Cooperative Normsetting" of HHU Düsseldorf. This work was done in part while the first author was affiliated to Heinrich-Heine-Universität Düsseldorf and to Nanyang Technological University, Singapore, and while he was visiting Universität Trier, while the third author was visiting Stanford University and the University of Rochester, and while the first and third authors were visiting NICTA, Sydney, and the University of Newcastle in Australia, and we thank our hosts for their hospitality.

References

[1] E. Baharad, Z. Neeman, The asymptotic strategyproofness of scoring and condorcet consistent rules, Rev. Econ. Des. 7 (3) (2002) 331–340.
[2] J. Bartholdi III, J. Orlin, Single transferable vote resists strategic voting, Soc. Choice Welf. 8 (4) (1991) 341–354.
[3] J. Bartholdi III, C. Tovey, M. Trick, The computational difficulty of manipulating an election, Soc. Choice Welf. 6 (3) (1989) 227–241.
[4] J. Bartholdi III, C. Tovey, M. Trick, How hard is it to control an election? Math. Comput. Model. 16 (8/9) (1992) 27–40.
[5] S. Berg, Paradox of voting under an urn model: The effect of homogeneity, Public Choice 47 (2) (1985) 377–387.
[6] S. Brams, R. Sanver, Voting systems that combine approval and preference, in: S. Brams, W. Gehrlein, F. Roberts (Eds.), The Mathematics of Preference, Choice, and Order: Essays in Honor of Peter C. Fishburn, Springer, 2009, pp. 215–237.
[7] P. Cheeseman, B. Kanefsky, W. Taylor, Where the really hard problems are, in: Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1991, pp. 331–337.
[8] T. Coleman, V. Teague, On the complexity of manipulating elections, in: Proceedings of Computing: The 13th Australasian Theory Symposium, vol. 65, 2007, pp. 25–33.
[9] V. Conitzer, Making decisions based on the preferences of multiple agents, Commun. ACM 53 (3) (2010) 84–94.
[10] V. Conitzer, T. Sandholm, J. Lang, When are elections with few candidates hard to manipulate? J. ACM 54 (3) (2007), Article 14.
[11] J. Davies, G. Katsirelos, N. Narodytska, T. Walsh, An empirical study of Borda manipulation, in: V. Conitzer, J. Rothe (Eds.), Proceedings of the 3rd International Workshop on Computational Social Choice, Universität Düsseldorf, September 2010, pp. 91–102.
[12] J. Davies, G. Katsirelos, N. Narodytska, T. Walsh, Complexity of and algorithms for Borda manipulation, in: Proceedings of the 25th AAAI Conference on Artificial Intelligence, AAAI Press, August 2011, pp. 657–662.
[13] S. Dobzinski, A. Procaccia, Frequent manipulability of elections: The case of two voters, in: Proceedings of the 4th Workshop on Internet & Network Economics, in: Lecture Notes in Computer Science, vol. 5385, Springer-Verlag, December 2008, pp. 653–664.
[14] G. Erdélyi, L. Hemaspaandra, J. Rothe, H. Spakowski, Frequency of correctness versus average polynomial time, Inf. Process. Lett. 109 (16) (2009) 946–949.
[15] G. Erdélyi, L. Hemaspaandra, J. Rothe, H. Spakowski, Generalized juntas and NP-hard sets, Theor. Comput. Sci. 410 (38–40) (2009) 3995–4000.
[16] G. Erdélyi, M. Fellows, J. Rothe, L. Schend, Control complexity in Bucklin and fallback voting: A theoretical analysis, J. Comput. Syst. Sci. 81 (4) (2015) 632–660, in this issue.
[17] P. Faliszewski, A. Procaccia, AI's war on manipulation: Are we winning? AI Mag. 31 (4) (2010) 53–64.
[18] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, J. Rothe, A richer understanding of the complexity of election systems, in: S. Ravi, S. Shukla (Eds.), Fundamental Problems in Computing: Essays in Honor of Professor Daniel J. Rosenkrantz, Springer, 2009, pp. 375–406, chapter 14.
[19] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, J. Rothe, Llull and Copeland voting computationally resist bribery and constructive control, J. Artif. Intell. Res. 35 (2009) 275–341.
[20] P. Faliszewski, E. Hemaspaandra, L. Hemaspaandra, Using complexity to protect elections, Commun. ACM 53 (11) (2010) 74–82.
[21] E. Friedgut, G. Kalai, N. Nisan, Elections can be manipulated often, in: Proceedings of the 49th IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, October 2008, pp. 243–249.
[22] I. Gent, T. Walsh, Phase transitions from real computational problems, in: Proceedings of the 8th International Symposium on Artificial Intelligence, 1995, pp. 356–364.
[23] E. Hemaspaandra, L. Hemaspaandra, J. Rothe, Anyone but him: The complexity of precluding an alternative, Artif. Intell. 171 (5–6) (2007) 255–285.
[24] E. Hemaspaandra, L. Hemaspaandra, C. Menton, Search versus decision for election manipulation problems, in: Proceedings of the 30th International Symposium in Theoretical Aspects of Computer Science, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, February 2013, pp. 377–388.
[25] M. Isaksson, G. Kindler, E. Mossel, The geometry of manipulation: A quantitative proof of the Gibbard–Satterthwaite theorem, in: Proceedings of the 51st IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, October 2010, pp. 319–328.
[26] L. Levin, Average case complete problems, SIAM J. Comput. 15 (1) (1986) 285–286.
[27] E. Mossel, A. Procaccia, M. Rácz, A smooth transition from powerlessness to absolute power, J. Artif. Intell. Res. 48 (2013) 923–951.
[28] N. Narodytska, T. Walsh, L. Xia, Manipulation of Nanson's and Baldwin's rules, in: Proceedings of the 25th AAAI Conference on Artificial Intelligence, AAAI Press, August 2011, pp. 713–718.
[29] B. Peleg, A note on manipulability of large voting schemes, Theory Decis. 11 (4) (1979) 401–412.
[30] A. Procaccia, J. Rosenschein, Junta distributions and the average-case complexity of manipulating elections, J. Artif. Intell. Res. 28 (2007) 157–181.
[31] J. Rothe, L. Schend, Control complexity in Bucklin, fallback, and plurality voting: An experimental approach, in: Proceedings of the 11th International Symposium on Experimental Algorithms, in: Lecture Notes in Computer Science, Springer-Verlag, June 2012, pp. 356–368.
[32] J. Rothe, L. Schend, Typical-case challenges to complexity shields that are supposed to protect elections against manipulation and control: A survey, in: Website Proceedings of the Special Session on Computational Social Choice at the 12th International Symposium on Artificial Intelligence and Mathematics, January 2012, pp. 161–193.
[33] J. Rothe, L. Schend, Control complexity in Bucklin, fallback, and plurality voting: An experimental approach, Technical Report, March 2012, arXiv:1203.3967 [cs.GT], Computing Research Repository, arXiv.org/corr/, March 2012. Revised August, 2012.
[34] L. Schend, G. Erdélyi, J. Rothe, Control complexity in Bucklin and fallback voting: A theoretical and experimental analysis, in: E. Elkind, C. Klamler, J. Rosenschein, R. Sanver (Eds.), Dagstuhl Seminar 12101: "Computation and Incentives in Social Choice". Dagstuhl Seminar Proceedings, March 2012.
[35] A. Slinko, On asymptotic strategy-proofness of classical social choice rules, Theory Decis. 52 (4) (2002) 389–398.
[36] A. Slinko, How large should a coalition be to manipulate an election? Math. Soc. Sci. 47 (3) (2004) 289–293.
[37] T. Walsh, Where are the really hard manipulation problems? The phase transition in manipulating the veto rule, in: Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI, July 2009, pp. 324–329.
[38] T. Walsh, An empirical study of the manipulability of single transferable voting, in: Proceedings of the 19th European Conference on Artificial Intelligence, IOS Press, August 2010, pp. 257–262.
[39] L. Xia, V. Conitzer, Generalized scoring rules and the frequency of coalitional manipulability, in: Proceedings of the 9th ACM Conference on Electronic Commerce, ACM Press, June 2008, pp. 109–118.
[40] L. Xia, V. Conitzer, A sufficient condition for voting rules to be frequently manipulable, in: Proceedings of the 9th ACM Conference on Electronic Commerce, ACM Press, June 2008, pp. 99–108.
[41] L. Xia, V. Conitzer, A. Procaccia, A scheduling approach to coalitional manipulation, in: Proceedings of the 11th ACM Conference on Electronic Commerce, ACM Press, June 2010, pp. 275–284.
[42] M. Zuckerman, A. Procaccia, J. Rosenschein, Algorithms for the coalitional manipulation problem, Artif. Intell. 173 (2) (2009) 392–412.