Solving Hard Control Problems in Voting Systems via Integer Programming

Sergey Polyakovskiy
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia.

Rudolf Berghammer
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
Olshausenstraße 40, 24098 Kiel, Germany.

Frank Neumann
Optimisation and Logistics
School of Computer Science
The University of Adelaide
Adelaide, SA 5005, Australia.

August 27, 2014

Abstract

We address various voting systems and types of control of elections and show how hard control problems can be modeled as integer programs. By means of the off-the-shelf solver CPLEX we then demonstrate that the approach allows to treat the control of larger elections successfully. As a consequence, the hardness of a certain control problem is not a secure protection for the fraudulent falsification of outcomes of elections using this type of manipulation.

1 Introduction

When a group of people with individual preferences has to decide which alternative to choose from a given set of alternatives, an election is frequently applied in such a situation. The voting rule underlying the election can be regarded as an algorithm that computes from the individual preferences of the people (which in this context are called the voters) those alternatives which are accepted as ‘best’ choices by the whole group. Ideally it should be exactly one such alternative,
the winning one. However, there are many different voting rules to determine the winners of elections, each coming with different advantages and drawbacks. If, for example, a voting rule leads to one winner only and that can be computed very efficiently, then this is surely regarded as an advantage. On the other hand, if it is easily possible to effect the result of an election in a fraudulently manner by, say, a certain actor that alters the election structure only but not the individual preferences of the voters, then the underlying voting rule surely is regarded as bad in view of susceptibility to manipulation. So, voting rules not only should be efficient, it also should be hard (ideally even impossible) to manipulate them.

Computational social choice (see [Brandt et al., 2013] for an overview) is a relatively new field that applies techniques of computer science, mainly from algorithmics and complexity theory, to problems from social choice theory. Among the most interesting algorithmic and computational questions of computational social choice in view of elections and voting rules are the efficiency of voting rules and their susceptibility to manipulation. In this paper we are concerned with the second question and investigate a specific kind of manipulation, called control. Here we have exactly the situation described above: Some actor seeks to achieve a desired result by altering the structure of the election only but not the individual preferences of the voters.

In [Bartholdi et al., 1992] a new line of research is initiated that investigates the susceptibility to control by techniques from complexity theory. The goal is to prevent attacks using certain types of control by showing that they lead to NP-hard decision problems. Following [Bartholdi et al., 1992], numerous papers have investigated the complexity of control problems for elections; see e.g., [Conitzer et al., 2007] [Hemaspaandra et al., 2007] [Faliszewski et al., 2008] [Faliszewski et al., 2011], and [Rothe and Schend, 2012]. For many election systems it was shown that certain control problems are hard. Such systems are regarded as secure against this attempt to manipulate. But it is well-known that for computationally hard problems algorithms may exist that work for many cases fast enough. The successful history of SAT solvers presents impressive examples in this regard. In the context of computational social choice in [Conitzer and Sandholm, 2006] a simple manipulation algorithm for elections is presented that works fast and yields for most inputs (according to a suitably chosen probability distribution) the desired result. Another approach to solve hard control problems in practice is presented in [Berghammer et al., 2014] [Berghammer and Schnoor, 2014], and [Berghammer, pear]. It combines relation algebra and the BDD-based computer algebra system RelView and yields algorithms which are correct for all instances. The present paper follows the line initiated in [Gurski and Roos, 2014]. In this paper binary integer programming is used to solve some hard control problems for two closely related voting systems, known as Copeland voting and Llull voting, respectively. To our knowledge, this is the only paper that uses binary integer programming for such tasks. Continuing and expanding [Gurski and Roos, 2014], we show how the hard control problems of the (quite different) voting systems we will introduce in Section 2 can be specified as integer programs and present results of computational ex-
The remainder of the paper is organized as follows. In Section 2 we introduce the notion of a voting system and present the specific voting systems we will consider in this paper. The subsequent Section 3 is devoted to several control problems in voting systems and their computational complexities. How to model the hard types of control for the voting systems of Section 2 as integer programs is shown in Section 4, which constitutes the core of the paper. Herein, we mainly concentrate on a constructive control where the actor seeks to achieve a win for his favourite alternative. Section 5 presents the results of our computational experiments, which we have performed on the test benchmark suite of the Preference Library (see [Mattei and Walsh, 2013]) using one of the well-known off-the-shelf solvers for integer programs. The results of our experiments show that our approach is able to solve all the test instances to optimality quite fast by means of an ordinary personal computer. Moreover, even the hard instances related to larger elections can be handled in reasonable time. Consequently, the hardness of a control problem is not a secure protection for the fraudulent falsification of outcomes of elections using this type of manipulation. Finally, Section 6 contains some concluding remarks and presents topics for future investigations in this area of work.

2 Voting Systems

In the following, we introduce the notions of a voting system and an election as generally used in social choice theory as well as the particular voting systems we will treat in this paper. For more details on voting in social choice theory and additional voting systems see [Tideman, 2007] [Brams and Fishburn, 2007] [Laslier, 2012], and [Brandt et al., 2013].

In social choice theory, a voting system (also called a voting protocol) consists of a finite and non-empty set C of candidates (alternatives, proposals, options), a finite and non-empty set V of voters (players, agents, individuals), the individual preferences (choices, wishes) of the single voters, and a voting rule that aggregates the winners from the individual preferences. In the remainder of the paper we assume that the two sets C and V are given as C = \{c_1, \ldots, c_m\} and V = \{v_1, \ldots, v_n\}, where m, n ∈ \mathbb{N}_{>0} are non-zero natural numbers.

2.1 Approval Voting

A well-known voting system is approval voting which has been introduced in [Brams and Fishburn, 2007]. Presently, it is for example used by several scientific organizations including the Mathematical Association of America and the Institute of Management Science. Here each voter may approve (that is, vote for) as many candidates as he wants and then the candidate with more approvals than all other candidates is defined as the winner. If we model the individual preferences by functions a_v : C → \{0, 1\} such that a_v(c) = 1 if voter v approves candidate c, for all v ∈ V and c ∈ C, then an instance of approval voting (in the
remainder of the paper we call instances of voting systems elections) can be specified formally as a triple \((C, V, (a_v)_{v \in V})\) and a candidate \(c^*\) is then defined as the winner iff
\[
\sum_{v \in V} a_v(c^*) > \sum_{v \in V} a_v(c),
\]
for all \(c \in C \setminus \{c^*\}\). Note, that this specification implies winners to be unique. In case that the inequality (1) holds, candidate \(c^*\) is said strictly to dominate candidate \(c\).

The strict dominance relation \(D\) on the set \(C\), for all \(c, d \in C\) defined by \(c D d\) iff \(\sum_{v \in V} a_v(c) > \sum_{v \in V} a_v(d)\), is asymmetric. But it may not relate each pair of different candidates. As a consequence, a candidate that strictly dominates all other ones (the winner) does not necessarily exist. For this reason, in the literature a variant of approval voting is also investigated, where dominance is weak. Here, candidate \(c^*\) wins the election \((C, V, (a_v)_{v \in V})\) iff for all \(c \in C\) it holds \(\sum_{v \in V} a_v(c^*) \geq \sum_{v \in V} a_v(c)\). The advantage of the variant is that winners always exist, while the disadvantage is that they have not to be unique.

In this paper we concentrate on voting systems with strict dominance and the unique-winner condition. It is not hard to translate all results to the variants with weak dominance and possibly multiple winners.

### 2.2 Scoring-based Voting

Approval voting can be regarded as a specific instance of a scoring-based voting system. Elections of such voting systems are specified as triples \((C, V, (s_v)_{v \in V})\), where each scoring function \(s_v : C \to \mathbb{N}\) specifies how many points gives voter \(v\) to each candidate, for all \(v \in V\). A candidate with strictly more points than all other candidates is defined as the winner. Therefore, candidate \(c^*\) wins the scoring-based election \((C, V, (s_v)_{v \in V})\) iff
\[
\sum_{v \in V} s_v(c^*) > \sum_{v \in V} s_v(c),
\]
for all \(c \in C \setminus \{c^*\}\). The well-known Borda voting system (also called Borda count and developed already in the 18th century by the French mathematician and political scientist J.-C. de Borda) is a specific scoring-based voting system. Here it is demanded that all scoring functions \(s_v\) are injective and fulfill \(s_v(C) = \{0, 1, \ldots, |C| - 1\}\). By means of the points he gives, therefore each single voter ranks all candidates from top to bottom without ties.

### 2.3 Preference-based Voting

The four voting systems we consider in the remainder of this section are preference-based. This means that, as in case of Borda voting, each single voter ranks all candidates from top to bottom without ties. In contrast with Borda voting, however, now the individual preferences of the single voters \(v\) are modeled by means of linear strict orders \(\succ_v\) (that is, asymmetric and transitive relations,
where each pair of different elements is comparable) on the set \( C \). For a given election \((C, V, (>_{v})_{v \in V}\) with a so-called preference profile \((>_{v})_{v \in V}\) the four following preference-based voting systems only differ in their voting rules.

In the Condorcet voting system (named after the 18-century French mathematician and philosopher N. de Condorcet) candidate \( c \) strictly dominates another candidate \( d \) iff the number of voters \( v \) with \( c >_{v} d \) is strictly larger than the number of voters \( v \) with \( d >_{v} c \). As a consequence, candidate \( c^{*} \) wins the Condorcet election \((C, V, (>_{v})_{v \in V}\) iff

\[
|\{v \in V : c^{*} >_{v} c\}| > |\{v \in V : c >_{v} c^{*}\}|,
\]

for all \( c \in C \setminus \{c^{*}\} \). Already N. de Condorcet noted a voting paradox that nowadays is called Condorcet paradox. In our terminology it means that the strict dominance relation of a Condorcet election may contain cycles – even if it relates each pair of different candidates (i.e., is a so-called tournament relation). In such a case it may happen that there exists no winner.

When the plurality voting system is used, the most common voting system in the Anglo-saxon world, then candidate \( c \) strictly dominates another candidate \( d \) iff the number of voters with \( c \) as top preference is strictly larger than the number of voters with \( d \) as top preference. Thus, candidate \( c^{*} \) wins the plurality election \((C, V, (>_{v})_{v \in V}\) iff

\[
|\{v \in V : c^{*} = \max_{v} C\}| > |\{v \in V : c = \max_{v} C\}|,
\]

for all \( c \in C \setminus \{c^{*}\} \). In \((4)\) \( \max_{v} C \) denotes the greatest element of the set \( C \) w.r.t. the linear strict order \( >_{v} \), i.e., that element \( c \in C \) for which \( c >_{v} d \) for all \( d \in C \setminus \{c\} \).

The maximin voting system uses the maximin principle, originally formulated for two player zero-sum games, and also defines the winner by means of the cardinalities of the sets \( \{v \in V : c >_{v} d\} \). If we call \( |\{v \in V : c >_{v} d\}| \) the advantage of candidate \( c \) over candidate \( d \) and define the function

\[
\Phi : C \to \mathbb{N} \\
\Phi(c) = \min\{\{v \in V : c >_{v} d\} : d \in C \setminus \{c\}\}
\]

that yields for each candidate the minimum of all its advantages over all other candidates, then candidate \( c^{*} \) wins the maximin election \((C, V, (>_{v})_{v \in V}\) iff

\[
\Phi(c^{*}) > \Phi(c),
\]

for all \( c \in C \setminus \{c^{*}\} \), that is, if it maximizes the minimum of all advantages over all other candidates and this maximum advantage is unique.

Finally, we consider the Bucklin voting system, named after the American J.W. Bucklin but already proposed by N. de Condorcet. It bases on the candidates’ Bucklin scores, which are computed via the function

\[
\Psi : C \to \mathbb{N}_{>0} \\
\Psi(c) = \min\{k \in \mathbb{N}_{>0} : |\{v \in V : c \in \text{rank}_{v,k}\}| > \frac{n}{2}\}
\]

where \( \text{rank}_{v,k} := \{c \in C : |\{d \in C : d >_{v} c\}| < k\} \) is the set of candidates which are ranked among the top \( k \) positions by voter \( v \), for all \( k \in \mathbb{N}_{>0} \) and \( v \in V \).
In words, the Bucklin score $\Psi(c)$ of candidate $c$ is the least (positive) natural number $k$ such that $c$ is ranked among the top $k$ positions by (strictly) more than half of the voters. By definition then the candidate $c^*$ wins the Bucklin election $(C, V, (>_V)_u \in V)$ iff

$$\Phi(c^*) < \Phi(c),$$

for all $c \in C \setminus \{c^*\}$, that is, if it minimizes the Bucklin scores and this minimum is unique.

3 Control Problems in Voting Systems

Voting systems give rise to computational tasks which ideally should be hard to solve. As we have mentioned in the introduction, in this paper we concentrate on the control of elections. This section introduces the different types of control we will consider in this paper and presents their computational complexities for the voting systems we have introduced in the previous section.

3.1 Constructive and Destructive Control by Removals

If control problems in voting systems are modeled mathematically, then it is assumed that the authority conducting the election (the actor mentioned in the introduction, in the literature on voting systems usually called the chair) knows all individual preferences of the single voters. His goal then is to achieve a specific result by a strategic manipulation of the set of candidates or voters, respectively, but not of the individual preferences of the voters. To hide his mind, the chair furthermore tries to manipulate these sets as little as possible.

The literature on voting systems investigates several types of strategic manipulation. In the present paper we allow only the removal of candidates and of voters, respectively, as the chair’s possibilities. The chair’s knowledge of the individual preferences and the ability to delete candidates (alternatives, proposals) by excuses like ‘to expensive’ or ‘legally not allowed’ and to debar voters from ballots by dirty tricks like mistimed meetings are worst-case assumptions. They are not entirely unreasonable in certain settings, for instance in case of commissions of political institutions, governing boards, and meetings of members of a sports club.

In the present paper we mainly focus on constructive control as investigated at the first time in [Bartholdi et al., 1992] in view of computational complexity. Using this type of control, the chair’s goal is to make his favourite candidate $c^*$ the winner. The counterpart of constructive control is destructive control. Here the chair tries to prevent a specific disliked candidate $c^*$ from being the winner. First results on the computational complexity of this type of control are published in [Hemaspaandra et al., 2007].

Usually, the controls of elections by removals are specified as minimization-problems; see [Bartholdi et al., 1992] and [Hemaspaandra et al., 2007]. If constructive control is done by deleting candidates, then the problem is as follows: Given an election $(C, V, \ldots)$ and the specific candidate $c^*$, compute a minimum
set of candidates $M$ such that $c^* \in C \setminus M$ and the removal of $M$ from $C$ and of its candidates from the individual preferences makes $c^*$ to the winner of the resulting election. To allow for an easier modeling in Section 4 we consider the dual maximization-problem and ask for

(a) a maximum subset $C^* \subseteq C$ such that $c^* \in C^*$ and $c^*$ wins the election $(C^*, V, \ldots)$, in which the original individual preferences are restricted to $C^*$.

It is obvious that from the set $C^*$ then the desired set $M$ is obtained by defining $M := C \setminus C^*$. In an analogous manner we specify the constructive control problem by the removal of voters as maximization-problem for a given election $(C, V, \ldots)$ and the specific candidate $c^*$. Again we ask for

(b) a maximum subset $V^*$ of $V$ such that $c^*$ wins the election $(C, V^*, \ldots)$, in which the original individual preferences are restricted to $V^*$.

Using the specifications (a) and (b), we can immediately obtain specifications of the destructive variants of the controls via removal by replacing the phrase ‘such that $c^*$ wins’ by the phrase ‘such that $c^*$ does not win’.

In Section 2 we have explained by means of approval voting and the Condorcet paradox that in voting systems with strict dominance and the unique-winner condition it may happen that no candidate wins. This implies that also solutions of the control problems not necessarily have to exist. If we later model control problems as integer programs, then the non-existence of a winner of a control problem will be expressed by the fact that the modeling program has no feasible solution.

### 3.2 Complexity of Control by Removals

Given a voting system, some control problems may be easy, some may be hard, and in some cases it may even be impossible for the chair to reach his goal. If a control problem is easy one says that the voting system is vulnerable to this type of control, if it is hard one says that it is resistant to this type of control, and if it is unsolvable one says that it is immune to this type of control. To be easy means that there exists an efficient algorithm that solves the problem to optimality in polynomial time. As usual, to be hard means NP-hardness of the decision problem corresponding to the original optimization-problem, with a bound for the size as an additional input. For example, in case of constructive control by the removal of candidates an instance of the decision problem corresponding to the original minimization-problem consists of an election $(C, V, \ldots)$, the specific candidate $c^*$, and a natural number $k$, and the question is whether it is possible to delete at most $k$ candidates such that $c^*$ wins the resulting election. Finally, to be unsolvable means that it is never possible for the chair to reach his goal by the corresponding control action. In other words, no feasible solution exists for the unsolvable control problem.
Inspired by the seminal paper [Bartholdi et al., 1992], scientists have investigated the hardness of control problems via methods of complexity theory. See e.g., the references given in the introduction or in Section 3.2 of [Brandt et al., 2013]. In the following, we summarize the results concerning the voting systems we have discussed in Section 2 and the four types of control we have considered above.

Approval voting and Condorcet voting are vulnerable to destructive control by deleting voters and to constructive control by deleting candidates, resistant to constructive control by deleting voters, and immune to destructive control by deleting candidates. For the constructive control types these results are proved in [Bartholdi et al., 1992] and for the destructive control types they are proved in [Hemaspaandra et al., 2007]. Since we have introduced approval voting as a specific instance of scoring-based voting, also the latter kind of voting is resistant to constructive control by deleting voters and immune to destructive control by deleting candidates. Plurality voting is vulnerable to constructive as well as destructive control by deleting voters and resistant to constructive as well as destructive control by deleting candidates. Here the proofs for the constructive control types can again be found in [Bartholdi et al., 1992] and those for the destructive control types again in [Hemaspaandra et al., 2007]. In view of the candidates and voters, in maximin voting the situation is exactly contrary to plurality voting. The maximin voting system is vulnerable to constructive as well as destructive control by deleting candidates and resistant to constructive as well as destructive control by deleting voters. Concerning proofs of these facts we refer to [Faliszewski et al., 2011]. Finally, Bucklin voting is vulnerable to destructive control by deleting voters and resistant to the three other types of control, i.e., destructive control by deleting candidates and constructive control by deleting candidates as well as voters. These facts are shown in [Rothe and Schend, 2012].

4 Modeling Control Problems as Integer Programs

Linear programming, usually abbreviated as LP, is a very powerful method for the efficient solution of many optimization-problems in various fields; see [Chvátal, 2007], for example. Using the so-called standard form, a LP problem consists of a linear objective function \( f : \mathbb{R}^n_{\geq 0} \rightarrow \mathbb{R}_{\geq 0} \), where \( f(x_1, \ldots, x_n) = \sum_{j=1}^{n} c_j x_j \), that has to be maximized, linear inequality constraints of the form \( \sum_{j=1}^{n} a_{i,j} x_j \leq b_i \), and the non-negative variables condition saying that all variables \( x_j \) of the program range over the set \( \mathbb{R}_{\geq 0} \) of non-negative real numbers. In many practical applications it is additionally required that the variables range over the set \( \mathbb{N} \) only. Then LP is called integer programming (abbreviated as IP). In contrast to LP the IP problem is NP-hard as shown in [Karp, 1972] even for the special case of binary integer programming (abbreviated as BIP), where only 0 and 1 are allowed as values of the variables. Nevertheless, there are tools
available that also allow to solve larger instances of IP and BIP by techniques like relaxation and branch-and-bound. Examples are the MATLAB LP solver, Xpress, Gurobi and the CPLEX tool of IBM.

In this section we demonstrate how the hard control problems of the voting systems of Section 3 can be specified as integer programming problem and binary integer programming problem, respectively. Doing so, we firstly assume $c^* = c_1$, i.e., that the chair’s goal is to make the first candidate a winner, respectively to avoid win for him. Note, that this precondition gives no restriction and the problem can be formulated concerning any candidate. Secondly, we consider the control problems as maximization-problems as introduced in Section 3 via the specifications (a) and (b). Finally, we mainly restrict us to constructive control since, as we will sketch in Section 4.6, the models proposed here for constructive control can be easily adopted for destructive one.

4.1 Scoring-Based Election Model

Since approval voting is the specific case of scoring-based voting where all scores are zero or one, we start our modeling with the constructive control of scoring-based voting by deleting voters. To this end, we assume a scoring-based election $(C, V, (s_v)_{v \in V})$ to be given. As a first step, we combine the list of scoring functions $(s_v)_{v \in V}$ into a single matrix $A \in \mathbb{N}^{m \times n}$ such that

$$A_{ij} = s_{v_j}(c_i),$$

for all $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$. Next, we represent a solution of the control problem by the binary decision vector $x \in \{0, 1\}^n$ such that $x_j = 1$ iff the voter $v_j$ is allowed to vote, for all $j \in \{1, \ldots, n\}$. Consequently, we arrive at the following binary integer program (SBE) that models the given problem:

$$\text{max} \sum_{j=1}^n x_j$$

subject to

$$\sum_{j=1}^n A_{1j}x_j - \sum_{j=1}^n A_{ij}x_j \geq 1 \quad i \in \{2, \ldots, m\}$$

$$x_j \in \{0, 1\} \quad j \in \{1, \ldots, n\}$$

Because we ask for a maximum subset $V^*$ of $V$ such that $c_1$ wins the scoring-based election $(C, V^*, (s_v)_{v \in V^*})$, (7) describes the objective function as the maximum number of voters allowed to take part in voting. The set of constraints (8) supposes that the candidate $c_1$ is the unique winner of $(C, V^*, (s_v)_{v \in V^*})$ since it collects the largest total amount of scores. Finally, (9) states the variables $x_1, \ldots, x_n$ as binary. Note, that the solution of the proposed (SBE) program is in the dual form respecting the initial problem statement given in Section 2.2. In fact, each value $x_j$, where $j \in \{1, \ldots, n\}$, with $x_j = 0$ of the solution vector $x$ defines the voter $v_j$ to be excluded from the voting process in the standard form.
We have already mentioned that a solution of a control problem not necessarily has to exist and this is expressed by the fact that the modeling integer program has no feasible solution. The sufficient condition of the existence of a feasible solution for the program (SBE) implicates the existence of at least one voter whose preference list quotes $c_1$ as the best candidate. Otherwise, removing any subset of voters cannot lead to a feasible solution where $c_1$ wins.

4.2 Condorcet Election Model

As the second problem, we investigate the constructive control by deleting voters for a given Condorcet election $(C, V, (>_v), v \in V)$. Doing so, we represent the preference profile $(>_v), v \in V$ by a single binary matrix $A \in \{0, 1\}^{(m-1) \times n}$ such that for all $i \in \{1, \ldots, m-1\}$ and $j \in \{1, \ldots, n\}$ it holds

\[
A_{ij} = 1 \iff c_1 > v_j c_{i+1}.
\]

To give an example, if we assume the set $C = \{c_1, c_2, c_3, c_4\}$ of candidates, the set $V = \{v_1, v_2, v_3\}$ of voters, and the preference profile

\[
v_1 : \quad c_1 > v_1 c_2 > v_1 c_3 > v_1 c_4
\]
\[
v_2 : \quad c_1 > v_2 c_3 > v_2 c_2 > v_2 c_4
\]
\[
v_3 : \quad c_4 > v_3 c_3 > v_3 c_2 > v_3 c_1,
\]

then the binary matrix $A \in \{0, 1\}^{3 \times 3}$ that represents this preference profile looks as follows:

\[
A = \begin{pmatrix}
1 & 1 & 0 \\
1 & 1 & 0 \\
1 & 1 & 0
\end{pmatrix}
\]

We establish again the binary decision vector $x \in \{0, 1\}^n$ to represent the solution $V^*$ of the problem, that is, have $x_j = 1$ iff the voter $v_j$ is permitted to vote in $(C, V^*, (>_v), v \in V^*)$, for all $j \in \{1, \ldots, n\}$, and candidate $c_1$ is the unique winner of this election. Thus, we arrive at the following binary integer program (CE):

\[
\text{max } \sum_{j=1}^{n} x_j \tag{10}
\]

s.t. \[\sum_{j=1}^{n} (2A_{ij} - 1) x_j \geq 1 \quad i \in \{1, \ldots, m-1\} \tag{11}\]

\[x_j \in \{0, 1\} \quad j \in \{1, \ldots, n\} \tag{12}\]

Again (10) defines the objective function as the maximum number of voters allowed to take part in voting. The constraints (11) ensure that for all $i \in \{2, \ldots, m\}$ the number of voters who gives a vote to the candidate $c_1$ over the candidate $c_i$ is strictly greater than the number of those who prefers $c_i$ over $c_1$. 

10
Indeed, the form of (11) is equivalent to the form

\[
\sum_{j=1}^{n} A_{ij} x_j > \sum_{j=1}^{n} (1 - A_{ij}) x_j,
\]

which is based on the idea that for any voter \( v_j \) either \( c_1 \) dominates \( c_i \) and \( A_{ij} = 1 \), or \( c_1 \) is dominated by \( c_i \), and therefore \( A_{ij} = 0 \) and the coefficient \( (1 - A_{ij}) \) is 1. In fact, this set of constraints makes \( c_1 \) a unique winner. Finally, (12) again states the variables \( x_1, \ldots, x_n \) as binary.

4.3 Plurality Election Model

As the third problem, we consider the constructive control by deleting candidates for a given plurality election \((C, V, (>_{v})_{v \in V})\). Here we assume the election’s preference profile \((>_{v})_{v \in V}\) to be specified by a list of \( n \) binary matrices \( A^1, \ldots, A^n \in \{0,1\}^{m \times m} \) such that for all \( i, k \in \{1, \ldots, m\} \) and \( j \in \{1, \ldots, n\} \) it holds

\[
A_{jk} = 1 \iff c_i >_{v_j} c_k.
\]

Note, that each \( A^j \) is nothing else than the binary matrix representation of the linear strict order \( >_{v_j} \). Hence, in case of the example from Section 4.2, we get the following binary matrices \( A^1, A^2, A^3 \in \{0,1\}^{4 \times 4} \).

\[
A^1 = \begin{pmatrix}
0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}, \quad
A^2 = \begin{pmatrix}
0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0
\end{pmatrix}, \quad
A^3 = \begin{pmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0
\end{pmatrix}
\]

Since we seek for a maximum subset \( C^* \) of the set \( C \) of candidates such that \( c_1 \) wins the plurality election \((C^*, V, (>^*_v)_{v \in V})\), where \((>^*_v)_{v \in V}\) denotes the restriction of the preference profile \((>_{v})_{v \in V}\) to the set \( C^* \), we represent the solution by a binary decision vector \( x \in \{0,1\}^m \) such that \( x_i = 1 \) iff the candidate \( c_i \) is admitted to take part in the election, for all \( i \in \{1, \ldots, m\} \). Furthermore, we introduce a set of auxiliary binary variables \( z^j_i \), where \( z^j_i = 1 \) holds iff candidate \( c_i \) is of the highest preference for the voter \( v_j \) among the set of candidates chosen by the vector \( x \), for all \( i \in \{1, \ldots, m\} \) and \( j \in \{1, \ldots, n\} \). In such a way, the solution of the problem can be derived by the following binary
integer program (PE):

\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{m} x_i \\
\text{s.t.} & \quad \sum_{k=1}^{m} A_{ki}^j x_k + mz_i^j \leq m \quad i \in \{1, \ldots, m\}, \ j \in \{1, \ldots, n\} \\
& \quad \sum_{k=1}^{m} A_{ki}^j x_k + z_i^j - x_i \geq 0 \quad i \in \{1, \ldots, m\}, \ j \in \{1, \ldots, n\} \\
& \quad \sum_{j=1}^{n} z_i^j \leq nx_i \quad i \in \{2, \ldots, m\} \\
& \quad \sum_{j=1}^{n} z_i^j - \sum_{j=1}^{n} z_i^j \geq 1 \quad i \in \{2, \ldots, m\} \\
& \quad x_i \in \{0, 1\} \quad i \in \{1, \ldots, m\} \\
& \quad z_i^j \in \{0, 1\} \quad i \in \{1, \ldots, m\}, \ j \in \{1, \ldots, n\} 
\end{align*}
\]

Here (13) defines the objective function as the maximum number of candidates admitted to take part in the election. The constraints (14) imply that for all \(i \in \{1, \ldots, m\}\) and \(j \in \{1, \ldots, n\}\) the voter \(v_j\) may give the highest preference to the candidate \(c_i\) over all other candidates selected by the vector \(x\) only if there exists no candidate \(c_k, k \in \{1, \ldots, m\}\), that is preferred by \(v_j\) over \(c_i\). In its turn, (15) imposes that when the candidate \(c_i\) is allowed to contest, either \(c_i\) must be of the highest preference for the voter \(v_j\), or a candidate \(c_k\) preferred over \(c_i\) must exist. The constraints (16) enforce that the candidate \(c_i\) cannot be of the highest preference for any voter if he is not permitted to participate in the election, for all \(i \in \{2, \ldots, m\}\). Thus, the constraint requires for all \(i \in \{2, \ldots, m\}\) and \(j \in \{1, \ldots, n\}\) that \(z_i^j = 0\) if candidate \(c_i\) has been deleted from the election. The next property (17) ensures that candidate \(c_1\) is the candidate with the highest number of voters having \(c_1\) as top priority, and therefore is strictly preferred over other candidates. Implicitly, this set of constraints requires \(x_1 = 1\) and hence candidate \(c_1\) has to be selected in any feasible solution. Finally, (18) and (19) state the variables \(x_1, \ldots, x_m\) and \(z_1^1, \ldots, z_n^m\) as binary.

There always exists a feasible solution for the program (PE), which is guaranteed by possible deleting of all candidates but candidate \(c_1\) as the worst case.

4.4 Maximin Election Model

The fourth problem we consider is the constructive control by deleting voters in elections with the maximin voting rule. Once again, we assume that the preference profile \(>_{V} \in V\) is given by a list of \(n\) binary matrices \(A^1, \ldots, A^n \in \{0, 1\}^{m \times m}\) such that

\[
A_{ik}^j = 1 \iff c_i >_{v_j} c_k,
\]
for all \(i, k \in \{1, \ldots, m\}\) and \(j \in \{1, \ldots, n\}\). Similarly, we use a binary decision vector \(x \in \{0, 1\}^n\) to represent the solution, where \(x_j = 1\) iff the voter \(v_j\) has a permission to vote, for all \(j \in \{1, \ldots, n\}\). Thus, the advantage of candidate \(c_i\) over candidate \(c_k\) can be computed as

\[
adv_{\text{maximin}}(c_i, c_k) = \sum_{j=1}^{n} A_{ik}^j x_j,
\]

for all \(i, k \in \{1, \ldots, m\}\). Now, let the candidate \(c_1\) be the winner of the maximin election and let the positive integer variable \(b\) define the minimum advantage of \(c_1\) over any other candidate. Subsequently, let for all \(i, k \in \{1, \ldots, m\}\) with \(i \neq 1\) and \(i \neq k\) the auxiliary binary variable \(z_{ik}\) denote a situation when

\[
adv_{\text{maximin}}(c_i, c_k) < b.
\]

Then, the solution of the posed problem can be derived by the following integer program (MME):

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} x_j & \quad (20) \\
\text{s.t.} & \quad \sum_{j=1}^{n} A_{ik}^j x_j - n (1 - z_{ik}) \leq b - 1 & \quad i \in \{2, \ldots, m\}, k \in \{1, \ldots, m\}, i \neq k & \quad (21) \\
& \quad \sum_{k=1, k \neq i}^{m} z_{ik} \geq 1 & \quad i \in \{2, \ldots, m\} & \quad (22) \\
& \quad b \leq \sum_{j=1}^{n} A_{ik}^j x_j & \quad k \in \{2, \ldots, m\} & \quad (23) \\
& \quad x_j \in \{0, 1\} & \quad j \in \{1, \ldots, n\} & \quad (24) \\
& \quad z_{ik} \in \{0, 1\} & \quad i \in \{2, \ldots, m\}, k \in \{1, \ldots, m\}, i \neq k & \quad (25) \\
& \quad b \in \mathbb{N}_{>0} & \quad \quad (26)
\end{align*}
\]

Here \((20)\) defines the objective function as the maximum number of voters allowed to take part in voting. Next, the set of constraints \((21)\) strictly bound the advantage values of any candidate but \(c_1\) by \(b\). Its combination with the constraints \((22)\) forces at least the minimal advantage value of each candidate \(c_i\) be bounded by \(b\), for all \(i \in \{2, \ldots, m\}\). In its turn, the constraints \((23)\) bound, and therefore define \(b\) as the minimal advantage of \(c_1\). Finally, \((24)\) and \((25)\) state the variables \(x_1, \ldots, x_n\) and \(z_{21}, \ldots, z_{mm}\) as binary, while \((26)\) states \(b\) as positive integer.

No feasible solution exists in the case of the program (MME) if the candidate \(c_1\) is not a winner concerning at least one of the voters. In fact, if there is no voter \(v_j, j \in \{1, \ldots, n\}\), whose top preference is \(c_1\), then there exists also no set of voters to be deleted to make \(c_1\) the winner of the maximin-based rule election.
4.5 Bucklin Election Model

The last two constructive control problems address elections with the Bucklin voting rule, which is resistant to the constructive control as by deleting voters as well as by deleting candidates.

We start first with the variant of the problem which stipulates deleting voters. Doing so, we assume now that the preference profile \((v)_{v \in V}\) is described by a list of \(n\) binary matrices \(A^1, \ldots, A^n \in \{0, 1\}^{m \times m}\) such that for all \(i, k \in \{1, \ldots, m\}\) and \(j \in \{1, \ldots, n\}\) it holds

\[ A^j_{ik} = 1 \iff \left\{ \begin{array}{l} k \in \{m', \ldots, m\} \\
\text{and the voter } v_j \text{ ranks the } c_i \text{ as } m'-\text{th in his preference list.} \end{array} \right. \]

In other words, the candidate \(c_i\) may obtain at least \(m'\) as the personal score from the voter \(v_j\). In such a way, \(c_i\) gets 1 for each entry in the row \(i\) of the binary matrix \(A^j\) when is the most preferred by \(v_j\) over other candidates.

In case of the example from Section 4.2 we get the following binary matrices \(A^1, \ldots, A^3 \in \{0, 1\}^{4 \times 4}\):

\[
\begin{align*}
A^1 &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
A^2 &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \\
A^3 &= \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}
\end{align*}
\]

Subsequently, to represent the problem’s solution we employ a binary decision vector \(x \in \{0, 1\}^n\), where \(x_j = 1\) iff the voter \(v_j\) participates in voting. Therefore, \(\sum_{j=1}^n A^j_{ik} x_j\) determines the number of voters ready to give the score \(k\) to the candidate \(c_i\). Let, for all \(i, k \in \{1, \ldots, m\}\), the auxiliary binary variable \(z_{ik}\) describe the situation when strictly more than a half of the voters allowed to vote agree to give the score of \(k\), i.e., rank \(c_i\) among the top most \(k\) candidates. Then, the solution of the problem can be obtained by the following
binary integer program (BEV):

\[
\text{max } \sum_{j=1}^{n} x_j \quad (27)
\]

s.t. \[
\sum_{j=1}^{n} \left( \frac{1}{2} - A_{ik}^j \right) x_j + nz_{ik} \leq n - \frac{1}{2} \quad i, k \in \{1, \ldots, m\} \quad (28)
\]

\[
\sum_{j=1}^{n} \left( \frac{1}{2} - A_{ik}^j \right) x_j + nz_{ik} \geq 0 \quad i \in \{2, \ldots, m\}, k \in \{1, \ldots, m\} \quad (29)
\]

\[
\sum_{i=1}^{m} l z_{1i} + (n - k) z_{ik} \leq n - 1 \quad i \in \{2, \ldots, m\}, k \in \{1, \ldots, m\} \quad (30)
\]

\[
\sum_{i=1}^{m} z_{1i} \geq 1 \quad (31)
\]

\[
x_j \in \{0, 1\} \quad j \in \{1, \ldots, n\} \quad (32)
\]

\[
z_{ik} \in \{0, 1\} \quad i \in \{1, \ldots, m\}, k \in \{1, \ldots, m\} \quad (33)
\]

Again (27) defines the objective function as the maximum number of voters allowed to take part in voting. First, (28) implies that the candidate \(c_i\) may earn the score \(k\) iff it obtains votes of strictly more than a half of the participating voters, for all \(i, k \in \{1, \ldots, m\}\). Specifically, the form of (28) is the reduction of the form

\[
\frac{1}{2} + \frac{1}{2} \sum_{j=1}^{n} x_j - \sum_{j=1}^{n} A_{ik}^j x_j + nz_{ik} \leq n,
\]

where the first constant ensures the strictness concerning the half of the total number of votes given to \(c_i\), the second term defines the half of all available votes, the third term calculates the number of participating voters ready to give the score \(k\) to \(c_i\), and finally the combination of the fourth term and the last constant introduces the trigger variable \(z_{ik}\) to handle the corresponding situation. Next, for all \(i \in \{2, \ldots, m\}\) and \(k \in \{1, \ldots, m\}\), the constraints (29) force the indicator \(z_{ik}\) be equal to 1 every time when \(c_i\) earns votes of more than a half of the participating voters. The form of (29) is in fact equivalent to the inequality

\[
\frac{1}{2} \sum_{j=1}^{n} x_j - \sum_{j=1}^{n} A_{ik}^j x_j + nz_{ik} \geq 0.
\]

When each of the variables \(z_{ik}\) reveals that the given threshold is reached, then (30) requires the minimal score obtained by the first candidate \(c_1\) be strictly less than the scores obtained by any other candidates. Indeed, the form of the constraints (30) results from the form

\[
\sum_{i=1}^{m} l z_{1i} + 1 \leq k z_{ik} + n (1 - z_{ik}),
\]
where the left hand side defines the minimal score obtained by \( c_1 \) and ensures the strictness of the inequality of the voting rule, while the right hand side defines the score obtained by \( c_i \) and bounds the former. The next constraints (30) ask at least one variable \( z_{ik} \) related to \( c_1 \) be set to 1. In fact, (31) guarantees that at least the indicator pointing to the least score obtained by \( c_1 \) will trigger. Finally, (32) and (33) state the variables \( x_1, \ldots, x_n \) and \( z_{11}, \ldots, z_{mm} \) as binary.

Similarly to the program (CE), the same necessary condition on the existence of a feasible solution must hold for the program (BEV). It requires the existence of at least one voter who prefers \( c_1 \) over \( c_i \), for each \( i \in \{2, \ldots, m\} \). If this condition fails, then no subset of voters can be deleted in order to guarantee \( c_1 \)'s win. As the sufficient condition for the feasible solution, it is required that at least one voter exists, who gives a top preference to \( c_1 \) over any other candidates.

Herein, we deal with the second variant of the control problem, where a subset of candidates may be deleted from the Bucklin election. Compared with the first case we change the input. Now, we suppose that the preference profile \((>_v)_{v \in V}\) is modeled as in cases of plurality voting and maximin voting, that is, by a list of \( n \) binary matrices \( A^1, \ldots, A^n \in \{0, 1\}^{m \times m} \) such that
\[
A^j_{ik} = 1 \iff c_i >_v c_k,
\]
for all \( i, k \in \{1, \ldots, m\} \), and \( j \in \{1, \ldots, n\} \). A binary decision vector \( x \in \{0, 1\}^m \) is used for the solution description, where \( x_i = 1 \) iff the candidate \( c_i \) is admitted to take part in the election, for all \( i \in \{1, \ldots, m\} \).

We construct the model in such a way that for all \( i \in \{1, \ldots, m\} \) and \( j \in \{1, \ldots, n\} \) the candidate \( c_i \) may obtain at least \( m' \) as a personal score from the voter \( v_j \) when exactly \( m' - 1 \) candidates have higher ranks in \( v_j \)'s preference list. To reveal this fact, we use a set of auxiliary binary variables \( y^j_{il} \), where \( j \in \{1, \ldots, n\} \) and \( i, l \in \{1, \ldots, m\} \), such that \( y^j_{il} = 1 \) when the candidate \( c_i \) can get the score of value \( l \) from the voter \( v_j \), i.e., when the number of available candidates preferred by \( v_j \) over \( c_i \) is strictly less than \( l \). Subsequently, we determine the number of voters ready to give the score \( l \) to the candidate \( c_i \) as \( \sum_{j=1}^n y^j_{il} \). Let the further auxiliary binary variables \( z_{il} \) for all \( i, l \in \{1, \ldots, m\} \) denote the situation when strictly more than a half of the voters rank candidate \( c_i \) among the top most \( l \) candidates. Then, the solution of the posed problem

16
can be computed via the following binary integer program (BEC):

\[
\text{max } \sum_{i=1}^{m} x_i \tag{34}
\]

s.t. \[
\sum_{k=1}^{m} A_{ki}^j x_k + (m + 1) y_{il}^j \leq m + l \quad j \in \{1, \ldots, n\}, i, l \in \{1, \ldots, m\} \tag{35}
\]

\[
\sum_{k=1}^{m} A_{ki}^j x_k + my_{il}^j - m x_i \geq l - m \quad j \in \{1, \ldots, n\}, i, l \in \{1, \ldots, m\} \tag{36}
\]

\[
\sum_{j=1}^{n} \sum_{l=1}^{m} y_{il}^j - n m x_i \leq 0 \quad i \in \{2, \ldots, m\} \tag{37}
\]

\[
nz_{il} - \sum_{j=1}^{n} y_{il}^j \leq \frac{n - 1}{2} \quad i, l \in \{1, \ldots, m\} \tag{38}
\]

\[
\sum_{j=1}^{n} y_{il}^j - nz_{il} \leq \frac{n}{2} \quad i \in \{2, \ldots, m\}, l \in \{1, \ldots, m\} \tag{39}
\]

\[
\sum_{q=1}^{m} q z_{1q} + (n - l) z_{il} \leq n - 1 \quad i \in \{2, \ldots, m\}, l \in \{1, \ldots, m\} \tag{40}
\]

\[
\sum_{q=1}^{m} z_{1q} \geq 1 \tag{41}
\]

\[
x_i \in \{0, 1\} \quad j \in \{1, \ldots, m\} \tag{42}
\]

\[
y_{il}^j \in \{0, 1\} \quad j \in \{1, \ldots, n\}, i, l \in \{1, \ldots, m\} \tag{43}
\]

\[
z_{il} \in \{0, 1\} \quad i, l \in \{1, \ldots, m\} \tag{44}
\]

Here (34) defines the objective function as the maximum number of candidates admitted to take part in the election. Each of constraints (35) permits the candidate \( c_i \) to get a personal score of value \( l \) iff the number candidates preferred by the voter \( v_j \) over \( c_i \) is strictly less than \( l \). The form of (35) is the outcome of

\[
\sum_{k=1}^{m} A_{ki}^j x_k + y_{il}^j \leq m \quad j \in \{1, \ldots, n\}, i, l \in \{1, \ldots, m\}
\]

where the first term defines for the voter \( v_j \) the number of candidates which dominate \( c_i \), while the remaining part introduces the trigger variable \( y_{il}^j \). Concurrently, (36) implies that the candidate \( c_i \) gets the personal score \( l \) from the voter \( v_j \) when ranked among the top most \( l \) candidates and it participates at the election. Indeed, the form of (36) is equivalent to

\[
\sum_{k=1}^{m} A_{ki}^j x_k + my_{il}^j + m (1 - x_i) \geq l.
\]
Subsequently, for all $i \in \{2, \ldots, m\}$ the constraints of (37) restrict each candidate $c_i$ to get any personal scores when it is not allowed to contest. For all $i, l \in \{1, \ldots, m\}$ the set of constraints (38) allows the candidate $c_i$ to earn the score $l$ iff it obtains strictly more than a half of all votes. In fact, the form of (38) results from
\[
\frac{1}{2} + \frac{n}{2} - \sum_{j=1}^{n} y_{il} + nz_{il} \leq n,
\]
where the first constant ensures the strictness concerning the half of the total number of votes given to $c_i$, the second term defines the half of all votes, the third term calculates the number of voters ready to give the score $l$ to $c_i$, and finally the combination of the fourth term and the last constant introduces the trigger variable $z_{il}$ to handle the corresponding situation. In its turn, the constraints of (39) force each indicator $z_{il}$ be equal to 1 every time when $c_i$ earns votes of more than a half of voters, for all $i \in \{2, \ldots, m\}$ and all $l \in \{1, \ldots, m\}$. When $z_{il}$ reveals that the given threshold is reached, (40) requires the minimal score obtained by the first candidate $c_1$ be strictly less than the scores obtained by any other candidates. The form of (40) is equivalent to
\[
\sum_{q=1}^{m} qz_{1q} + 1 \leq lz_{il} + n (1 - z_{il}),
\]
where the left hand side defines the minimal score obtained by $c_1$ and ensures the strictness of the inequality of the voting rule, while the right hand side defines the score obtained by the candidate $c_i$ and bounds the former. The constraint (41) implies that $c_1$ must get at least one of scores, thus at least one variable $z_{1q}$ must be set to 1. In fact, it guarantees that at least the indicator pointing to the least score obtained by $c_1$ will trigger. Finally, the sets of constraints (42), (43) and (44) declare $x_1, \ldots, x_m, y_{11}, \ldots, y_{mm}$ and $z_{11}, \ldots, z_{mm}$ as binary.

There always exists a feasible solution for the (BEC) program which is guaranteed by deleting of all possible candidates but $c_1$ as the worst case.

4.6 Transition from Constructive Control to Destructive Control

At the beginning of Section 4 we have promised to sketch how our models proposed for problems of constructive control can be adopted for ones of destructive control. Here, we explain the possible transition by the example of the scoring-based voting. Recall the decisive constraints (8) of the program (SBE). If we combine these constraints into the single formula
\[
\sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{2j} x_j \geq 1 \land \ldots \land \sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{mj} x_j \geq 1,
\]
then the binary decision vector $x \in \{0, 1\}^n$ of the program (SBE) represents a subset $V^*$ of the set of voters $V$ such that candidate $c_1$ wins the scoring-based
election \((C, V^*, (s_v)_{v \in V^*})\) iff (45) holds. As a consequence, \(c_1\) is not the winner of \((C, V^*, (s_v)_{v \in V^*})\) iff the negation of (45) holds, or, equivalently, iff

\[
\sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{2j} x_j \leq 0 \lor \ldots \lor \sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{mj} x_j \leq 0
\]

(46)
is true. Such constraints with disjunctions are frequently called \(k\)-fold alternative constraints, where \(k\) defines the least number of constraints of the set which must be satisfied. In fact, mathematical programs with alternative constraints are no longer linear. But there is a standard technique to transform alternative constraints into a set of equivalent linear constraints; for details, see Chapter 9 of [Bradley et al., 1977]. In our case it uses \(k = 1\) and replaces (46) by

\[
\sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{ij} x_j \leq M(1 - y_i) \quad i \in \{2, \ldots, m\} \quad (47)
\]

\[
\sum_{i=2}^{m} y_i \geq 1 \quad (48)
\]

\[
y_i \in \{0, 1\} \quad i \in \{2, \ldots, m\} \quad (49)
\]

where each of auxiliary binary variables \(y_2, \ldots, y_m\) reflects the satisfaction of the corresponding inequality of (46), and \(M\) is a large constant. In fact, \(y_i = 1\) when the candidate \(c_1\) obtains less or equal number of scores than candidate \(c_i\). The constant \(M\) ensures that for all \(i \in \{2, \ldots, m\}\) the formula \(\sum_{j=1}^{n} A_{1j} x_j - \sum_{j=1}^{n} A_{ij} x_j \leq M(1 - y_i)\) holds if \(y = 1\). Here \(M\) is set as \(M = n \times \max_{i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}} A_{ij}\). The constraint (48) implies that at least one inequality of (46) holds, and therefore there exists at least one candidate with greater or equal scores than the candidate \(c_1\) possesses. In such a way, to obtain the model for the destructive control problem one needs to replace constraints (8) of the (SBE) model by (47) and (48), and add those of (49).

To adopt other proposed models in order to tackle the destructive control problems it is necessary to substitute constraints responsible for the winner’s determination by corresponding \(k\)-fold constraints.

## 5 Computational Investigation

In this section the performance of the integer programs proposed in Section 4 is evaluated in terms of their solution quality and the needed computation time. We have implemented the program code in the programming language JAVA using the CPLEX 12.6 library with default settings such as 1e-6 for the feasibility and optimality tolerances and 1e-5 for the integrality tolerance. The experiments have been carried out on a desktop PC with an Intel Core i7 processor with 2.0 GHz and 8 Gb RAM.

In the following the efficiency of the proposed integer programming election models is evaluated via the PrefLib library assembled by [Mattei and Walsh, 2013].
Despite the voters of each instance of the library are grouped according to the equality of their preference lists, in our experiments we consider each voter separately as a unique unit. Furthermore, we select the first candidate of the list of candidates provided by each instance as the target winner \( c_1 \).

### 5.1 Scoring-based Voting

To evaluate the scoring-based election model from Section 4.1, we adopt the instances of the “Tied Order - Complete List” benchmark suite of the PrefLib library [Mattei and Walsh, 2013]. Within this suite both the relation of equivalence and the strict order relation between the candidates may exist for each of the voters. For every instance of the suite we construct the election as Borda-like election\(^1\), i.e., by assigning the score of value \( m - 1 \) to the most preferred candidates of a voter, \( m - 2 \) to ones at the second place, and so on, while the least score of 0 is given to the candidate at the last place \( m \) in a linearly ordered preference list (without ties) only. The up-to-date version of the benchmark suite consists of 331 instances. The largest instance in terms of the number of voters contains 299,664 voters and 5 candidates, while the largest instance in terms of the number of candidates has 2,819 of them and 4 voters. Exactly 293 of the 331 instances have been solved either to optimality or to optimality respecting the optimality tolerance. For each of the other 38 instances the infeasibility of the solution concerning the first candidate as a winner has been detected. The specific reasons that result in infeasibility are discussed in Section 4.1. The computation time per instance is at most 3.5 seconds, while the median over all instances is 0.016 seconds only. In fact, all the scoring-based election instances from the presented benchmark suite can be easily solved. The total computation time over the whole suite is 25 seconds.

### 5.2 Preference-based Voting

To test the other proposed integer programming election models of Section 4 we employ the instances of the “Strict Orders - Complete List” benchmark suite of the PrefLib library [Mattei and Walsh, 2013]. Within this suite only a strict order relation between each pair of candidates is given for every voter such that the candidates are linearly ordered by each single voter. The suite contains 627 instances in total. The largest instance in terms of the number of voters has 14,081 of them and 3 candidates, while the largest instance in terms of the number of candidates has 242 of them and 5 voters. Table 4 contains the information concerning the computation times required by the CPLEX solver to find the optimal solutions for the instances of the Condorcet, plurality, maximin and Bucklin election models. The whole set of test instances is partitioned into four classes according to the number \( m \) of candidates. Thus, the first class contains instances whose number of candidates \( m \) falls into the range from 1 to 9, while the second, third and fourth classes have the number of candidates

\(^1\)Note, that the elections are not Borda elections since the instances allow ties.
in the ranges 10-99, 100-199, and greater than 200, respectively. The classes correspond to the last four columns of the table one-to-one. The first four rows of the table report the number of candidates \( m \), the number of voters \( n \), the median \( n' \) of the number of voters, and the number of instances \( c \) in each of the classes, respectively. Therefore, most of the instances are rather small in terms of the number of candidates. Specifically, 523 of them are contained in the first class. The remaining rows are grouped and present the minimum, median, average and maximum computation time over each of the classes for every election model we have presented.

Table 1: Running times used by Condorcet, plurality, maximin and Bucklin election models

| Model | \( m \) | \( n \) | \( n' \) | \( c \) |
|-------|-------|-------|-------|------|
| CE    | 1-9   | 4-1081| 21    | 523  |
| CE    | 10-99 | 4-5000| 4     | 77   |
| CE    | 100-199| 4-4   | 4     | 21   |
| CE    | \( \geq 200 \) | 5     | 6     |      |

- **CE**: Condorcet election model
- **CE**: CE
- **CE**: median
- **CE**: average
- **CE**: max

| Model | \( m \) | \( n \) | \( n' \) | \( c \) |
|-------|-------|-------|-------|------|
| PE    | 1-9   | 0.015 | 0.009 | 0.078|
| PE    | 10-99 | 0.015 | 0.006 | 0.047|
| PE    | 100-199| 0.015 | 0.004 | 0.016|
| PE    | \( \geq 200 \) | 0.015 | 0.015 | 0.032|

- **PE**: plurality election model
- **PE**: median
- **PE**: average
- **PE**: max

| Model | \( m \) | \( n \) | \( n' \) | \( c \) |
|-------|-------|-------|-------|------|
| MME   | 1-9   | 0.016 | 0.027 | 0.218|
| MME   | 10-99 | 0.031 | 0.058 | 1.061|
| MME   | 100-199| 0.203 | 0.299 | 2.013|
| MME   | \( \geq 200 \) | 1.099 | 15.452 | 27.518|

- **MME**: maximin election model
- **MME**: median
- **MME**: average
- **MME**: max

| Model | \( m \) | \( n \) | \( n' \) | \( c \) |
|-------|-------|-------|-------|------|
| BEV   | 1-9   | 0.032 | 0.042 | 0.281|
| BEV   | 10-99 | 0.125 | 0.431 | 4.680|
| BEV   | 100-199| 1.763 | 3.810 | 17.800|
| BEV   | \( \geq 200 \) | 9.905 | 50.807 | 93.054|

- **BEV**: Bucklin election model
- **BEV**: median
- **BEV**: average
- **BEV**: max

For the Condorcet election model of Section 4.2, exactly 605 out of 627 instances have been solved to optimality. For the remaining 22 instances infeasibility respecting the target win of the first candidate has been shown. It takes significantly less than a second to solve any of the instances, while the whole suite has been computed in 5.5 seconds.

For the plurality election model of Section 4.3 optimal solutions have been found for all instances of the suite. Here 613 instances, thus almost all of the 627 instances, require less than a second to be solved. However, one instance occurred that needs significantly more computation time comparing to the others. It has the largest \( mn \) product, i.e., the value that strongly correlates with the number of constraints used by the model. Specifically, its \( mn \) product is 50000 and it takes 324 seconds to find the optimal solution. The whole suite can be evaluated in 407 seconds.

For the maximin election model of Section 4.4 exactly 605 out of 627 instances have been solved to optimality, where 619 instances require less than a
second of computation time. For the unsolved 22 instances infeasibility respecting the win of the first candidate has been proved. The maximum computation time over all instances results in 28 seconds. In fact, only the instances of the last class corresponding to the largest $m$ incur considerable computation time. The whole suite has been evaluated in 117 seconds.

For the Bucklin election model with deleting voters of Section 4.5 again for 605 out of 627 instances optimal solutions have been obtained. Exactly 593 instances are computed rather fast; each within one second. Only the last two classes of instances with a larger value of $m$ are time-consuming. The whole suite for this model is solved in 430 sec.

The Bucklin election model with deleting candidates of Section 4.5 has shown to be considerably harder to solve. Despite of the fact that 433 of the 627 instances have been solved within a second, there are hard instances with a maximum computation time of around 8.5 hours. Specifically, all the instances with a number of candidates greater than 100 are rather time-consuming. Even the instances with a small number of candidates but large number of voters require considerably more time than those of other election models. This is mainly because of the increased number of auxiliary variables and constraints used for the problem representation. For all instances the optimal solutions have been found in a total computation time of approximately 28 hours.

6 Conclusion

We have introduced various voting systems and types of control of elections and shown how hard control problems can be modeled as integer programs. Using the solver CPLEX and test suites from the Preference Library we have demonstrate that the approach allows to treat also larger instances successfully. Our experiments show that a proven hardness result of a control type is not a secure protection for the fraudulent falsification of outcomes of elections using this type.

As a future work, we are interested in extensions of our approach to other kinds of voting systems and other kinds of manipulation. Examples for the first are fallback voting and SP-AV, examples for the latter are partition of the voters and bribery. In respect thereof, of great value it may be to investigate related computing methods like SAT-solving, constraint programming, and functional-logic programming.

7 Acknowledgements

Sergey Polyakovskiy and Frank Neumann have been supported through ARC Discovery Project DP130104395.
References

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

[Berghammer, pear] Berghammer, R. (2014, to appear). Relation algebra, relview and plurality voting. In Computer Algebra in Scientific Computation, Lecture Notes in Computer Science. Springer Berlin Heidelberg.

[Berghammer et al., 2014] Berghammer, R., Danilenko, N., and Schnoor, H. (2014). Relation algebra and relview applied to approval voting. In Höfner, P., Jipsen, P., Kahl, W., and Müller, M., editors, Relational and Algebraic Methods in Computer Science, volume 8428 of Lecture Notes in Computer Science, pages 173–199. Springer Berlin Heidelberg.

[Berghammer and Schnoor, 2014] Berghammer, R. and Schnoor, H. (2014). Control of condorcet voting: Complexity and a relation-algebraic approach (extended abstract). In Proceedings of AAMAS-14, pages 1365–1366. IFAA-MAS.

[Bradley et al., 1977] Bradley, S., Hax, A., and Magnati, T. (1977). Applied mathematical programming. Addison-Wesley.

[Brams and Fishburn, 2007] Brams, S. and Fishburn, P. (2007). Approval voting. Second edition, Springer.

[Brandt et al., 2013] Brandt, F., Conitzer, V., and Endriss, U. (2013). Computational social choice. In Weiss, G., editor, Multiagent systems, pages 213–283. Second edition, MIT Press.

[Chvátal, 2007] Chvátal, V. (2007). Linear programming. Freeman and Company.

[Conitzer and Sandholm, 2006] Conitzer, V. and Sandholm, T. (2006). Nonexistence of voting rules that are usually hard to manipulate. In Proceedings of the 21. AAAI, pages 627 – 634. AAAI Press.

[Conitzer et al., 2007] Conitzer, V., Sandholm, T., and Lang, J. (2007). When are elections with few candidates hard to manipulate? Journal of the ACM, 54(14).

[Faliszewski et al., 2011] Faliszewski, P., Hemaspaandra, E., and Hemaspaandra, L. A. (2011). Multimode control attacks on elections. J. Artif. Int. Res., 40(1):305–351.

[Faliszewski et al., 2008] Faliszewski, P., Hemaspaandra, E., and Schnoor, H. (2008). Copeland voting: Ties matter. In Proceedings of AAMAS-08. IFAA-MAS.
[Gurski and Roos, 2014] Gurski, F. and Roos, M. (2014). Binary linear programming solutions and non-approximability for control problems in voting systems. *Discrete Applied Mathematics*, 162(0):391 – 398.

[Hemaspaandra et al., 2007] Hemaspaandra, E., Hemaspaandra, L. A., and Rothe, J. (2007). Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(56):255 – 285.

[Karp, 1972] Karp, R. (1972). Reducibility among combinatorial problems. In Miller, R. E., Thatcher, J. W., and Bohlinger, J. D., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US.

[Laslier, 2012] Laslier, J.-F. (2012). And the loser is... plurality voting. In Felsenthal, D. S. and Machover, M., editors, *Electoral Systems*, Studies in Choice and Welfare, pages 327–351. Springer Berlin Heidelberg.

[Mattei and Walsh, 2013] Mattei, N. and Walsh, T. (2013). Preflib: A library for preferences http://www.preflib.org. In Perny, P., Pirlot, M., and Tsoukiás, A., editors, *ADT*, volume 8176 of *Lecture Notes in Computer Science*, pages 259–270. Springer.

[Rothe and Schend, 2012] Rothe, J. and Schend, L. (2012). Control complexity in bucklin, fallback, and plurality voting: An experimental approach. In Klasing, R., editor, *Experimental Algorithms*, volume 7276 of *Lecture Notes in Computer Science*, pages 356–368. Springer Berlin Heidelberg.

[Tideman, 2007] Tideman, N. (2007). *Collective decisions and voting*. Ashgate.