Journal of Artificial Intelligence Research 50 (2014) 409-446 Submitted 12/13; published 6/14

A Multivariate Complexity Analysis of Lobbying in Multiple Referenda

Robert Bredereck robert.bredereck@tu-berlin.de
Jiehua Chen jiehua.chen@tu-berlin.de
Sepp Hartung sepp.hartung@tu-berlin.de
Stefan Kratsch stefan.kratsch@tu-berlin.de
Rolf Niedermeier rolf.niedermeier@tu-berlin.de
TU Berlin, Germany

Ondřej Suchý ondrej.suchy@fit.cvut.cz
Czech Technical University in Prague, Czech Republic

Gerhard J. Woeginger gwoegi@win.tue.nl
TU Eindhoven, The Netherlands

Abstract

Assume that each of n voters may or may not approve each of m issues. If an agent (the lobby) may influence up to k voters, then the central question of the NP-hard Lobbying problem is whether the lobby can choose the voters to be influenced so that as a result each issue gets a majority of approvals. This problem can be modeled as a simple matrix modification problem: Can one replace k rows of a binary $n \times m$-matrix by k all-1 rows such that each column in the resulting matrix has a majority of 1s? Significantly extending on previous work that showed parameterized intractability (W[2]-completeness) with respect to the number k of modified rows, we study how natural parameters such as n, m, k, or the "maximum number of 1s missing for any column to have a majority of 1s" (referred to as "gap value g") govern the computational complexity of Lobbying. Among other results, we prove that Lobbying is fixed-parameter tractable for parameter m and provide a greedy logarithmic-factor approximation algorithm which solves Lobbying even optimally if $m \leq 4$. We also show empirically that this greedy algorithm performs well on general instances. As a further key result, we prove that Lobbying is LOGSNP-complete for constant values $g \geq 1$, thus providing a first natural complete problem from voting for this complexity class of limited nondeterminism.

1. Introduction

Campaign management comprises all sorts of activities for influencing the outcome of an election, including well-known scenarios such as bribery (Faliszewski, Hemaspaandra, & Hemaspaandra, 2009; Dorn & Schlotter, 2012; Schlotter, Elkind, & Faliszewski, 2011; Elkind, Faliszewski, & Slinko, 2012) and control (Bartholdi III, Tovey, & Trick, 1992; Elkind, Faliszewski, & Slinko, 2011; Erdélyi, Piras, & Rothe, 2011). While these works relate to campaigning in case of classical voting scenarios where one typically wants to make a specific candidate win or to prevent him from winning, Christian, Fellows, Rosamond, and Slinko (2007) introduced the scenario of lobbying in multiple referenda. Intuitively, the point here is that there are n voters, each of them providing a yes- or no-answer to each of m issues. In other words, we have a referendum on m issues the voters can decide on. Naturally, before the referendum is held, campaigns will be run by various parties and interest groups to influence the outcome of the referendum. Assuming complete knowledge about the current voter opinions and assuming the extreme scenario that an external agent–the "lobby"–gains complete control over specific voters, Christian et al. modeled this very basic scenario as 0/1-matrix modification problem, where the rows represent voters, the columns represent issues to vote on with yes (1) or no (0), and the lobby goal is represented by a 0/1-vector.

Lobbying
Input: A matrix $A \in \{0,1\}^{n \times m}$ and an integer $k \geq 0$.
Question: Can one modify the entries of at most k rows in A such that in the resulting matrix every column has more 1s than 0s?

In our context, *modifying* a row means to simply flip all 0s to 1s. Hence, modifying a minimum number of rows can be interpreted as the lobby influencing a minimum number of voters to reach a certain goal. In difference to Christian et al., we assume that the desired outcome of each column is 1 instead of providing a goal vector of the lobby. That is, 1 corresponds to agreement with the lobby goal and 0 corresponds to disagreement. Clearly, by appropriately flipping all entries of a column this can be always ensured. Furthermore, we assume that any column with a majority of 1s is removed from the input matrix. The following example, which is an extract from the real-world data from Section 6.2, illustrates our model.

Example 1. Consider the following four issues and voting behavior of the five faction leaders extracted from the recorded votes of the German parliament in 2013. (See Section 6.2 for details about the full data set.)

Selected issues:
1. Water access is a human right.
2. Forbid the Nationalist Party.
3. Financial help for Ireland.
4. Financial help for Cyprus.

	1	2	3	4
Brüderle	No	No	Yes	Yes
Gysi	Yes	Yes	No	No
Kauder	No	No	Yes	Yes
Künast	Yes	Yes	Yes	Yes
Steinmeier	No	Yes	Yes	Yes

Assume that the lobby wants to approve the first two issues and disapprove the last two issues. Then, the matrix translates into the following binary matrix.

$$\begin{matrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

The second column can be removed because the majority of voters already agree with the lobby. Modifying the first and third row yields a solution.

Lobbying is NP-complete (Christian et al., 2007). Moreover, in the setting of parameterized complexity analysis (Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006), Christian et al. showed that it is W[2]-complete with respect to the parameter "number k of rows to modify", that is, even if only a small number of voters shall be influenced the problem is computationally intractable. In this work, we provide a significantly more refined view on the parameterized and multivariate computational complexity (Fellows, Jansen, & Rosamond, 2013; Niedermeier, 2010) of Lobbying. To this end, we identify the following parameters naturally occurring in Lobbying and analyze their influence on its computational complexity. The studied parameters are:

- n: number of rows;
- m: number of columns[1];
- k: number of rows to modify;
- $k'{:=}\lceil(n+1)/2\rceil - k$: below-guarantee parameter;[2]
- $g{:=}\max_{j=1}^{m} g_j$: maximum gap value over all columns, where the gap value $g_j{:=}\lceil(n+1)/2\rceil - \sum_{i=1}^{n} A_{i,j}$ is the number of missing 1s to make column j have more 1s than 0s;
- s: maximum number of 1s per row;
- t: maximum number of 0s per row.

1.1 Parameter Choice

The parameters n, m, and k are naturally occurring parameters as they measure the input size and the solution size, respectively. Scenarios with only few voters or issues are clearly interesting and realistic restrictions. Furthermore, also the restriction to instances with small solutions is very natural as a limited budget of the lobby may only allow a (very) small amount of bribery or, more positively, advertisement. Additionally, k' complements the parameter k and seems promising because it measures the distance from a trivial type of yes-instances. Moreover, observe that the maximum gap value g is a lower bound on the number of rows that have to be modified; it can be precomputed in linear time. Furthermore, sets of issues where each single issue needs only a small amount of additional approvals or disapprovals bear good prospects for a lobby with limited budget. Finally, s and t measure the density of the input matrix. In our model the density of the matrix can be seen as the degree of agreement between the voters and the lobby. Hence, it is worth to investigate whether high or low density leads to computational tractability.

1.2 Parameter Relations

There are various relations between the parameters' values above. For instance, columns containing a majority of 1s can be safely removed (also implying positive gap values for all columns). As this implies that the input matrix has at least as many 0s as 1s, it follows that there has to be at least one row where at least half of the entries are 0s. Hence, $t \leq m \leq 2t$. In addition, if the input matrix contains a column only consisting of 0s, then one has to modify $\lceil(n+1)/2\rceil$ rows, implying that the corresponding Lobbying instance is

1. Christian et al. (2007) argued that m seldom exceeds 20.
2. Clearly, lobbying $\lceil(n+1)/2\rceil$ voters, that is, modifying $\lceil(n+1)/2\rceil$ rows in the input matrix yields always the desired solution, making $\lceil(n+1)/2\rceil$ a trivial upper bound for k.

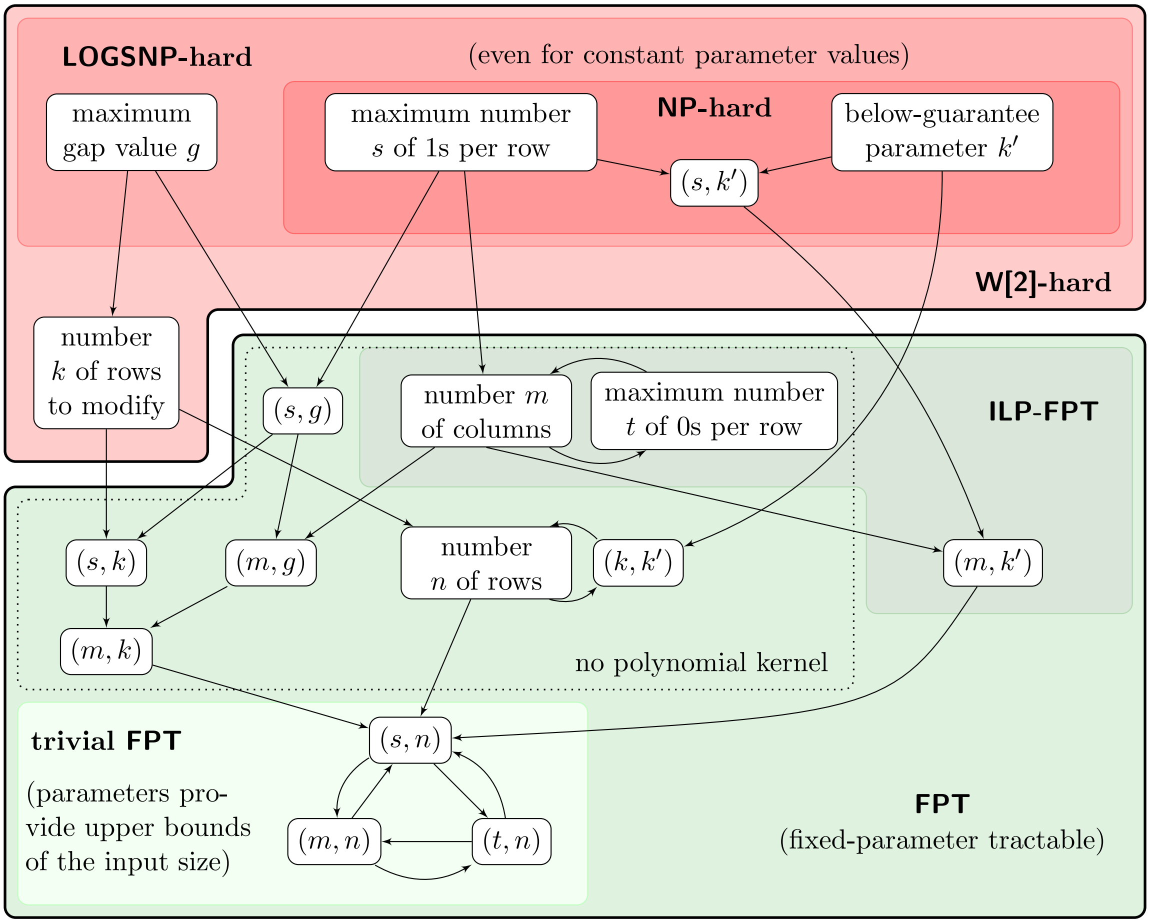

Figure 1: Parameterized complexity of Lobbying and relations between the parameters considered in this paper. An arc from x to y means that there is some function f with $x \leq f(y)$. We omit combined parameters where one component is upper-bounded by a function of the other component, for example (n, k) is omitted, because $k \leq n$. ILP-FPT means that the fixed-parameter tractability bases on a formulation as integer linear program.

a yes-instance if and only if $k \geq \lceil (n+1)/2 \rceil$. Also, we assume $m \leq n \cdot s$ as otherwise there has to be a column consisting only of 0s which is a trivial input instance since we would only have to check whether $k \geq \lceil (n+1)/2 \rceil$. The relations between these parameters and the combinations considered in this paper are illustrated in Figure 1.

Finally, we remark that in this article we focus on the "plain" lobbying in multiple referenda scenarios. We leave considerations of more fine-grained aspects of the lobbying process (such as allowing different forms of modifying the input matrix) as a natural next step for future work.

1.3 Related Work

Our central point of reference is the work by Christian et al. (2007) whose key result is the proof of W[2]-completeness of Lobbying with respect to the number k. Erdélyi, Hemaspaandra, Rothe, and Spakowski (2007) generalized Lobbying to a weighted scenario and showed that an efficient greedy algorithm achieves a logarithmic approximation factor $(1 + \ln(m \cdot \lceil (n+1)/2 \rceil))$ for the weighted case. Moreover, they showed that essentially no better approximation ratio can be proven for their algorithm. Later, models for lobbying in a probabilistic environment were proposed by Binkele-Raible, Erdélyi, Fernau, Goldsmith, Mattei, and Rothe (2014), providing (parameterized) hardness and tractability results; they interpret lobbying as a form of bribery.

As a special case of the *combinatorial reverse auction* problem (Sandholm, Suri, Gilpin, & Levine, 2002), the optimization version of Lobbying is relevant to combinatorial markets in multi-agent systems. In combinatorial reverse auctions, there are different items that a buyer wants to acquire at the lowest cost from the sellers. More precisely, the buyer wants to have an amount of units from each item, $U = (u_1, u_2, \ldots, u_m)$. Each seller i submits how many goods he has for each item, $(\lambda^i_1, \lambda^i_2, \ldots, \lambda^i_m)$ and the price p_i for which he would sell. The goal is to minimize the cost while acquiring the required amount of units of each item. It is easy to see how to translate our lobbying problem into a combinatorial reverse auction problem: Each issue j corresponds to an item j. The buyer's requirement for each item j is the gap value of the corresponding column, $u_j := g_j, \forall j \in \{1, \ldots, m\}$. Each row i corresponds to a seller with offer $\lambda^i_j := 1 - A_{i,j}, \forall j \in \{1, \ldots, m\}$ and price $p_i := 1$.

Lobbying is also closely related to *bribery in judgment aggregation* (Baumeister, Erdélyi, & Rothe, 2011) where the judges submit binary opinions on different propositions and the goal is to bribe as few judges as possible in order to obtain a certain outcome. A Lobbying instance can be formulated as an equivalent instance of the bribery in judgment aggregation problem using a premise-based procedure. More precisely, given a binary matrix A, each propositional variable v_j corresponds to a column j, each judge i corresponds to a row i with the judgment set consisting of those variables v_j satisfying $A_{i,j} = 1$, and the agenda consists of all propositional variables as premises and no conclusions. Now the goal of modifying as few rows as possible in order to have more 1s than 0s in each column is equivalent to bribing as few judges as possible in order to have all possible variables approved by more than half of the judges.

Finally, we refer to survey articles for a discussion on the importance of parameterized complexity results in the context of artificial intelligence (AI) (Gottlob, Scarcello, & Sideri, 2002; Gottlob & Szeider, 2008) and voting (Betzler, Bredereck, Chen, & Niedermeier, 2012).

1.4 Our Contributions

We initiate a systematic parameterized and multivariate complexity analysis for the Lobbying problem. We contribute to the theoretical as well as to the empirical side. See the preliminaries in Section 2 for definitions of parameterized complexity classes and the like. Our complexity results are summarized in Table 1. Let us sketch a few highlights.

- In Section 3, we show that Lobbying remains NP-hard for $s = 3$ and $k' = 1$. Thus, there is no hope for fixed-parameter tractability with respect to the parameters s or k'. Moreover, a special highlight of our work is to show that even if the gap parameter g

<table>
<tr><th></th><th>m</th><th>t</th><th>s</th><th>k'</th><th>g</th><th>k</th><th>n</th></tr>
<tr><td>m</td><td colspan="4" rowspan="2">ILP-FPT (Thm. 9, Cor. 2)
$(2^m)^{2.5\cdot 2^m+o(2^m)}\cdot m\cdot n$</td><td colspan="2" rowspan="2">FPT (Thm. 10)
$(g+1)^m\cdot n^2\cdot m$</td><td rowspan="7">FPT (Prop. 2)
$2^n\cdot m$</td></tr>
<tr><td>t</td></tr>
<tr><td>s</td><td colspan="2"></td><td>$s\geq 3$: NP-c (Thm. 1)
$s\leq 2$: P (Thm. 6)</td><td rowspan="2">NP-c
(Thm. 1)</td><td colspan="2">FPT (Cor. 3)
$(g+1)^{4s}\cdot s\cdot n^2+16^g\cdot m$</td></tr>
<tr><td>k'</td><td colspan="3"></td><td>XP, FPT for const. g
$m^{2g+1}\cdot 2^{2k'}$ (Thm. 11)</td><td></td></tr>
<tr><td>g</td><td colspan="4"></td><td>W[2]-h[a]
LOGSNP-c (Thm. 2)</td><td rowspan="2">W[2]-c[a]</td></tr>
<tr><td>k</td><td colspan="5"></td></tr>
<tr><td>n</td><td colspan="6"></td></tr>
</table>

a. (Christian et al., 2007)

Table 1: Summary of results. For each parameter combination (row by column) the entries indicate whether Lobbying is fixed-parameter tractable (FPT), fixed-parameter tractable based on a formulation as integer linear program (ILP-FPT), polynomial-time solvable for constant parameter values (XP), W[2]-hard (W[2]-h), W[2]-complete (W[2]-c), LOGSNP-complete, meaning completeness for a class of limited nondeterminism lying between P and NP for constant parameter values (LOGSNP-c), or NP-complete even for constant parameter values (NP-c). Entries on the main diagonal represent results with respect to single parameters. Furthermore, for (m,n), (t,n), and (s,n), we have problem kernels of polynomial size because these parameters naturally bound the input size (see Figure 1). For (m,k') we are not aware of kernel size bounds. For all other parameter combinations above, under some reasonable complexity-theoretic assumptions, there cannot be a polynomial-size problem kernel (see Section 4).

is equal to one, Lobbying remains intractable in the sense of being complete for the limited nondeterminism class LOGSNP (Papadimitriou & Yannakakis, 1996) (also see the survey in Goldsmith, Levy, & Mundhenk, 1996). This is of particular interest for at least two reasons. First, it provides a first natural voting problem complete for LOGSNP. Second, this is one of the so far rare examples where this complexity class is used in the context of parameterized complexity analysis.

- In Section 4, we reveal limitations of effective polynomial-time preprocessing for Lobbying, that is, we prove that polynomial-size problem kernels are unlikely to exist.

- In Section 5, we show that Lobbying is fixed-parameter tractable for parameter m by means of describing an ILP formulation with at most 2^m variables. We further provide two efficient algorithms yielding provably optimal results for input matrices with up to four columns. One of these two algorithms is based on a simple greedy strategy and provides a logarithmic approximation ratio for cases with more than four columns.

Furthermore, we develop several fixed-parameter algorithms for various parameter combinations and show that Lobbying is polynomial-time solvable for $s \leq 2$.

- Finally, in Section 6, we experimentally compare our greedy heuristic, with the heuristic in the work of Erdélyi et al. (2007), and an implementation of our ILP formulation providing solutions guaranteed to be optimal. Our empirical results with random data and real-world data indicate that Lobbying can be solved efficiently.

2. Preliminaries

Our aim is to provide a deeper understanding of the computational complexity of the NP-complete Lobbying problem. To this end, we employ classical complexity classes such as P (polynomial time) and NP (nondeterministic polynomial time) (Garey & Johnson, 1979) as well as the class LOGSNP of limited nondeterminism (lying between P and NP) (Papadimitriou & Yannakakis, 1996; Goldsmith et al., 1996) and parameterized complexity classes such as FPT (fixed-parameter tractability), W[2] (second level of the "weft hierarchy" of presumable parameterized intractability), and XP (Downey & Fellows, 2013; Flum & Grohe, 2006; Niedermeier, 2006). Throughout this paper, we denote by log the logarithm to base two.

2.1 LOGSNP

LOGSNP has been introduced by Papadimitriou and Yannakakis (1996) to precisely characterize the computational complexity of certain problems in NP that are neither known to be NP-complete nor known to be solvable in polynomial time. LOGSNP is a subclass of problems in NP which can be decided in polynomial time with an initial phase of $O(\log^2 N)$ nondeterministic steps, where N is the overall input size. However, LOGSNP does not include all problems decidable in polynomial time after the nondeterministic phase since it puts additional restrictions on the computation. Roughly speaking, one is only allowed to use some constant number of "elements" of the "guessed" solution.

It is widely believed that LOGSNP is properly intermediate between P and NP. Problems complete for LOGSNP under polynomial-time reductions include Rich Hypergraph Cover (see Section 3 for the definition) and Log Adjustment (Papadimitriou & Yannakakis, 1996), the latter being of particular importance in artificial intelligence. In Log Adjustment, a boolean expression in conjunctive normal form with r variables and a truth assignment T are given, and the question is whether there is a satisfying truth assignment whose Hamming distance from T is at most $\log r$.

Alternative characterizations of LOGSNP exist (Cai, Chen, Downey, & Fellows, 1997; Flum & Grohe, 2006, Sec. 15.2).

2.2 Fixed-Parameter Tractability and XP

The concept of parameterized complexity was pioneered by Downey and Fellows (2013) (see also further textbooks: Flum & Grohe, 2006; Niedermeier, 2006). The fundamental goal is to find out whether the seemingly unavoidable combinatorial explosion occurring in algorithms to solve NP-hard problems can be confined to certain problem-specific parameters. If such a parameter assumes only small values in applications, then an algorithm with a running time that is exponential exclusively with respect to this parameter may be efficient. For Lobbying, we suggest a number of (potentially small) parameters. A *combined parameter* (which is a vector of parameters) can, for the sake of convenience, be simply seen as the sum of its components.

Formally, a problem is *fixed-parameter tractable* (equivalently, contained in the parameterized complexity class FPT) with respect to a parameter p (equivalently, when parameterized by p) if any instance (I, p) of this problem can be solved in $f(p) \cdot |I|^{O(1)}$ time, where f solely depends on p. If the problem can only be solved in polynomial running time where the degree of the polynomial depends on p (such as $|I|^{f(p)}$), then, for parameter p, the problem is said to lie in the—strictly larger (Downey & Fellows, 2013)—parameterized complexity class XP. Note that containment in XP ensures polynomial-time solvability for a constant parameter p whereas FPT additionally ensures that the degree of the corresponding polynomial is independent of the parameter p.

2.3 Kernelization

Another way of showing fixed-parameter tractability is through kernelization. A *kernelization algorithm* takes as input a problem instance I together with a parameter p and transforms it in polynomial time into an instance I' with parameter p' such that (I, p) is a yes-instance if and only if (I', p') is a yes-instance and there is a function f such that $p' \leq f(p)$ and $|I'| \leq f(p)$. The function f measures the size of the *(problem) kernel* (I', p'). A problem kernel is said to be a *polynomial kernel* if f is polynomially bounded. Note that it is well-known that a decidable problem is fixed-parameter tractable with respect to a parameter if and only if it is "kernelizable" (Cai et al., 1997). The corresponding kernels, however, may have exponential size and it is of particular interest to determine which problems, with respect to which parameter(s), allow for polynomial-size problem kernels (Bodlaender, 2009; Guo & Niedermeier, 2007) since the total running time complexity may vary dependent on the kernel size. Using techniques developed by Bodlaender, Downey, Fellows, and Hermelin (2009) and by Fortnow and Santhanam (2011), Szeider (2011) recently proposed to examine the power of kernelization for several problems in Artificial Intelligence. See Section 4 for more discussion.

2.4 Parameterized Intractability

Downey and Fellows (2013) also introduced a framework of *parameterized intractability*. In this framework, the two basic complexity classes of parameterized intractability are W[1] and W[2], and a problem is shown to be W[1]- or W[2]-hard by providing a *parameterized reduction* from a W[1]- or W[2]-hard problem to the problem in question. A parameterized reduction from a problem Π_1 to another problem Π_2 is a function f computable in FPT time such that there is a function g and the instance (I_2, p_2) produced by f on instance (I_1, p_1) satisfies the following:

- (I_1, p_1) is a yes-instance of problem Π_1 if and only if (I_2, p_2) is a yes-instance of problem Π_2, and
- $p_2 \leq g(p_1)$.

W[1]-complete problems include Clique parameterized by the solution size (Downey & Fellows, 2013) while W[2]-complete problems include Set Cover parameterized by the solution size (see Section 3.1 for the formal definition).

If a problem is shown to be NP-hard even if parameter p is a constant, then it cannot be contained in XP unless P = NP.

3. Intractable Cases

In this section, we examine the worst-case computational complexity of Lobbying. Besides NP-completeness in one very restricted case, we also prove a LOGSNP-completeness result for the case that in each column only one 1-entry is missing to reach a majority of 1s.

3.1 NP-Completeness

For some of the hardness reductions presented in this work, NP-complete variants of the following Set Cover (SC) problem are used.

Set Cover (SC)

Input: A family of sets $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ over a universe $\mathcal{U} = \{u_1, \ldots, u_r\}$ of elements and an integer $h \geq 0$.

Question: Is there a size-h *set cover*, that is, a collection of h sets in $\mathcal{S}$ whose union is $\mathcal{U}$?

Note that even 3-Set Cover (3-SC), where each set from $\mathcal{S}$ is of size at most three, is NP-complete (Karp, 1972). In order to make the reduction work, we need to let h, $|\mathcal{S}|$, and the number of occurrences of every element in the sets in $\mathcal{S}$ fulfill the following properties whose correctness is easy to see:

1. We add (multiple copies of) singletons to the family $\mathcal{S}$ to ensure that $|\mathcal{S}| - 2h + 1 \geq 0$ and to ensure that every element appears in at least h sets in $\mathcal{S}$.

2. We also add a new element u^* to the universe $\mathcal{U}$, add h copies of the singleton $\{u^*\}$ to the family $\mathcal{S}$, and set $h := h + 1$ to ensure that each element appears in at most $|\mathcal{S}| - h$ sets in $\mathcal{S}$.

A common starting point for each of our SC reductions is to transform $\mathcal{S}$ (and $\mathcal{U}$) into a binary matrix in a similar way as Christian et al. (2007) transformed Dominating Set instances into binary matrices. The corresponding $|\mathcal{S}| \times |\mathcal{U}|$-matrix is called SC-*matrix* and is defined as

$$M(\mathcal{S},\mathcal{U}) = (x_{i,j}) \text{ with } x_{i,j} := \begin{cases} 1 & \text{if } u_j \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

Observe that each column $j \in \{1, \ldots, |\mathcal{U}|\}$ has 1s in at most $(|\mathcal{S}| - h)$ rows. We will make use of the SC-matrix in Section 4 and in the following theorem.

An instance of Lobbying with $k \geq \lceil (n+1)/2 \rceil$ (that is, more than half of the rows can be modified) is a yes-instance. Following general ideas of Mahajan and Raman (1999) in the context of "above and below guarantee parameterization", we investigate the complexity of Lobbying in case the maximum number of 1s per row is bounded by a constant and k is slightly below the bound $\lceil (n+1)/2 \rceil$. Such a parameterization is called *below guarantee*.

$$\begin{aligned}
\mathcal{U} &= \{1,\ 2,\ 3,\ 4,\ 5\} \\
\mathcal{S} &= \{\ S_1 = \{2,4\} \\
&\qquad S_2 = \{3,5\} \\
&\qquad S_3 = \{1,2,5\} \\
&\qquad \mathbf{S_4} = \{\mathbf{1},\mathbf{2},\mathbf{3}\} \\
&\qquad \mathbf{S_5} = \{\mathbf{4},\mathbf{5}\}\ \}
\end{aligned}
\qquad\qquad
\begin{array}{r|ccccc|cc|}
 & 1 & 2 & 3 & 4 & 5 & & \\
S_1: & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
S_2: & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
S_3: & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
S_4: & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
S_5: & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}$$

$$h = 2 \qquad\qquad k := |\mathcal{S}| - h = 3$$

Figure 2: Illustration of the reduction from 3-SC to Lobbying. Left: an 3-SC instance; Right: the constructed Lobbying instance. The SC-matrix together with two dummy rows and two dummy columns (inside the dashed polygon) ensure that any set of rows which are not selected by some solution of size three corresponds to a set cover. The 3-SC solution (S_4 and S_5) is highlighted in boldface on the left. The Lobbying solution (modifying the first three rows) is highlighted by gray backgrounds.

Theorem 1. Lobbying *remains* NP*-complete for input matrices where the maximum number s of ones per row is three and for $k' = 1$ of the below guarantee parameter $k' = (\lceil (n+1)/2 \rceil - k)$.*

Proof. Obviously, given a matrix A and $k = \lceil (n+1)/2 \rceil - 1$ rows of A, we can check in polynomial time whether each row of A has at most three 1s and whether modifying the k given rows makes every column have more 1s than 0s. Hence, Lobbying with $s = 3$ and $k' = 1$ remains in NP.

For the NP-hardness result, we describe a polynomial-time reduction from 3-SC to Lobbying where each row contains at most three 1s. The reduction is illustrated with an example in Figure 2. Let $(\mathcal{S}, \mathcal{U}, h)$ denote a 3-SC instance. First, compute the SC-matrix $M(\mathcal{S}, \mathcal{U})$, which we refer to as *original rows* and *original columns* in the following. Second, add $|\mathcal{S}| - 2h + 1$ additional *dummy rows* and $|\mathcal{S}| - 2h + 1$ additional *dummy columns* containing only 0s. Recall that we can assume that $|\mathcal{S}| - 2h + 1 \geq 0$. Finally, for $1 \leq i \leq |\mathcal{S}| - 2h + 1$, change the entry in the ith dummy column of the ith dummy row to 1. To complete the construction we set $k := |\mathcal{S}| - h$.

Each original row has at most three 1s since each set of $\mathcal{S}$ has at most three elements. Each dummy row has exactly one 1. Further, the below guarantee parameter $k' = \lceil (n+1)/2 \rceil - k = \lceil (2|\mathcal{S}| - 2h + 1)/2 \rceil - (|\mathcal{S}| - h) = 1$. The quota for a column to have more 1s than 0s is $|\mathcal{S}| - h + 1$. Since every original column has 1s in at most $|\mathcal{S}| - h$ original rows and no 1s in the dummy rows, no original column has more 1s than 0s in the matrix. More precisely, the jth original column has gap value $|\mathcal{S}| - h + 1 - |\{S_i \mid u_j \in S_i\}|$. Each dummy column has gap value $|\mathcal{S}| - h$. Hence the maximum gap value g is $|\mathcal{S}| - h$.

Obviously, the reduction runs in polynomial time. Now, we show that $(\mathcal{S},\mathcal{U},h)$ is a yes-instance of 3-SC if and only if the constructed matrix can be modified to have more 1s than 0s in every column by changing at most k rows to all-1-rows.

For the "only if" part, suppose that $(\mathcal{S},\mathcal{U},h)$ is a yes-instance for 3-SC. Let $\mathcal{S}' \subseteq \mathcal{S}$ denote a set cover of size h. By modifying each original row i that corresponds to a set $S_i \notin \mathcal{S}'$, every column has more 1s than 0s, that is, at least $(|\mathcal{S}| - h + 1)$ 1s: Each jth dummy column gains $(|\mathcal{S}| - h)$ 1s from the modified original rows and another 1 from the jth dummy row. Each original column gains $(|\mathcal{S}|-h)$ 1s from the modified original rows and another 1 from an unmodified original row, since the sets corresponding to the unmodified original rows form a set cover.

For the "if" part, suppose that the constructed matrix has more 1s than 0s in every column by modifying at most $k = |\mathcal{S}| - h$ rows. First, observe that the number of modified rows is exactly $|\mathcal{S}| - h$ since the maximum gap value is $|\mathcal{S}| - h$. Second, no dummy row is modified: Assume towards a contradiction that the i'th dummy row is modified. Since the i'th dummy column has only one 1 in the matrix, namely, at the i'th dummy row, this column cannot get a majority of $(|\mathcal{S}| - h + 1)$ 1s by the modification of any $|\mathcal{S}| - h$ rows including the i'th dummy row—a contradiction. Third, we show that the sets corresponding to the unmodified original rows form a size-h set cover for $(\mathcal{S},\mathcal{U},h)$. Each original column j' gets exactly $(|\mathcal{S}| - h)$ 1s from the modified original rows and no 1 from any dummy row. To get a majority of $(|\mathcal{S}| - h + 1)$ 1s, column j' must contain another 1 in some unmodified original row. Hence, the sets corresponding to the unmodified rows form a set cover of size k. This shows the correctness of the construction. □

Proposition 1. Lobbying *remains* NP*-complete for every constant integer value* $k' > 1$ *of the below-guarantee parameter* $k' = (\lceil (n+1)/2 \rceil - k)$.

Proof. The NP-containment follows analogously to Theorem 1. To show the NP-hardness for any constant $k' > 1$, we take the SC-matrix $M(\mathcal{S},\mathcal{U})$ and add x additional dummy columns and xk' additional dummy rows where $x := (|\mathcal{S}| - 2h - 1)/k' + 2$. Note that we can add singletons to the family $\mathcal{S}$ to make sure k' is a divisor of $(|\mathcal{S}| - 2h - 1)$. We fill the added entries as follows: For each $j \in \{1, \ldots, |\mathcal{U}|\}$, set the entries of $k' - 1$ arbitrary dummy rows at the original column j to 1. For $1 \leq j \leq x$, set the entry in the jth dummy column of the $[(j-1)k' + 1]$th, $[(j-1)k' + 2]$th, …, $[(j-1)k' + k']$th dummy rows to 1. Set $k := |\mathcal{S}| - h$.

In total, the constructed matrix has $2|\mathcal{S}| - 2h + 2k' - 1$ rows and $|\mathcal{U}| + x$ columns. The quota for a column to have more 1s than 0s is $q = |\mathcal{S}| - h + k'$. Each original column has gap value $q - (|\{S_i \mid u_j \in S_i\}| + k' - 1) = |\mathcal{S}| - h + 1 - (|\{S_i \mid u_j \in S_i\}|)$ and each dummy column has gap value $q - k' = |\mathcal{S}| - h$.

The reduction runs in polynomial time. Now, it remains to show the correctness.

Suppose that $(\mathcal{S},\mathcal{U},h)$ has a set cover $\mathcal{S}' \subseteq \mathcal{S}$ of size h. By modifying each original row i that corresponds to a set $S_j \notin \mathcal{S}'$, every column has more 1s than 0s.

Conversely, suppose that the constructed matrix has more 1s than 0s after modifying at most k rows. Using the analogous reasoning as in the proof for Theorem 1, we can show that exactly k original rows are modified and the original rows which are not modified correspond to the sets which can form a set cover for $(\mathcal{S},\mathcal{U},h)$. □

3.2 LOGSNP-Completeness

In this section, we show that Lobbying for instances with constant maximum gap value $g = 1$ is LOGSNP-complete (Papadimitriou & Yannakakis, 1996) (see also Section 2 for some discussion about the class). This implies that,

- unless LOGSNP = P, Lobbying restricted to instances with constant value g cannot be solved in polynomial time, and hence, Lobbying parameterized by g is not in XP, and
- unless LOGSNP = NP, Lobbying restricted to instances with constant value g is not NP-hard.

Theorem 2. Lobbying *for instances with constant maximum gap value g is in LOGSNP.*

Proof. First of all, we show that if $k \geq g \cdot (\lfloor \log m \rfloor + 1)$, then the Lobbying instance (A, k) is a yes-instance. Recall that there is a row containing 0s in at least half of the columns (see the parameter discussion in Section 1). Modify this row and continue with the columns in which nothing was modified so far. Again, there has to be a row containing 0s in at least half of the columns, which one can modify. We repeat this until each column has been touched at least once, decreasing g by at least one. This takes at most $(\lfloor \log m \rfloor + 1)$ steps. Then, we remove all columns with gap value zero. We repeat this procedure at most g times to end up with an empty matrix, modifying altogether at most $g \cdot (\lfloor \log m \rfloor + 1)$ rows.

It follows from the observations above that any Lobbying instance (A, k) can be solved by modifying at most $g \cdot (\lfloor \log m \rfloor + 1)$ rows (independent of the value of k). Since each row can be identified by $O(\log n)$ bits, there is a certificate for yes-instances of the problem that uses only $O(\log^2(n + m))$ many bits. This implies that the Lobbying problem belongs to problems decidable in polynomial time after $O(\log^2(n + m))$ non-deterministic steps. This already indicates that Lobbying with constant maximum gap lies somewhere between P and NP. To show the containment of LOGSNP, we reduce Lobbying with constant maximum gap to the LOGSNP-complete Rich Hypergraph Cover problem (Papadimitriou & Yannakakis, 1996) in polynomial time.

Rich Hypergraph Cover (RHC)

Input: A family of sets $\mathcal{S} = \{S_1, \ldots, S_\ell\}$ over a universe $\mathcal{U} = \{u_1, \ldots, u_r\}$ of an even number r of elements, where $|S_j| \geq r/2$ for all S_j, and an integer $h \geq 0$.

Question: Is there a subset $\mathcal{U}' \subseteq \mathcal{U}$ of size at most h that has a non-empty intersection with every set $S_j \in \mathcal{S}$?

Let $(A \in \{0, 1\}^{n \times m}, k)$ be an input instance to Lobbying with constant maximum gap value g. First, for each column $j \in \{1, \ldots, m\}$, let Z_j be the set of rows that have 0s at column j and let g_j be the gap value of column j, that is, the number of missing 1s to make column j have more 1s than 0s. We assume that (A, k) is non-trivial, that is, no column consists of only 0s and all gap values are positive. Now, consider all possible size-$(|Z_j| - g_j + 1)$ subsets of row set Z_j. If there is a set X of rows which intersects each of these subsets, then modifying all rows in X make column j have more 1s than 0s. We exploit this observation and construct an instance of RHC from $(A \in \{0, 1\}^{n \times m}, k)$.

For each row $i \in \{1, \ldots, n\}$, add an element u_i to the universe $\mathcal{U}$. For each column $j \in \{1, \ldots, m\}$, add to the family $\mathcal{S}$ all possible size-$(|Z_j| - g_j + 1)$ subsets of elements in set $\{u_i \mid i \in Z_j\}$. We say that all these newly added subsets correspond to column j. Let the solution size h equal k. This completes the reduction.

Since g is constant and at most $m \cdot n^g$ different subsets are added to $\mathcal{S}$, the reduction runs in polynomial time. It remains to show the correctness, that is, (A, k) is a yes-instance to Lobbying if and only if the constructed instance $(\mathcal{S}, \mathcal{U}, h)$ is a yes-instance to Rich Hypergraph Cover.

For the "only if" part, assume that (A, k) is a yes-instance. Let R' be a set of rows of size at most k such that modifying them makes every column have more 1s than 0s. Let $\mathcal{U}' = \{u_i \mid i \in R'\}$ be the set of corresponding elements. We show that $\mathcal{U}'$ intersects with every set $S \in \mathcal{S}$. Suppose for the sake of contradiction that there is a set $S^* \in \mathcal{S}$ with $S^* \cap \mathcal{U}' = \emptyset$. By construction of the subsets in $\mathcal{S}$, let set S^* correspond to column j and, hence, $|S^*| = |Z_j| - g_j + 1$. Then R' can contain at most $(g_j - 1)$ rows from Z_j which makes column j not have more 1s than 0s—a contradiction.

For the "if" part, assume that $(\mathcal{S}, \mathcal{U}, h)$ is a yes-instance. Let $\mathcal{U}'$ be a set of elements of size at most h which intersects every set $S \in \mathcal{S}$. Let $R' = \{i \mid u_i \in U'\}$ be the set of corresponding rows. We show that modifying all rows in R' results in a matrix where every column has more 1s than 0s. Suppose for the sake of contradiction that there is a column j which has at most as many 1s as 0s after modifying all rows in R'. This means that $|Z_j \cap R'| \leq g_j - 1$, and hence $Z_j \setminus R'$ contains at least $|Z_j| - g_j + 1$ rows. Thus, it is possible to find a size-$(|Z_j| - g_j + 1)$ set of elements in $\{u_i \mid i \in Z_j \setminus R'\} \subset Z_j$ which does not intersect with $\mathcal{U}'$—a contradiction. □

As a consequence of the above proof, we can assume that any non-trivial instance of Lobbying fulfills $k < g \cdot (\lfloor \log m \rfloor + 1)$. If we try all combinations of at most k rows to modify, and for each of them check whether every column has more 1s than 0s, then we obtain an algorithm solving Lobbying in $O(n^{g \cdot (\lceil \log m \rceil + 1)} \cdot m)$ time. Now, suppose that Lobbying with constant g were NP-hard, implying there is a polynomial-time reduction from some NP-complete problem $\mathcal{P}$ to Lobbying with constant g. Then we would have $n = |I|^{O(1)}$ and $m = |I|^{O(1)}$ for the constructed Lobbying instance with $|I|$ being the input size of problem $\mathcal{P}$. Thus, $\mathcal{P}$ would be decidable in $|I|^{O(\log |I|)}$ time, implying the following corollary.

Corollary 1. Lobbying *for instances with constant maximum gap value g is not NP-hard unless all problems in NP can be solved in $|I|^{O(\log |I|)}$ time, where $|I|$ is the size of the input.*

Next, we show that Lobbying is LOGSNP-hard even if each column needs only one additional 1 to reach a majority of 1s, that is, $g = 1$.

Theorem 3. Lobbying *for instances with maximum gap value $g = 1$ is LOGSNP-hard.*

Proof. We reduce in polynomial time from Rich Hypergraph Cover (RHC) to Lobbying with maximum gap value $g = 1$. The definition of RHC is already given in the proof of Theorem 2.

Consider an arbitrary RHC instance $(\mathcal{S}, \mathcal{U}, h)$ with $|\mathcal{S}| = \ell$ and $|\mathcal{U}| = r$. As already mentioned by Papadimitriou and Yannakakis (1996), we can assume that $h \leq \lfloor \log \ell \rfloor + 1$ as otherwise $(\mathcal{S},\mathcal{U},h)$ is a trivial yes-instance: Since each set in $\mathcal{S}$ contains at least half of the elements from $\mathcal{U}$, there should be one element which appears in at least half of the sets in $\mathcal{S}$. Add to the solution an element which appears in at least half of the subsets and delete all sets in $\mathcal{S}$ which contain this element, and repeat. Now, let $w := \max(\lceil \log \ell \rceil + 2, \lceil \log r \rceil)$. This implies $r/2 \leq 2^{w-1}$, $h \leq w-1$ and $2^w \leq 8\ell + 2r$, that is, 2^w is upper-bounded by a polynomial in ℓ and r. Furthermore, let $\vec{v}_i$ with $1 \leq i \leq 2^w - 1$ be an enumeration of all 0/1-vectors of length w with the exception of the all-zero vector. For every such vector $\vec{v}_i$, let $\mathcal{V}(i)$ denote the set of all $\vec{v}_{i'}$ $(1 \leq i' \leq 2^w - 1)$ for which the inner product $\vec{v}_i \cdot \vec{v}_{i'}$ is odd; it is easily seen that $|\mathcal{V}(i)| = 2^{w-1}$.

Now, we are ready to construct an instance (A, k) for Lobbying. Add to matrix A one *element row* i for each element $u_i \in \mathcal{U}$ and one so-called *dummy row* i' for each 0/1-vector $\vec{v}_{i'}$ $(1 \leq i' \leq 2^w - 1)$ of length w except for the all-zero vector. The total number n of rows in A is $r + 2^w - 1$, which is odd, because r is even. The quota for a column to have more 1s than 0s is $q = r/2 + 2^{w-1}$. Let matrix A have a total of $m = \ell \cdot (2^w - 1)$ columns, one column $c_{(j,z)}$ for each pair (j, z) with $1 \leq j \leq \ell$ and $1 \leq z \leq 2^w - 1$. We fill the entries of A as follows: let column $c_{(j,z)}$ have 0 at element row i if the corresponding set $S_j \in \mathcal{S}$ contains element u_i, and 1 otherwise. Further, let column $c_{(j,z)}$ have 0s at $q - |S_j|$ arbitrarily chosen dummy rows i' with $\vec{v}_{i'} \in \mathcal{V}(z)$ (that is, the inner product $\vec{v}_{i'} \cdot \vec{v}_z$ is odd), and have 1s in the remaining dummy rows. Note that our choice of w implies $|S_j| \leq q$ and $0 \leq q - |S_j| = r/2 + 2^{w-1} - |S_j| \leq 2^{w-1}$ such that there are enough dummy rows i' with $\vec{v}_{i'} \in \mathcal{V}(z)$. In this way, we fix the entries of matrix A such that each column has exactly $q - 1$ ones and q zeros, which means that the maximum gap value g equals one.

Finally, set the parameter k to be h. This completes the reduction. Obviously, it runs in polynomial time. It remains to show that the RHC instance $(\mathcal{S},\mathcal{U},h)$ is a yes-instance if and only if the constructed Lobbying instance (A, k) is a yes-instance.

For the "only if" part, assume that the RHC instance $(\mathcal{S},\mathcal{U},h)$ is a yes-instance, which is certified by a subset $\mathcal{U}'$ of size at most $h = k$. By construction of matrix A, modifying the element rows corresponding the elements in $\mathcal{U}'$ makes every column $c_{(j,z)}$ gain at least one additional 1.

For the "if" part, assume that modifying at most k rows in matrix A results in a matrix where each column has more 1s than 0s. Let R' be the set of modified element rows, and define $D = \{d_1, d_2, \ldots, d_y\}$ so that $d_1, \ldots, d_y$ are the modified dummy rows. Thus, $y \leq k = h \leq w - 1$. Let $U' = \{u_i \mid i \in R'\}$ be the set of elements that correspond to the element rows from R'. For the sake of contradiction, suppose that there is a set $S_j \in \mathcal{S}$ which does not intersect with U'. Then, for every $1 \leq z \leq 2^w - 1$, column $c_{(j,z)}$ has 1s in all element rows from R'. We show that there even is a column $c_{(j,z)}$ which has 1s at every dummy row from D. In order to find this column, we first observe that by standard linear algebra there exists a vector $\vec{v}_z$ $(1 \leq z \leq 2^w - 1)$ that simultaneously satisfies all equations $\vec{v}_{d_i} \cdot \vec{v}_z = 0$ over the finite field of size two for $1 \leq i \leq y$. (The equation system is homogeneous, the dimension of the underlying space is w, and there are only $y \leq w - 1$ constraining equations.) Then, by definition, none of the vectors $\vec{v}_{d_1}, \ldots, \vec{v}_{d_y}$ are contained in the set $\mathcal{V}(z)$. Thus, column $c_{(j,z)}$ has 1s in all dummy rows $d_1, \ldots, d_y$—a contradiction since $c_{(j,z)}$ does not gain an additional 1 after modifying rows from $R' \cup D$. $\square$

4. Limits of Preprocessing

Efficient preprocessing and data reduction techniques are of key importance when trying to (exactly) solve NP-hard problems (Guo & Niedermeier, 2007); we also refer to Szeider (2011) for a general account on showing limits of preprocessing for AI problems. In this section, we prove that for almost all fixed-parameter tractability results stated in Table 1, Lobbying does not admit, under a reasonable complexity-theoretic assumption, an efficient preprocessing algorithm formalized as a *polynomial kernelization* in parameterized algorithmics. A sufficient assumption for almost all known lower bounds for kernelization is that $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$; it is known that $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ would imply a collapse of the polynomial hierarchy to its third level (Yap, 1983). One way for showing the non-existence of polynomial kernels is to give a so-called *polynomial time and parameter transformation* from an NP-hard problem that is already shown unlikely to admit a polynomial kernel (Bodlaender, Thomassé, & Yeo, 2011). A polynomial time and parameter transformation is a parameterized reduction that is required to be computable in polynomial time and the parameter in the instance where one reduces to is polynomially upper-bounded by the parameter in the instance that one reduces from.

We prove that Lobbying admits neither a polynomial-size kernel with respect to the combined parameter (m, k) nor with respect to n. By simple further observations on the relations between parameters, these two results imply the non-existence of polynomial kernels for most parameters and parameter combinations as listed in Table 1. Recall Figure 1 and the parameter discussion in Section 1 for details on the parameter relations. In some cases Lobbying is NP-hard even if the parameters are constants (see s, k', and (s, k')), in other cases the parameters can be bounded by functions only depending on n (see g, k, k', n, and combinations of them) or the parameters can be bounded by functions only depending on m and k (see t, m, g, k, and combinations of them). For (m, n), (t, n), and (s, n) we have problem kernels of linear ((m, n) and (t, n)) or polynomial ((s, n)) size, because these parameters naturally bound the input size. The question whether there are polynomial kernels remains open for the parameters (m, k') and (t, k'), which are equivalent with respect to this question.

To prove our conditional non-existence result for Lobbying parameterized by (m, k), we employ an "incompressibility result" for Set Cover due to the work of Dom, Lokshtanov, and Saurabh (2009).

Theorem 4. *Unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$,* Lobbying *does not admit a polynomial-size problem kernel with respect to the combined parameter (m, k), that is, the number of columns combined with the number of rows to modify .*

Proof. Dom et al. (2009) showed that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, Set Cover (SC) does not admit a polynomial kernel with respect to the combined parameter $(|\mathcal{U}|, h)$. To show that this result transfers to Lobbying with respect to (m, k) we describe a polynomial time and parameter transformation from SC with respect to $(|\mathcal{U}|, h)$.

Let $(\mathcal{S}, \mathcal{U}, h)$ be an SC-instance. Recall that adding multiple copies of each set to $\mathcal{S}$ does not change the answer to the instance, we can assume without loss of generality that each element occurs in at least h and in at most $|\mathcal{S}| - h$ sets. First, compute the SC-matrix $M(\mathcal{S}, \mathcal{U})$, which we refer to as *original rows* and *original columns* in the following, and invert 0s and 1s. Second, add $|\mathcal{S}|-2h+1$ *dummy rows* containing only 0s. In every original column j, invert $|\mathcal{S}| - |\{S_i : u_j \in S_i\}| - h + 1$ of the 0s from arbitrary dummy rows to 1s. Add one *dummy column* containing 1s in the dummy rows, and 0s elsewhere. Finally, let $k := h$.

Note that the number of 0s in the dummy column is $|\mathcal{S}|$, the number of 0s in any other column is $|\mathcal{S}| - h + 1$, and the total number of rows is $2|\mathcal{S}| - 2h + 1$. Thus, the gap of each original column is exactly one and the gap in the dummy column is h.

We show that $(\mathcal{S}, \mathcal{U}, h)$ is a yes-instance of SET COVER if and only if the constructed matrix can be modified to have more 1s than 0s in every column by changing at most $k = h$ rows to all-1-rows.

For the "only if" part, assume that $(\mathcal{S}, \mathcal{U}, h)$ is a yes-instance. Let $\mathcal{S}' \subseteq \mathcal{S}$ be a size-h set cover. Modifying all (original) rows corresponding to the sets of $\mathcal{S}'$ results in a matrix where each column has more 1s than 0s: Since each of the h modified original rows has a 0 in the dummy column, the dummy column gains h additional 1s. Furthermore, each original row gets at least one additional 1, because $\mathcal{S}'$ is a set cover.

For the "if" part, suppose that the constructed instance is a yes-instance. This means that modifying exactly $k = h$ rows results in a matrix where each column has more 1s than 0s since the maximum gap value is h. Since the gap value in the dummy column is h and no dummy row contains a 0 in the dummy column, a solution cannot modify any dummy row. Since the gap value of each original column is exactly one, the sets corresponding to the modified rows of any LOBBYING solution must form a set cover.

This reduction is a polynomial time and parameter transformation from SC parameterized by $|\mathcal{U}|$ and h to LOBBYING parameterized by m and k: The matrix in the resulting LOBBYING instance has $|\mathcal{U}| + 1$ columns and we ask to modify at most $k = h$ rows. Since LOBBYING is contained in NP and since SC is NP-hard, a polynomial kernel for LOBBYING with respect to (m, k) would imply a polynomial kernel for SC with respect to $(|\mathcal{U}|, h)$ which is not possible unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$ (Dom et al., 2009). $\square$

While LOBBYING is trivially fixed-parameter tractable with respect to the number n of rows, it is unlikely to have a polynomial-size problem kernel under this parameterization.

Theorem 5. *Unless* $\mathsf{NP} \subseteq \mathsf{coNP/poly}$*,* LOBBYING *does not admit a polynomial-size problem kernel with respect to the number* n *of rows (voters).*

Proof. Observe that in the polynomial time and parameter transformation in the proof of Theorem 4 the number of rows in the constructed input matrix for LOBBYING is at most twice the number of sets of the SC instance. Hence, if SC does not admit a polynomial kernel with respect to the number of sets (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$), then LOBBYING also does not admit a polynomial kernel with respect to n (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). Next, we show that, indeed, SC does not admit a polynomial kernel with respect to the number of sets.

SC is strongly related to the HITTING SET (HS) problem, in which, given a family of sets $\mathcal{S}_H$ over a universe $\mathcal{U}_H$ and an integer $h \geq 1$, one is asked to choose a set $U' \subseteq \mathcal{U}_H$ of size at most h such that each set in $\mathcal{S}_H$ has a non-empty intersection with U'. Dom et al. (2009) showed that, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$, HS does not admit a polynomial kernel with respect to $|\mathcal{U}_H|$.

There is a simple polynomial time and parameter transformation from HS parameterized by $|\mathcal{U}_H|$ to SC parameterized by $|\mathcal{S}|$: For each set $S_i \in \mathcal{S}_H$ in the HS instance add an element s_i to the universe set $\mathcal{U}$ of the SC instance. Finally, for each element $e \in \mathcal{U}_H$ of the HS instance add a set to the set family $\mathcal{S}$ of the SC instance that contains the element s_i for each $S_i \in \mathcal{S}_H$ with $e \in S_i$. It is easy to verify that $(\mathcal{S}_H, \mathcal{U}_H, h)$ is a yes-instance of HS if and only if $(\mathcal{S}, \mathcal{U}, h)$ is a yes-instance of SC. □

5. Tractable Cases

In this section, contrasting the intractability results of the previous sections, we demonstrate that Lobbying is efficiently solvable in practically relevant cases. To this end, we study numerous quantities (that is, parameters) and how they influence the computational complexity of Lobbying. For instance, in Section 3 we have seen that Lobbying remains NP-hard for input matrices with at most three 1s per row (Theorem 1)—here we show that it becomes polynomial-time solvable for matrices with at most two 1s per row. Moreover, we show fixed-parameter tractability of Lobbying with respect to the number m of columns (issues), for up to four columns even being linear-time solvable. Finally, we shed light on several further structural restrictions on the input matrix that can make solving Lobbying feasible.

Note that instances with only few rows (voters) are tractable, because already the very naive brute-force approach which simply tries to modify all size-k subsets of rows leads to an algorithm with FPT running time with respect to the parameter number n of rows.

Proposition 2. Lobbying *is solvable in* $O(2^n \cdot m)$ *time.*

5.1 At Most Two Ones per Row

The following result complements Theorem 1, altogether yielding a complexity dichotomy between containment in P and NP-completeness with respect to the maximum number s of 1s per row.

Theorem 6. Lobbying *restricted to input matrices with at most two* 1*s per row, that is,* $s \leq 2$*, is solvable in* $O(nm \log m)$ *time.*

Proof. The idea for the algorithm is to use the special structure of rows that are *not* modified and the fact that each row has at most two 1s. If we may modify $k \geq \lceil (n+1)/2 \rceil$ rows then we can trivially answer yes; otherwise each column requires $b := \lceil (n+1)/2 \rceil - k$ additional 1s outside the k modified rows in order to reach $\lceil (n+1)/2 \rceil$ 1s. Our algorithm will seek a selection of rows such that (1) each column has b 1s inside the selected rows and (2) there are at least k unselected rows left out of which any k can be modified. Modifying any k unselected rows then gives a total of at least $b + k = \lceil (n+1)/2 \rceil$ 1s in each column. Note that any solution with k modified rows requires that each column initially contains at least b 1s; otherwise we can safely answer no.

We will see that the problem of finding the desired selection of rows with b 1s per column comes down to a certain matching problem in an auxiliary graph G, which we will construct next. The graph G gets one vertex for each column, and we add to G an edge for each row which contains 1s in the corresponding two columns (a vertex pair may be connected by multiple edges). A *b-matching* in our graph is a subset of the edges such that each vertex is endpoint of at most b edges in that subset. Our final algorithm will essentially come down to the computation of a maximum cardinality b-matching in G. For clarity, however, let us first show how b-matchings in G and solutions to our lobbying question interact.

First, assume that there exists a solution for the Lobbying instance. Let $\mathcal{R}$ be the set of all rows and let $\mathcal{R}^*$ be a choice of k rows such that modifying these gives each column at least $\lceil (n+1)/2 \rceil$ 1s. (Should $\mathcal{R}^*$ be smaller than k then further rows could be added without harm.) It follows that all other rows, say, $\mathcal{R}' = \mathcal{R} \setminus \mathcal{R}^*$, contribute at least $b = \lceil (n+1)/2 \rceil - k$ 1s in each column. From $\mathcal{R}'$ we will get a large b-matching in G as follows: For each column we arbitrarily mark b rows that have a 1 in this column (so each row is marked at most twice since $s \leq 2$). Let $M \subseteq \mathcal{R}'$ denote the set of marked rows, and let $M_2 \subseteq M$ denote the set of rows that are marked twice. We observe that the size of M is $b \cdot m - |M_2|$ since we marked $b \cdot m$ times and used two marks for each row of M_2. Considering the edges of G which correspond to the elements in M_2 it is easy to see that each vertex is incident to at most b of them (we only marked b times per vertex, and only the twice marked rows are considered). Thus the edges corresponding to M_2 constitute a b-matching in G.

Second, consider the set M_2^* of rows that correspond to the edges of any *maximum* b-matching of G, that is, each vertex is incident to at most b edges; clearly $|M_2^*| \geq |M_2|$. Now greedily add to M_2^* rows such that each column has exactly b 1s in the created set of rows. (This is possible since each column contains at least b 1s; otherwise we answered no at the beginning.) We let M^* denote the obtained set of rows; i.e., those from M_2^* and the added ones. We observe that $|M^*| = b \cdot m - |M_2^*|$, since we add $b \cdot m - 2|M_2^*|$ further rows to M_2^*: We need $b \cdot m$ 1s total, M_2^* contributes $2|M_2^*|$ 1s, and the greedily added rows give one 1 each. It follows that

$$|M^*| = b \cdot m - |M_2^*| \leq b \cdot m - |M_2| = |M| \leq |\mathcal{R}'| = |\mathcal{R}| - k.$$

Thus, there are at least k rows left outside of M^* which can be modified to all-1 rows. Together with the rows of M^* this gives the required number $\lceil (n + 1)/2 \rceil = k + b$ of 1s for each column.

Thus, any solution gives rise to a b-matching and any maximum b-matching leads to a solution (if one exists). Our algorithm therefore begins by computing a maximum b-matching for G. Then, as in the previous paragraph we take the corresponding rows and greedily extend the set to get b 1s in each column. Finally, if this leaves at least k rows unused then modifying any k of them must give a solution (as we just argued). Crucially, if there is a solution, then we showed that this gives a sufficiently large b-matching in G.

The computation of a maximum b-matching is the most costly part; according to Gabow (1983), a maximum b-matching can be computed in $O(nm \log m)$ time (see also Schrijver (2003, Chapter 31) for a more general notion of b-matchings). □

5.2 Very Few and Few Columns

As already mentioned in the introductory section, Christian et al. (2007) pointed out that the number m of issues in multiple referenda rarely exceeds the value 20. This makes m a well-motivated, practically relevant parameter and leads to the question how m influences the computational complexity of Lobbying. In this subsection, we demonstrate how to solve Lobbying efficiently when $m \leq 4$ (by a matching-based and by a greedy algorithm). Moreover, we formulate Lobbying as an integer linear program with a number of variables bounded by a function in m and deduce fixed-parameter tractability from this. These findings are complemented by a "no-polynomial-kernel result" from Section 4 (Theorem 4) and experimental evaluations in Section 6. We start with a matching-based linear-time algorithm for Lobbying.

Theorem 7. *For input matrices with at most four columns* Lobbying *is solvable in linear time.*

Proof. We modify the matching-based algorithm from Theorem 6 to cover all cases with $m \leq 4$. If $m = 1$ or $m = 2$, then no modification is needed. For the cases $m = 3$ and $m = 4$ observe first that the matching-based algorithm can be asked to modify k rows such that there will be b_i 1s in column i for $i \in \{1, \ldots, m\}$.

Now, if $m = 3$ and the input matrix contains c all-1 rows, then we can remove them and execute the matching-based algorithm from Theorem 6 on the resulting matrix (where now every row contains at most two 1s) and ask it to achieve $\lceil (n+1)/2 \rceil - c$ 1s in each column.

For the case $m = 4$ we claim that there is an optimal solution in which at most one row with three 1s is modified. To prove that, consider a solution modifying the minimum number of such rows. Further assume that these rows are the last to be modified and consider the situation just before any row with three 1s is modified. Now suppose without loss of generality that row vector 0111 is modified. Then there is a positive gap in the first column and whenever there is a row with 0 in this column, then it is a 0111 row. Hence, the number of 0s in this column is at least $\lceil (n+1)/2 \rceil$ as it is not yet satisfied. But the number is at most $\lceil (n+1)/2 \rceil$, as otherwise we would have more than $\lceil (n+1)/2 \rceil$ row vectors 0111 and the columns 2, 3, and 4 would be satisfied in the given matrix—a contradiction. Thus, the gap in the first column is 1 and it is enough to modify the row vector 0111 to satisfy the first column. If there was another row with three 1s to be modified, say the row vector 1011, then this one together with all 0111 row vectors would make the gap values of the third and the fourth columns negative—a contradiction.

In summary, we can solve the case $m = 4$ by branching into at most five cases: One 0111 row is modified, one 1011 row is modified, one 1101 row is modified, one 1110 row is modified, and no row with three 1s is modified. Then, we modify the appropriate row and remove all rows containing at least three 1s. We count for each j the number c_j of 1s in column j of the removed rows. Finally, we ask the matching-based algorithm to modify at most $k - 1$ (or at most k, according to the chosen branch) of the remaining rows such that column j contains $\lceil (n+1)/2 \rceil - c_j$ 1-entries. The running time follows from Theorem 6. □

To attack matrices with more than four columns, we next develop a simple greedy strategy for this case. Indeed, it provides optimal results for input matrices with at most four columns. We further show that it is a logarithmic-factor approximation algorithm for an unlimited number of columns. In Section 6, we demonstrate its excellent heuristic properties.

Let $A \in \{0,1\}^{n \times m}$ be an input matrix. Recall that the *gap* g_j of a column $j \in \{1, \ldots, m\}$ is the number of additional 1s this column needs to gain a majority of 1s. Let $G := \{g_j \mid j \in \{1, \ldots, m\}\}$ be the set of different gap values, let $\langle G \rangle$ be the sequence of the elements from G in decreasing order, and let $\langle G \rangle_j$ denote the jth element from $\langle G \rangle$. Further, let gain(R) be the *gain vector* of a row vector R with respect to matrix A, defined as

$$\text{gain}(R) := (z_1, z_2, \dots, z_{|G|}),$$

where z_j denotes the number of zeros in row R in columns with gap value $\langle G \rangle_j$. To compare two gain vectors, we use the lexicographic order $\preceq$. A row R has the *highest* gain if R is the greatest element with respect to $\preceq$, that is, for every row R' in the matrix it holds that $R' \preceq R$.

We now present the algorithm MaxGapZeros which employs a greedy heuristic to decide which rows to modify. The basic idea is to repeatedly choose a row vector with highest gain in A and set it to all 1s until all columns are satisfied. Note that the gap value of each column may change after one row is modified. Hence, a row vector's gain may also change.

Algorithm 1 MaxGapZeros ($A \in \{0,1\}^{n \times m}$)

```
compute the gap values of all columns
while a column with positive gap value exists do
    compute the gain vector of each row vector
    modify an arbitrary row vector with highest gain
    update the gap values for all columns
return modified rows
```

Theorem 8. *Given an input matrix with n rows, m columns, and maximum gap g, MaxGapZeros finds a solution of size at most $\lceil \log m \rceil \cdot g$ in $O(m \cdot n^2)$ time; it even finds an optimal solution for* Lobbying *in $O(n^2)$ time if $m \leq 4$.*

Proof. First, we show the running time bound. Computing the gap values of all columns takes $O(mn)$ time and MaxGapZeros performs at most $\lceil (n+1)/2 \rceil$ iterations. In each iteration computing the gain vectors and finding a row with maximum gain takes $O(mn)$ time, because we can sort the m columns according to their gap values (positive integers of size at most n) in $O(m+n)$ time using counting sort, and compute row-wise the gain vectors while comparing with the current maximum gain vector in $O(mn)$ time. Updating the gap values takes $O(m)$ time. Altogether, MaxGapZeros runs in $O(mn + n(m+n+mn+m)) = O(m \cdot n^2)$ time.

Second, we show the logarithmic approximation factor for $m > 1$. MaxGapZeros takes in each iteration a row with a maximum number of 0s in columns with maximum gap value. We show that, doing this, MaxGapZeros reduces the maximum gap value by one in at most $\lceil \log m \rceil$ iterations.

Let m_g denote the number of the columns with gap value g, where g is the maximum gap value over all columns. If $g > 1$, then there is always a row with strictly more than $(m_g/2)$ 0s in columns with gap value g, because otherwise the matrix restricted to the columns with gap value g would contain at most as many 0s as 1s. MaxGapZeros will select such row because the gain vector of every row with less than $(m_g/2)$ 0s in columns with gap value g is smaller. Hence, the maximum gap is reduced by one in at most $\lceil \log(m_g) \rceil \leq \lceil \log m \rceil$ iterations.

If $g = 1$, it is possible that there is no row with strictly more than $(m_g/2)$ 0s, because the matrix restricted to columns with gap one can have as many 0s as 1s. However, in the ith iteration MaxGapZeros satisfies at least $m_g \cdot 2^{-i}$ columns, since there is always a row that contains at least as many 0s as 1s in columns with gap one. Hence, after $\lceil \log(m_g) \rceil$ iterations, at most one column with gap one survives. Thus, by showing that actually MaxGapZeros satisfies more than $m_g/2$ columns in the first iteration or MaxGapZeros satisfies more than $m_g/4$ columns in the second iteration, we show that in total MaxGapZeros needs $\lceil \log(m_g) \rceil$ iterations to satisfy all columns with gap one. Assume that MaxGapZeros satisfies exactly $m_g/2$ columns in the first iteration. Without loss of generality, let the row that was selected by MaxGapZeros in the first iteration contain 1s in the first $m_g/2$ columns with gap one. Since these columns have gap one, there must be more 0s than 1s in the remaining rows of the matrix restricted to these columns. Thus, there is a row with more than $(m_g/4)$ 0s in columns with gap one that is selected by MaxGapZeros in the second iteration. Hence, all columns with maximum gap one can be satisfied in $\lceil \log(m_g) \rceil \leq \lceil \log m \rceil$ iterations.

Summarizing, MaxGapZeros terminates in at most $\lceil \log m \rceil \cdot g$ iterations. Clearly, every solution of minimum size must contain at least g rows.

Third, we show that MaxGapZeros finds an optimal solution when $m \leq 4$. For input matrices with one or two columns, MaxGapZeros clearly finds a solution of minimum size. In the remainder of the proof, we show that MaxGapZeros also finds a minimum-size solution when the input matrix contains three or four columns. To this end, we analyze the stepwise modification of the input matrix A by MaxGapZeros and compare with the stepwise modification of the input matrix A following a minimum-size solution.

For some multiset of row vectors X, let $A(X)$ denote the matrix resulting from the modification of all row vectors from X in A, that is, replacing the row vectors from X with the same number of all-1-rows. Furthermore, $\langle X \rangle$ denotes a sequence of the row vectors from X and $\langle X \rangle_i$ denotes the ith element in this sequence.

We analyze the stepwise modification process as follows: Let $\langle R_{\text{MGZ}} \rangle$ be the sequence of the row vectors as modified by MaxGapZeros and let $\langle R_{\text{OPT}} \rangle$ be an arbitrary sequence of the row vectors from some minimum size solution R_{OPT}. Furthermore, let A_i be the matrix before the ith row modification from $\langle R_{\text{MGZ}} \rangle$, that is, $A_i := A(\{\langle R_{\text{MGZ}} \rangle_{i'} \mid i' < i\})$ and $g_j(A_i)$ denotes the gap value of the jth column in matrix A_i. Now, for $i = 1$ to $|R_{\text{MGZ}}|$, we compare $\langle R_{\text{MGZ}} \rangle_i$ with $\langle R_{\text{OPT}} \rangle_i$. Note that $\langle R_{\text{OPT}} \rangle_i$ will not run out of the sequence since we will show that $\langle R_{\text{MGZ}} \rangle_{i'} = \langle R_{\text{OPT}} \rangle_{i'}, 1 \leq i' < i$, and it is clear that if $\langle R_{\text{OPT}} \rangle$ and $\langle R_{\text{MGZ}} \rangle$ are identical in the first $i-1$ positions, then either both contain at least i elements or none. When comparing $\langle R_{\text{MGZ}} \rangle_i$ with $\langle R_{\text{OPT}} \rangle_i$, whenever $\langle R_{\text{MGZ}} \rangle_i \neq \langle R_{\text{OPT}} \rangle_i$ we replace or reorder elements in $\langle R_{\text{OPT}} \rangle$ such that afterwards $\langle R_{\text{MGZ}} \rangle_{i'} = \langle R_{\text{OPT}} \rangle_{i'}, 1 \leq i' \leq i$, and, as an invariant, $\langle R_{\text{OPT}} \rangle$ still corresponds to a solution of minimum size. This implies that $A_i = A(\{\langle R_{\text{MGZ}} \rangle_i \mid i' < i\}) = A(\{\langle R_{\text{OPT}} \rangle_i \mid i' < i\})$.

In the following, we assume $\exists i \geq 0$ with $\langle R_{\text{MGZ}} \rangle_{i'} = \langle R_{\text{OPT}} \rangle_{i'}, \forall i' < i$. For two row vectors r_1, r_2 from $\{0,1\}^m$ we write $r_1 \subseteq r_2$ if $r_1[x] = 0 \Rightarrow r_2[x] = 0$, $\forall 1 \leq x \leq m$. It is easy to see that $\nexists i'' \geq i$ with $\langle R_{\text{MGZ}} \rangle_i \subseteq \langle R_{\text{OPT}} \rangle_{i''} \wedge \langle R_{\text{MGZ}} \rangle_i \neq \langle R_{\text{OPT}} \rangle_{i''}$, because MaxGapZeros would have selected $\langle R_{\text{OPT}} \rangle_{i''}$ instead of $\langle R_{\text{MGZ}} \rangle_i$ already in iteration i. For each $1 \leq i \leq |\langle R_{\text{MGZ}} \rangle|$, if $\langle R_{\text{MGZ}} \rangle_i \neq \langle R_{\text{OPT}} \rangle_i$, then at least one of the following four cases occurs. Note that the case that $\langle R_{\text{MGZ}} \rangle_i$ is an all-0s row is subsumed by Case 1.

Furthermore, when there are only three columns Case 2 is also subsumed by Case 1. Cases 2–4 implicitly assume that Case 1 does not apply.

Case 1. $\exists i' \geq i$ with $\langle R_{\mathrm{OPT}}\rangle_{i'} \subseteq \langle R_{\mathrm{MGZ}}\rangle_i$. If $\exists i'' > i$ with $\langle R_{\mathrm{OPT}}\rangle_{i''} = \langle R_{\mathrm{MGZ}}\rangle_i$, then swap the ith and i''th element of $\langle R_{\mathrm{OPT}}\rangle$. Otherwise, replace the i'th element of $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_i$ and swap the ith and i'th element of $\langle R_{\mathrm{OPT}}\rangle$ if $i' \neq i$.

Case 2. $\langle R_{\mathrm{MGZ}}\rangle_i$ contains exactly three 0s. Without loss of generality let $\langle R_{\mathrm{MGZ}}\rangle_i$ have 0s in the first three columns. This implies that the last column's gap value can not be larger than each of the first three columns' gap values, that is, $g_x(A_i) \geq g_4(A_i), x \in \{1, 2, 3\}$. Since Case 1 does not apply, for every position $i' \geq i$ it holds that $\langle R_{\mathrm{OPT}}\rangle_{i'}$ has a 0 in column 4, but at least one 1 in some other column since otherwise our greedy algorithm would have selected this all-0-row $\langle R_{\mathrm{OPT}}\rangle_{i'}$ at step i. Thus, there are more than $g_4(A_i)$ such positions and we can safely replace the ith element in $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_i$.

Case 3. $\langle R_{\mathrm{MGZ}}\rangle_i$ contains exactly two 0s. Without loss of generality, $\langle R_{\mathrm{MGZ}}\rangle_i$ has 0s in the first two columns. Due to the design of MaxGapZeros it holds that the first two columns' gap values are at least as large as the last two columns' gap values, that is, $g_x(A_i) \geq g_y(A_i), x \in \{1, 2\}, y \in \{3, 4\}$. There is no position $i' \geq i$ in $\langle R_{\mathrm{OPT}}\rangle$ such that $\langle R_{\mathrm{OPT}}\rangle_{i'}$ also has 0s in the first two columns, because otherwise $\langle R_{\mathrm{MGZ}}\rangle_i \subseteq \langle R_{\mathrm{OPT}}\rangle_{i'}$, which is not possible since MaxGapZeros would have selected $\langle R_{\mathrm{OPT}}\rangle_{i'}$ instead of $\langle R_{\mathrm{MGZ}}\rangle_i$ already in iteration i. Thus, there are at least $g_1(A_i)$ positions $i_1 \geq i$ with $\langle R_{\mathrm{OPT}}\rangle_{i_1}$ containing a 1 in the second column and at least $g_2(A_i)$ positions $i_2 \geq i$ with $\langle R_{\mathrm{OPT}}\rangle_{i_2}$ containing a 1 in the first column. Moreover, since Case 1 does not apply and, hence, $\langle R_{\mathrm{OPT}}\rangle_{i'} \not\subseteq \langle R_{\mathrm{MGZ}}\rangle_i$ for every $i' > i$, each of the corresponding row vectors has at least two 0s. If $g_1(A_i) = g_2(A_i)$, then by its approximation guarantee MaxGapZeros also needs at most $\lceil \log 4 \rceil \cdot g_1(A_i) = g_1(A_i) + g_2(A_i)$ further rows. Thus, we can safely replace the i'th element of $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_{i'}$ for each $i' \geq i$. If $g_1(A_i) > g_2(A_i)$, then there must be some position $i' \geq i$ such that $\langle R_{\mathrm{OPT}}\rangle_{i'}$ contains a 0 in column $y \in \{3, 4\}$, but the column y is already satisfied in the matrix $A(\{\langle R_{\mathrm{OPT}}\rangle_{i''} \mid i'' < i'\})$. If there is any position $i^* \geq i$ with $\langle R_{\mathrm{OPT}}\rangle_{i^*}$ containing two 0s: one 0 in column y and one 0 in column 1 or 2, then replace the i^*th element of $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_i$ and swap the ith and i^*th element of $\langle R_{\mathrm{OPT}}\rangle$ if $i^* \neq i$. Otherwise, for every position $i^* \geq i$ it holds that if $\langle R_{\mathrm{OPT}}\rangle_{i^*}$ contains a 0 in column y then $\langle R_{\mathrm{OPT}}\rangle_{i^*}$ contains exactly three 0: one 0 in column 3, one 0 in column 4, and one 0 in column 1 or 2. This implies that both column 3 and column 4 do not have positive gap in the matrix $A(\{\langle R_{\mathrm{OPT}}\rangle_{i''} \mid i'' < i'\})$. Thus, replace the $i'$the element of $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_i$ and swap the ith and i'th element of $\langle R_{\mathrm{OPT}}\rangle$ if $i' \neq i$.

Case 4. $\langle R_{\mathrm{MGZ}}\rangle_i$ contains exactly one 0. We show that in this case there are at most two columns with positive gap values in A_i. Let j^* be the only column where $\langle R_{\mathrm{MGZ}}\rangle_i$ has one 0. For each column j, let $R_0(j)$ be the index set of rows containing a 0 in column j. Consider any other column j' with positive gap value. A_i contains no rows with 0s in both j^* and j', that is, $R_0(j^*) \cap R_0(j') = \emptyset$, because otherwise our greedy algorithm would have selected such rows. Thus, the gap values of both j^* and j' must be one, implying that the maximum gap value is one. If the matrix contains at least three columns with maximum gap one, then there must be a row containing 0s in at least two of these columns which should have been selected by our greedy algorithm. Thus, there are at most two columns with positive gap in A_i. Since MaxGapZeros is optimal for at most two columns, we can safely replace the i'th element of $\langle R_{\mathrm{OPT}}\rangle$ by $\langle R_{\mathrm{MGZ}}\rangle_{i'}$ for each $i' \geq i$.

Finally, we obtain a minimum-size solution R_{OPT} with $R_{\mathrm{OPT}} = R_{\mathrm{MGZ}}$. □

There are five-column input matrices where our algorithm may not provide an optimal solution. For example, the optimal solution for the 5×6-matrix

$$\begin{matrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{matrix}$$

contains two rows (row two and three) while our algorithm may output three rows as a possible solution since the first three row vectors all have the highest gain $(1, 2)$. Note that the first column has gap 2 while all other columns have gap 1. If MaxGapZeros decides to modify the first row before the other two rows, then it needs two more rows to satisfy all columns.

Theorems 7 and 8 show that in case of at most four issues Lobbying can be solved very efficiently. On the contrary, parameterized by the number m of columns, we only have a "theoretical fixed-parameter tractability result"; it is based on a famous theorem in mathematical programming of Lenstra (1983) and further improved by Frank and Tardos (1987) and Kannan (1987). Roughly speaking, it says that solving integer linear programs with a number of variables depending solely on parameter p is fixed-parameter tractable with respect to p. However, the (worst-case) upper bound on the running time of the corresponding algorithm is impractical and of classification nature only. Nevertheless, we experimented with the practical usefulness of the subsequent ILP formulation (see Section 6).

Theorem 9. Lobbying *is fixed-parameter tractable with respect to the parameter number m of columns.*

Proof. We describe an integer linear program (ILP) with at most 2^m variables that solves Lobbying.[3] Then, Lobbying is fixed-parameter tractable with respect to m, because any ILP with ρ variables and L input bits can be solved in $O(\rho^{2.5\rho+o(\rho)}L)$ time (Kannan, 1987; Frank & Tardos, 1987).

There are at most 2^m different rows in a binary matrix with m columns. Let $r_1, \ldots, r_l$ be an arbitrary ordering of all pairwise different rows in A, and for all $1 \leq i \leq l$ let $c(r_i)$ be the number of occurrences of r_i. For $1 \leq i \leq l$ and $1 \leq j \leq m$ let $B_j(r_i) = 1$ if the jth column of row r_i has value 0 and, otherwise, $B_j(r_i) = 0$. For $1 \leq i \leq l$ we introduce

3. Dorn and Schlotter (2012) already mentioned that their ILP for the Swap Bribery problem could be adapted to Lobbying.

the integer variable b_i, where $0 \le b_i \le c(r_i)$; the value of b_i indicates how often one has to modify a row of type r_i. The ILP is formulated as follows. Recall that g_j is the number of missing 1s to make column j have more 1s than 0s and k is the number of rows to modify.

$$\sum_{i=1}^{l} b_i \le k, \tag{1}$$

$$0 \le b_i \le c(r_i) \qquad \forall 1 \le i \le l, \tag{2}$$

$$g_j \le \sum_{i=1}^{l} b_i \cdot B_j(r_i) \qquad \forall 1 \le j \le m. \tag{3}$$

Constraint (1) ensures that at most k rows are modified. Constraint (2) ensures that the amount of rows to be modified from each type r_i is available in the input matrix. Constraint (3) ensures that for each column j at least g_j rows with a 0-entry in the jth position are modified. Hence, the ILP provides a solution for Lobbying with at most k modified rows. □

Since $m \le 2t$, fixed-parameter tractability for m (Theorem 9) implies the following.

Corollary 2. Lobbying *is fixed-parameter tractable with respect to the parameter maximum number t of zeros per row.*

5.3 Dynamic Programming for Few Columns and Small Gap

If we use the reduction proof by Christian et al. (2007), then we immediately gain a W[2]-hardness result with respect to the maximum gap value g for Lobbying. Furthermore, as discussed in Section 3.2, Lobbying is LOGSNP-complete even if g is 1. So, under reasonable complexity-theoretic assumptions, it is neither NP-hard nor in XP for constant g. In this section, we prove tractability of Lobbying with respect to parameter g when combining it either with the number m of columns (Theorem 10) or even with the maximum number s of 1s per row (Corollary 3).

Theorem 10. Lobbying *is solvable in $O((g+1)^m \cdot n^2 \cdot m)$ time.*

Proof. Let $A \in \{0,1\}^{n\times m}$ and $k \in \mathbb{N}$ be a Lobbying input instance and let $g_1, g_2, \ldots, g_m$ be the corresponding gap values in A. We solve the problem via Dynamic Programming employing a boolean table $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m]$, where $i \in \{0, \ldots, n\}$, $l \in \{0, \ldots, k\}$, and $\tilde{g}_j \in \{0, \ldots, g_j\}$ for all j. An entry $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m]$ is set to True if it is possible to reduce the gap of column j by at least $\tilde{g}_j$ by modifying exactly l rows in the first i rows of A. Otherwise set the entry to False. Clearly, (A, k) is a yes-instance of Lobbying if and only if $T[n, k, g_1, \ldots, g_m] = \text{True}$.

To initialize the table, we set $T[0, 0, \tilde{g}_1, \ldots, \tilde{g}_m]$ to be True if $\tilde{g}_j = 0$ for all j and False otherwise. To compute an entry $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m]$ we check two cases: First, we set $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m]$ to True if $T[i-1, l, \tilde{g}_1, \ldots, \tilde{g}_m] = \text{True}$ (treating the case where row i is not contained in a solution). Second, we set $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m]$ to True if $T[i-1, l-1, g'_1, \ldots, g'_m] = \text{True}$ where $g'_j = \tilde{g}_j$ if row i has 1 in the column j and $g'_j = \max\{0, \tilde{g}_j - 1\}$ if row i has 0 in the column j. If none of the two cases applies, then we set $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m] = \text{False}$.

Table T has $(g+1)^m \cdot (k+1) \cdot (n+1)$ entries and each table entry can be computed in $O(m)$ time, resulting in an overall running time of $O((g+1)^m \cdot k \cdot n \cdot m)$. □

Finally, we "harvest" a further fixed-parameter tractability result for Lobbying by simply relating parameter values.

Corollary 3. Lobbying *is solvable in* $O((g+1)^{4s} s \cdot n^2 + 16^g \cdot m)$ *time.*

Proof. First, we provide some useful inequality between (functions of) parameter values. Count the number of 1s in the input matrix. Since there are at most s of them in each row, there are at most ns of them in the whole matrix. In addition, there are at least $n/2 - g$ of them in each column and, hence, the total number of 1s is at least $(n/2 - g)m$. It follows that $(n/2 - g)m \leq ns$.

Now, we employ the just derived inequality to deduce fixed-parameter tractability for the combined parameter (g, s). If $g \leq n/4$, then $nm/4 \leq (n/2 - g)m \leq ns$, hence $m \leq 4s$ and we can use the $O((g+1)^m m \cdot n^2) = O((g+1)^{4s} s \cdot n^2)$ time algorithm from Theorem 10. Otherwise $n < 4g$ and we can solve the problem by brute force, testing all possible subsets of rows to be modified in $O(2^n \cdot m) = O(2^{4g} \cdot m)$ time. □

5.4 Close to Guarantee and Small Gap

Under some reasonable complexity-theoretic assumptions Lobbying is neither in XP for the below-guarantee parameter $k' := \lceil (n+1)/2 \rceil - k$ (see Theorem 1) nor in XP for the maximum gap value g over all columns (see Theorem 3). However, using some relations between the parameters and the brute-force algorithm from Proposition 2 we can show that Lobbying is in XP for the combined parameter (g, k'). More precisely, we show that Lobbying is fixed-parameter tractable with respect to k' if g is a constant.

Theorem 11. Lobbying *is solvable in* $O(m^{2g+1} \cdot 2^{2k'})$ *time.*

Proof. From the definition of k' it follows that $k + k' \geq n/2$. Furthermore, from the logarithmic factor approximation (see Theorem 8) it follows that $k \leq g \cdot \log m$. Combining this gives $n \leq 2(g \cdot \log m + k')$. Thus, using the brute-force algorithm from Proposition 2, Lobbying is solvable in $O(2^n \cdot m) = O(2^{2g \cdot \log m} \cdot 2^{2k'} \cdot m) = O(m^{2g+1} \cdot 2^{2k'})$ time. □

It remains open whether Lobbying is fixed-parameter tractable for the combined parameter (g, k') (not assuming that g is a constant).

6. Experimental Evaluation of the Greedy and ILP Algorithms

Recall that we consider the number m of columns to be the parameter with currently strongest support in terms of doing a parameterized complexity analysis. This is because m seldom exceeds 20 (Christian et al., 2007). Hence, in this section we present the experimental results we obtained from testing our greedy heuristic (MaxGapZeros) and our ILP formulation (ILP) from Section 5.2 as well as the slightly simpler greedy algorithm (MaxZeros) by Erdélyi et al. (2007). Roughly speaking, whereas MaxZeros simply picks a row with a maximum number of 0s, MaxGapZeros uses a gain measure where 0s in columns with higher gap value are given higher attention. Thus, by comparing MaxGapZeros and MaxZeros we evaluated the effect of our refined gain measure.

We evaluated both efficiency and solution quality of the greedy algorithms, comparing with exact solutions delivered by the ILP. Note that the heuristic by Erdélyi et al. (2007) was designed to solve the more general weighted variant of Lobbying, and hence, we do not expect MaxZeros to outperform MaxGapZeros in terms of solution quality. However, we are not aware of any other algorithm solving Lobbying to compare with.

We used Gurobi 5.0.1 as our (integer) linear program solver to handle the ILP formulation of Lobbying given in the proof of Theorem 9. Recall that the number of variables in the ILP formulation depends only on the number of different rows and thus can be upper-bounded by 2^m.

We tested the algorithms on a large set ($\approx 3.3 \cdot 10^5$ matrices) of random instances and instances generated from the actual voting records of the German parliament. We used two types of random models where each was controlled by several "density parameters", determining the fraction of the number of 1s to the number of 0s. Our "row-oriented" model randomly chooses a density for each row and then sets each row entry at random such that it equals 1 with a probability equal to the chosen density value. In a similar way, our "column-oriented" approach randomly chooses a density for each column and then generates the entries in this column at random using this density value as the probability for generating a 1. Section 6.1 contains a more detailed description of our random models. In Section 6.3 we present our experimental findings.

All our experiments were performed on an Intel Xeon E5-1620 3.6GHz machine with 64GB of memory running the Debian GNU/Linux 6.0 operating system. Both greedy heuristics are implemented in C++. The ILP implementation uses Gurobi[4] 5.0.1 through its Java API. The source code including the data generator is freely available.[5]

6.1 Random Instance Generation

In both models, the row-oriented and the column-oriented, the process of creating a random instance is controlled by two density parameters a and b with $0 \leq a \leq b \leq 1$. Subsequently, when describing our two models in detail, "randomly choosing a number" always refers to a random number generated using an i.i.d. process and a uniform distribution.

6.1.1 Row-Oriented Model

For certain values of n, m, a, and b with $a \leq b$, the row-oriented model creates a binary matrix $A \in \{0,1\}^{n \times m}$ as follows. For each row i, it chooses a random number d_i from the interval $[a, b]$. Then, for all $1 \leq j \leq m$, entry $A_{i,j}$ is set to 1 with probability d_i. Using the row-oriented model, we created instances for all combinations of $a \in \{0.1, 0.2, 0.3, 0.4\}$ and $b \in \{0.5, 0.6, 0.65, 0.7, 0.8\}$ where $a + b < 1$. Since the expected fraction of 1s in row i is d_i and the expected fraction of 1s in A is $(a + b)/2$, implying that in case of $a + b \geq 1$ a high fraction of columns have gaps at most zero, we required that $a + b < 1$. Thus, for each combination of n and m, we created 13 instances according to this model. Herein, we started with $m = 10$ and increased m by 10% in each step, rounding up to the nearest

4. http://www.gurobi.com/
5. http://akt.tu-berlin.de/menue/software/

integer if necessary (formally, $m \leftarrow \lceil m \cdot 1.1 \rceil$), until m was larger than 300. We assigned n 110 equidistant values within $[10, 991]$ (formally, $n \leftarrow n + 9$). Summarizing, we had 33 different values for m and 110 different values for n, obtaining a total of 47 190 instances in the row-oriented model.

6.1.2 Column-Oriented Model

Here, to create a matrix $A \in \{0,1\}^{n \times m}$ for fixed a and b, for each column i the column-oriented model chooses a random number d_i from $[a, b]$. Then, denoting the maximum possible gap $\lceil (n+1)/2 \rceil$ by $g_{\max}$, the model assigns 1s to $(1-d_i) \cdot g_{\max}$ many entries in column i, and 0 to the remaining entries. Observe that this ensures that the gap of column i is $d_i \cdot g_{\max}$ and thus the expected gap of a column is $((a+b)/2) \cdot g_{\max}$. We created instances for all combinations of $a \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ and $b \in \{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ where $a \leq b$. Thus, for each combination of n and m we created 51 instances. Thereby, we started with $m = 10$ and increased m by 10% in each step , once again rounding up to the nearest integer if necessary (formally, $m \leftarrow \lceil m \cdot 1.1 \rceil$) as long as it was smaller than 100. We assigned n 110 equidistant values within $[10, 991]$ (formally, $n \leftarrow n + 9$). Thus we had 21 different values for m and 110 for n, obtaining a total of 142 810 instances from the column-oriented model.

We discarded instances containing a column with gap at most zero. Furthermore, we do not discuss instances with less than 10 columns because all tested algorithms were extremely fast (less than 0.01 seconds on average) and our greedy heuristic computed optimal solutions for more than 99% of these instances.

6.2 Real-World Data Generation

To obtain a data set representing realistic binary preferences of politicians we use the well-documented voting records of the German parliament, the "Bundestag". Votes on issues in the Bundestag can be anonymous or recorded votes. Whenever a party from the parliament or 5% of the members request it, each voter's decision is recorded together with the voter's name. We generated our instances from the recorded votes in 2012 and 2013 which are freely available from `www.bundestag.de`:

- The issue of each recorded vote became an issue in our model. When there were several very similar issues (for example similar amendments on the same topic) we only took the last one.

- Each member of the Bundestag became a voter in our model. When a Bundestag member did not show up or abstained from voting we assume that his or her opinion was consistent with the majority of his or her party. When a Bundestag member left the parliament during the period, we did not collect his or her votes. (This happened twice.)

This resulted in a matrix with 67 columns and 620 rows. Approximately half of the columns have a small gap value (less than 25) and the other half has gap values greater than 90. Roughly one third of the columns have gap values greater than 150. Even the maximum possible gap value of 311 occurs twice.

6.2.1 The Goal of the Lobby

Since lobby actions are usually not very well-documented and one cannot (easily) identify voters that have been bribed, we decided to guess the goal of the lobby. Recall that in our model, a 1 (resp. a 0) in column j of row i of the input matrix means that voter i agrees (resp. disagrees) with the lobby with respect to issue j. Hence, to get input matrices for our model, we need the set of issues for which the lobby disagrees with a majority of the voters. Issues for which the goal of the lobby is consistent with the majority of voters can always be ignored.

6.2.2 Test Series

We performed three experiments, one with randomly selected issues of small gap values ($g < 30$), one with randomly selected issues of high gap values ($g > 90$), and one with randomly selected issues without any restriction. For each number m of columns from $\{5, 10, \ldots, 30\}$ we extracted 100 instances for each of the first two experiments. For each m from $\{5, 10, \ldots, 60\}$ we generated 100 instances for the last experiment. This covers a wide range of scenarios which differ in the number of issues the lobby wants to change as well as in the amount of changes each issue needs to reach a majority of approvals.

6.3 Results

We evaluated the experimental results concerning time efficiency and, for the greedy heuristics, additionally with respect to solution quality (closeness to optimal solutions), that is, the distance of the derived solutions from an optimal one.

To evaluate the efficiency for some fixed attribute values (e.g., the number of columns) we computed the average running times over all corresponding instances. The running times of the greedy algorithms were very small (below 0.1 seconds on average). To get reliable values we ran both greedy heuristics 20 times for each instance and stored the average value. The ILP algorithm solved more than 95% of the instances within five minutes. We counted the running time for the remaining instances as five minutes to get a lower bound for the correct average running times of the ILP algorithm.

To evaluate the optimality we computed the percentage of optimally solved instances as well as the average difference between the optimal number of lobbied voters and the number of voters lobbied by the heuristic solution. In the following, we denote this as the "distance from optimality".

6.3.1 Efficiency for the Row-Oriented Model

As expected, the heuristic algorithms were much faster than the ILP algorithm. Whereas both heuristics needed less than 0.1 seconds on instances with more than 300 columns or up to 1000 rows, the ILP needed more than ten seconds on average for instances with at least 150 columns or with at least 500 rows. See Figure 3 for details. Somewhat surprisingly, MaxGapZeros turned out to be faster than MaxZeros for instances with more than 24 columns as well as for instances with at least 100 rows. The reason seems to be that MaxGapZeros produces smaller solutions than MaxZeros allowing for earlier termination (see Figure 4).

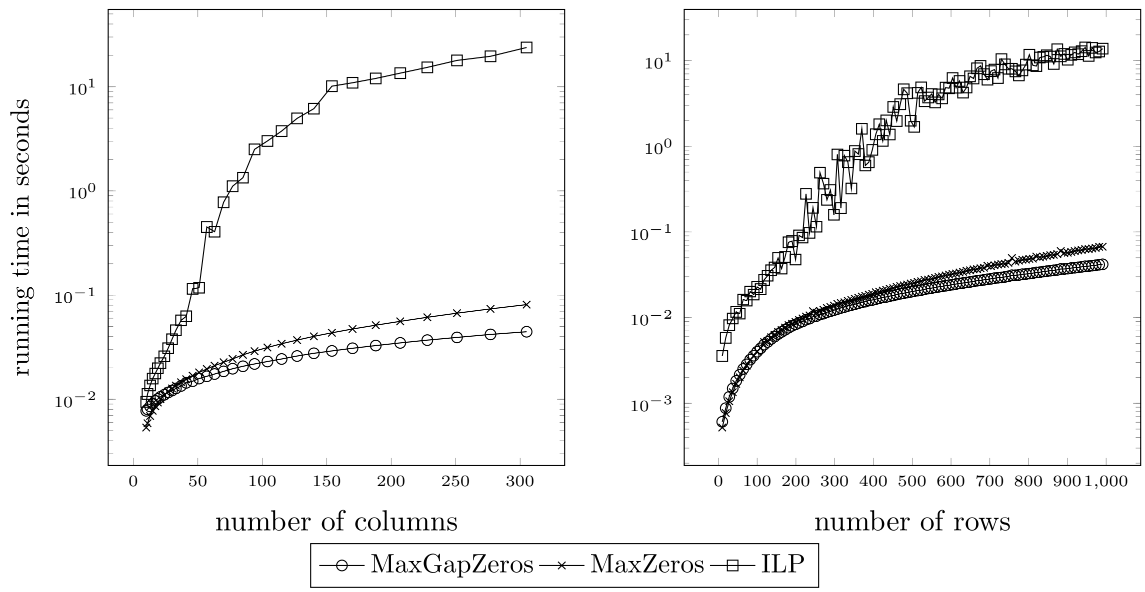

Figure 3: Row-oriented model: Running time depending on the number of columns and the number of rows, respectively.

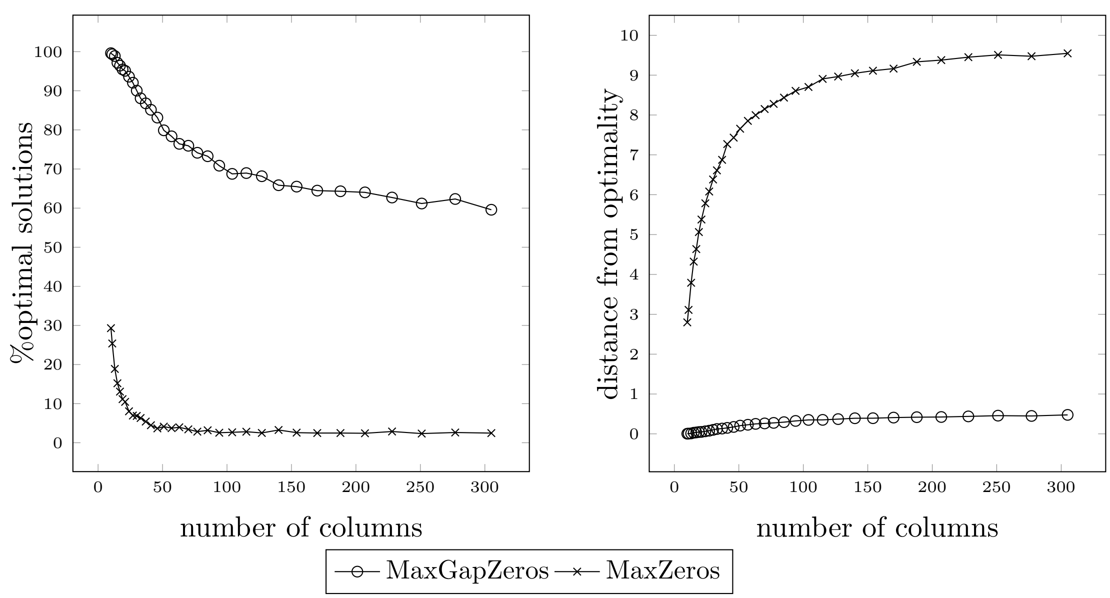

Figure 4: Row-oriented model: Percentage of optimal solutions and average distance from the optimal solution for both greedy algorithms, depending on the number of columns.

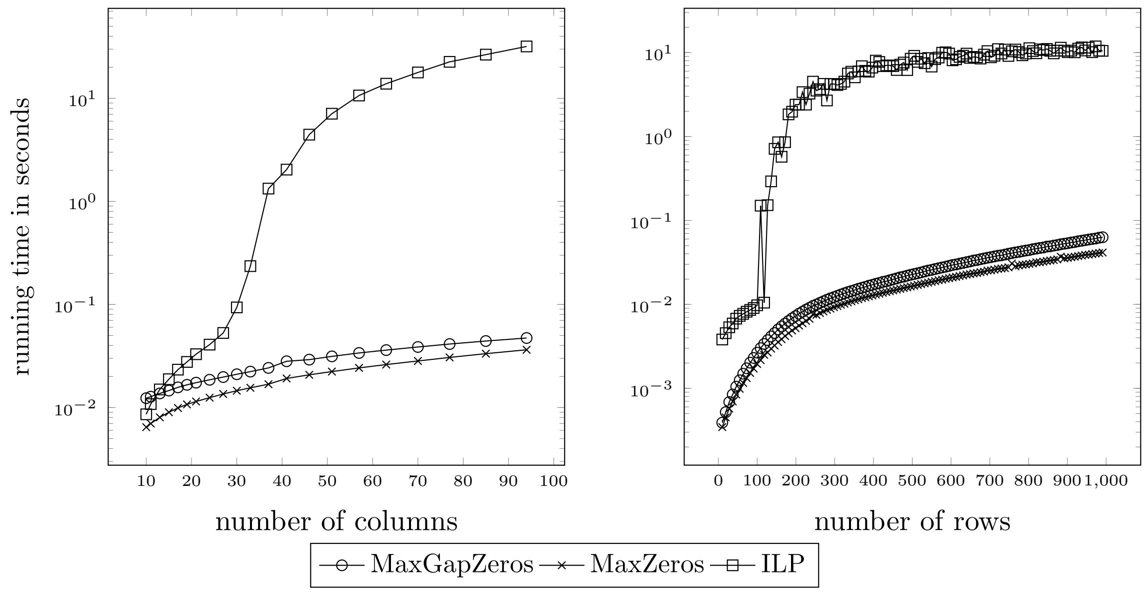

Figure 5: Column-oriented model: Running time depending on the number of columns and the number of rows, respectively.

6.3.2 Optimality for the Row-Oriented Model

Our greedy algorithm (MaxGapZeros) performed very well for this data set in terms of solution size. More than 50% of the instances were optimally solved, even those with more than 300 columns. In contrast, the simpler greedy algorithm (MaxZeros) could only solve very few instances optimally. As for the distance to optimality, MaxGapZeros results have an average distance of less than one whereas the average distance to optimality of the MaxZeros results exhibits a logarithmic growth with respect to the number of issues and it is always greater than two. See Figure 4 for details.

6.3.3 Efficiency for the Column-Oriented Model

Similarly to the row-oriented model, the heuristic algorithms were extremely fast for all tested instances. For all instances (which all have less than 100 columns), MaxZeros was slightly faster than MaxGapZeros but the difference between the average running time of MaxGapZeros and MaxZeros decreased with an increasing number of issues. See Figure 5 for details.

6.3.4 Optimality for the Column-Oriented Model

Similarly to the row-oriented model, MaxGapZeros computed solutions which are relatively close to the optimum. Whereas the percentage of instances having between 30 and 100 columns which were optimally solved by MaxGapZeros is slightly lower than for the row-oriented model, the percentage of instances which were optimally computed by MaxZeros is twice as high as for the row-oriented model. In contrast, as for the distance to optimality,

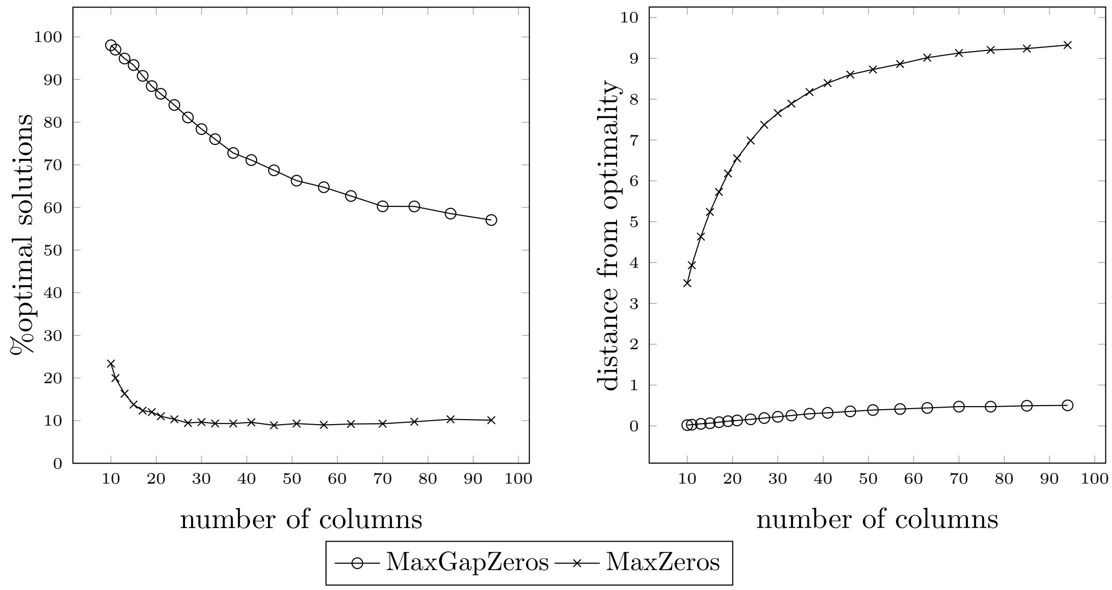

Figure 6: Column-oriented model: Percentage of optimal solutions and average distance from the optimal solution for both greedy algorithms depending on the number of columns.

the results for the column-oriented model behaved similarly to the results for the row-oriented model. Again, the average distance to optimality of the MaxZeros results exhibits a logarithmic growth with respect to the number of issues and is always greater than three, while the average distance to optimality of the MaxGapZeros results is lower than one for all tested instances. See Figure 6 for details.

6.3.5 Efficiency for the Real-World Data Set

Surprisingly, all algorithms including the ILP algorithm could solve every single instance extremely fast. MaxZeros was slightly faster than MaxGapZeros and even the ILP algorithm needed less than 0.07 seconds for each instance. A reason seems to be that politicians of the same party often vote similarly so that there are few different rows in the matrices and, hence, the ILP has much less variables to handle than in the worst case (only up to 103 instead of 620). As expected, instances with small maximum gap values could be computed (slightly) faster.

6.3.6 Optimality for the Real-World Data Set

Somewhat unexpectedly, MaxGapZeros could solve all instances optimally. Recall that this was not the case for random data instances of similar sizes. In contrast to MaxGapZeros, for most instances MaxZeros could not find an optimal solution. Especially for the instances with only high gap values and for the instances without restrictions on the gap values, the distance of MaxZeros's solution size to the optimal solution size is quite large. See Figure 7 and Figure 8 for details. Interestingly, the optimal solution size equals the maximum gap

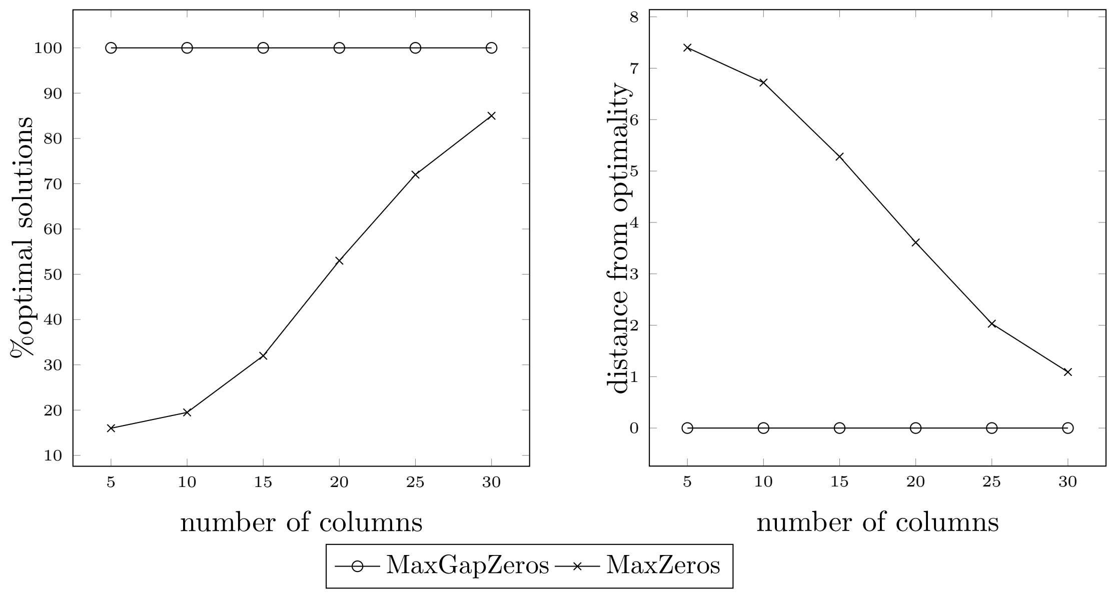

Figure 7: Real-world data, instances with low gap values only: Percentage of optimal solutions and average distance from the optimal solution for both greedy algorithms depending on the number of columns.

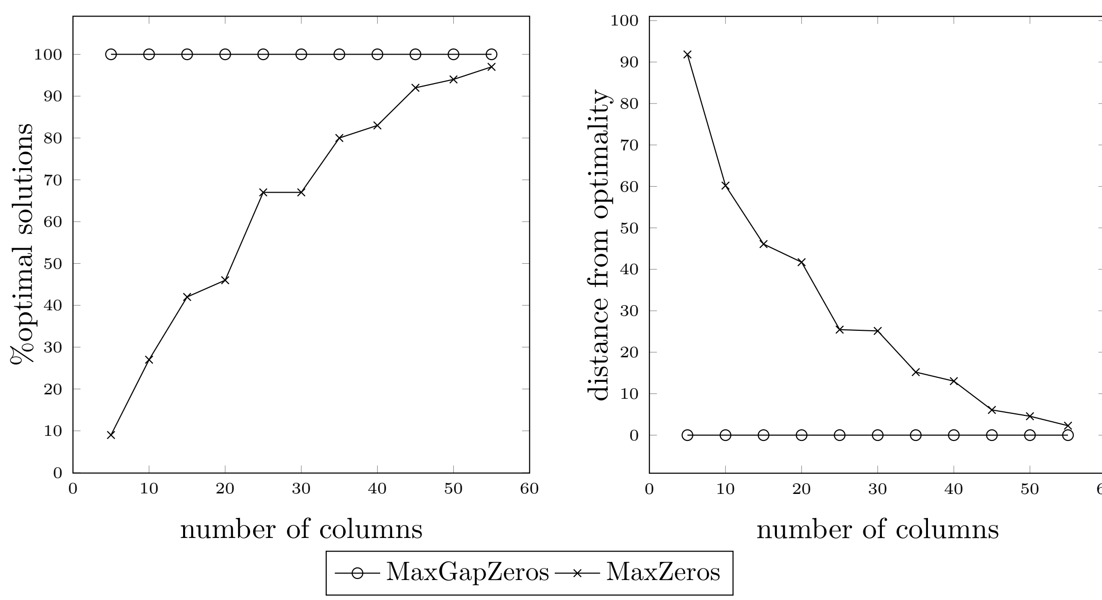

Figure 8: Real-world data, instances with low and high gap values: Percentage of optimal solutions and average distance from the optimal solution for both greedy algorithms depending on the number of columns. The results for instances with high gap values only are very similar.

value for almost all instances. Note that this cannot be the only explanation for tractability since such instances are in general still NP-hard to compute; see the reduction used in the proof of Theorem 1.

6.3.7 Conclusion

We showed that both heuristics are very efficient on synthetic random data of reasonable size, that is, for up to 300 columns and up to 1000 rows. Whereas the greedy heuristic by Erdélyi et al. (2007) computes solutions that are relatively far away from the optimum, our greedy algorithm computed optimal solutions for most instances. However, note that our greedy algorithm is directly designed to solve Lobbying whereas the algorithm by Erdélyi et al. (2007) is designed for the more general weighted variant of Lobbying. Somewhat unexpectedly, also the exact ILP algorithm solved most of our instances within five minutes. Even all instances with up to 37 columns and up to 100 rows were optimally solved by the ILP within five minutes.

On the data set based on real-world data, the algorithms behaved very similar to the synthetic case. Our greedy algorithm could solve all instances optimally and the ILP algorithm could solve all instances very fast.

7. Conclusion

Lobbying as studied here is a fundamental matrix modification problem with a rich combinatorial structure.[6] We started exploiting this structure in terms of a number of natural parameterizations and a corresponding parameterized and multivariate complexity analysis. Table 1 in Section 1 summarizes our results. Indeed, the space of investigations could be extended by introducing further parameters and also looking at further parameter combinations, all the time with the ultimate goal to obtain (improved) fixed-parameter tractability results (Komusiewicz & Niedermeier, 2012). So far, our results indicate that making use of the parameter number m of columns is particularly promising albeit only a "theoretical fixed-parameter tractability" result based on integer linear programming could be proven. We found a very well performing greedy algorithm that turns out to deliver provably optimal results for input matrices with up to four columns. Based on our positive experimental results, we suggest this algorithm to be the current method of choice in solving also larger Lobbying instances getting close to optimal solutions.

On the methodological side, our probably most innovative contribution is to show LOGSNP-completeness (Papadimitriou & Yannakakis, 1996) in case of input matrices which only have gap value 1 for all columns, that is, a situation where the lobby may hope to achieve the goal at low cost (meaning few modified rows). This result is of particular interest since only few natural LOGSNP-complete problems are known and since it seems to be the first use of this "tool" in assessing the parameterized complexity of parameterized problems between XP and the class of problems that are NP-hard even for constant parameter values. This can be of independent future interest when studying the parameterized complexity of other problems.

6. Very recently, Lobbying was very useful in assessing the computational complexity of a matrix modification problem arising in machine learning (Froese, van Bevern, Niedermeier, & Sorge, 2013).

In future work, our findings for the plain Lobbying problem may be extended to natural variants and generalizations. These include

- to allow the lobby to only partially influence the voters (that is, the rows may not be changed to all-1 rows),
- to only head for getting a majority of approvals for a certain (pre-specified) percentage of issues (columns),
- to consider the modification operation of adding voters (rows), and
- to consider the modification operation of deleting voters (rows).

Moreover, it is of interest to extend the multivariate studies to the scenarios of probabilistic and weighted lobbying as have been studied in previous work (Binkele-Raible et al., 2014; Erdélyi et al., 2007).

We conclude with a few concrete open questions for future research on Lobbying.

- Can one replace the integer linear program (see Theorem 9 in Section 5.2) by a direct (more efficient) combinatorial algorithm in the fixed-parameter tractability result for Lobbying parameterized by the number m of columns?
- Our fixed-parameter tractability results for the parameter combinations (m, k), (m, g), (m, s), (k, s), and (g, s) are of theoretical nature only—can they be made practical?
- We showed that Lobbying is fixed-parameter tractable with respect to k' when g is a constant. Is it also fixed-parameter tractable for the combined parameter (g, k')?
- For the combined parameters (m, n), (t, n), and (s, n) there are trivial polynomial-size problem kernels for Lobbying because there are polynomials only depending on the respective parameters which upper-bound the input size. We showed that, except for (m, k') which remains open, for all of our other fixed-parameter tractability results there are no polynomial-size problem kernels, unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$. Is there any possibility of (provable) efficient and effective preprocessing and data reduction for Lobbying?
- All our results adhere to a worst-case analysis—what is the complexity of Lobbying on average? Our findings with the greedy heuristic suggest that there is reason to believe that Lobbying is computationally not so hard as worst-case analysis suggests. The situation is similar in the recent studies concerning the seemingly "pathological" NP-hardness of manipulation and its very good solvability in experimental results (Betzler, Niedermeier, & Woeginger, 2011; Davies, Katsirelos, Narodytska, & Walsh, 2011; Davies, Narodytska, & Walsh, 2012; Walsh, 2011). A more thorough investigation in this direction concerning Lobbying seems promising.

Acknowledgments

An extended abstract of this work (without coauthor G.J. Woeginger) appeared in the Proceedings of the 26th Conference on Artificial Intelligence (AAAI '12) (Bredereck, Chen, Hartung, Kratsch, Niedermeier, & Suchý, 2012). In this long version, now exclusively focusing on the plain Lobbying problem, we provide numerous details that have been omitted in the extended abstract. Moreover, we have the following new contributions: We prove LOGSNP-completeness for the case of "gap-1 instances". We show that our greedy algorithm as already presented in the extended abstract has a logarithmic approximation ratio. Finally, we present experimental results with greedy algorithms and an integer linear program formulation for Lobbying.

Robert Bredereck is supported by the German Research Foundation (DFG), research project PAWS (NI 369/10). Jiehua Chen is supported by the Studienstiftung des Deutschen Volkes. Main work was done while Stefan Kratsch was with Utrecht University supported by the Netherlands Organization for Scientific Research (NWO), project KERNELS (OND1336203), and visiting TU Berlin, Germany. Ondřej Suchý is supported by the DFG Cluster of Excellence on Multimodal Computing and Interaction (MMCI) and the DFG project DARE (GU 1023/1-2). Main work was done while he was with the Universität des Saarlandes, Saarbrücken and visiting TU Berlin, Germany. Gerhard J. Woeginger is supported by DIAMANT (a mathematics cluster of the Netherlands Organization for Scientific Research NWO). Main work was done while he was staying as a recipient of a Humboldt Research Award at TU Berlin.

We are grateful to the anonymous referees of *JAIR* for providing numerous insightful remarks that helped to significantly improve the paper. We thank Kolja Stahl for his great support in extracting and converting the real-world data for our experiments.

References

Bartholdi III, J. J., Tovey, C. A., & Trick, M. A. (1992). How hard is it to control an election?. *Mathematical and Computer Modeling*, *16*(8-9), 27–40.

Baumeister, D., Erdélyi, G., & Rothe, J. (2011). How hard is it to bribe the judges? A study of the complexity of bribery in judgment aggregation. In *Proceedings of the 2nd International Conference on Algorithmic Decision Theory*, Vol. 6992 of *LNCS*, pp. 1–15. Springer.

Betzler, N., Bredereck, R., Chen, J., & Niedermeier, R. (2012). Studies in computational aspects of voting—a parameterized complexity perspective. In *The Multivariate Algorithmic Revolution and Beyond*, Vol. 7370 of *LNCS*, pp. 318–363. Springer.

Betzler, N., Niedermeier, R., & Woeginger, G. J. (2011). Unweighted coalitional manipulation under the Borda rule is NP-hard. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 55–60. AAAI Press.

Binkele-Raible, D., Erdélyi, G., Fernau, H., Goldsmith, J., Mattei, N., & Rothe, J. (2014). The complexity of probabilistic lobbying. *Discrete Optimization*, *11*, 1–21.

Bodlaender, H. L. (2009). Kernelization: New upper and lower bound techniques. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation*, Vol. 5917 of *LNCS*, pp. 17–37. Springer.

Bodlaender, H. L., Downey, R. G., Fellows, M. R., & Hermelin, D. (2009). On problems without polynomial kernels. *Journal of Computer System Sciences*, *75*(8), 423–434.

Bodlaender, H. L., Thomassé, S., & Yeo, A. (2011). Kernel bounds for disjoint cycles and disjoint paths. *Theoretical Computer Science*, *412*(35), 4570–4578.

Bredereck, R., Chen, J., Hartung, S., Kratsch, S., Niedermeier, R., & Suchý, O. (2012). A multivariate complexity analysis of lobbying in multiple referenda. In *Proceedings of the 26th Conference on Artificial Intelligence*, pp. 1292–1298. AAAI Press.

Cai, L., Chen, J., Downey, R. G., & Fellows, M. R. (1997). Advice classes of parameterized tractability. *Annals of Pure and Applied Logic*, *84*(1), 119–138.

Christian, R., Fellows, M., Rosamond, F., & Slinko, A. (2007). On complexity of lobbying in multiple referenda. *Review of Economic Design*, *11*(3), 217–224.

Davies, J., Katsirelos, G., Narodytska, N., & Walsh, T. (2011). Complexity of and algorithms for Borda manipulation. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 657–662. AAAI Press.

Davies, J., Narodytska, N., & Walsh, T. (2012). Eliminating the weakest link: Making manipulation intractable?. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pp. 1333–1339. AAAI Press.

Dom, M., Lokshtanov, D., & Saurabh, S. (2009). Incompressibility through colors and IDs. In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming*, Vol. 5555 of *LNCS*, pp. 378–389. Springer.

Dorn, B., & Schlotter, I. (2012). Multivariate complexity analysis of swap bribery. *Algorithmica*, *64*(1), 126–151.

Downey, R. G., & Fellows, M. R. (2013). *Fundamentals of Parameterized Complexity.* Texts in Computer Science. Springer.

Elkind, E., Faliszewski, P., & Slinko, A. (2011). Cloning in elections: Finding the possible winners. *Journal of Artificial Intellligence Research*, *42*, 529–573.

Elkind, E., Faliszewski, P., & Slinko, A. (2012). Clone structures in voters' preferences. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pp. 496–513. ACM.

Erdélyi, G., Hemaspaandra, L. A., Rothe, J., & Spakowski, H. (2007). On approximating optimal weighted lobbying, and frequency of correctness versus average-case polynomial time. In *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory*, Vol. 4639 of *LNCS*, pp. 300–311. Springer.

Erdélyi, G., Piras, L., & Rothe, J. (2011). The complexity of voter partition in Bucklin and fallback voting: Solving three open problems. In *Proceedings of the 10th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 837–844. IFAAMAS.

Faliszewski, P., Hemaspaandra, E., & Hemaspaandra, L. A. (2009). How hard is bribery in elections?. *Journal of Artificial Intelligence Research*, *35*, 485–532.

Fellows, M. R., Jansen, B. M., & Rosamond, F. (2013). Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *European Journal of Combinatorics*, *34*(3), 541–566.

Flum, J., & Grohe, M. (2006). *Parameterized Complexity Theory*. Springer.

Fortnow, L., & Santhanam, R. (2011). Infeasibility of instance compression and succinct PCPs for NP. *Journal of Computer System Sciences*, *77*(1), 91–106.

Frank, A., & Tardos, É. (1987). An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, *7*(1), 49–65.

Froese, V., van Bevern, R., Niedermeier, R., & Sorge, M. (2013). A parameterized complexity analysis of combinatorial feature selection problems. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science*, Vol. 8087 of *LNCS*, pp. 445–456. Springer.

Gabow, H. N. (1983). An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, pp. 448–456. ACM.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability—A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Goldsmith, J., Levy, M. A., & Mundhenk, M. (1996). Limited nondeterminism. *SIGACT News*, *27*(2), 20–29.

Gottlob, G., Scarcello, F., & Sideri, M. (2002). Fixed-parameter complexity in AI and nonmonotonic reasoning. *Artificial Intelligence*, *138*(1-2), 55–86.

Gottlob, G., & Szeider, S. (2008). Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems. *The Computer Journal*, *51*(3), 303–325.

Guo, J., & Niedermeier, R. (2007). Invitation to data reduction and problem kernelization. *SIGACT News*, *38*(1), 31–45.

Kannan, R. (1987). Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, *12*(3), 415–440.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103. Plenum Press.

Komusiewicz, C., & Niedermeier, R. (2012). New races in parameterized algorithmics. In *Proceedings of the 37th Mathematical Foundations of Computer Science*, Vol. 7464 of *LNCS*, pp. 19–30. Springer.

Lenstra, H. W. (1983). Integer programming with a fixed number of variables. *Mathematics of Operations Research*, *8*(4), 538–548.

Mahajan, M., & Raman, V. (1999). Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, *31*(2), 335–354.

Niedermeier, R. (2006). *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Niedermeier, R. (2010). Reflections on multivariate algorithmics and problem parameterization. In *Proceedings of the 27th International Symposium on Theoretical Aspects of Computer Science*, Vol. 5 of *Leibniz International Proceedings in Informatics*, pp. 17–32.

Papadimitriou, C. H., & Yannakakis, M. (1996). On limited nondeterminism and the complexity of the V-C dimension. *Journal of Computer and System Sciences*, *53*(2), 161–170.

Sandholm, T., Suri, S., Gilpin, A., & Levine, D. (2002). Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Conference on Autonomous Agents and Multiagent Systems*, pp. 69–76. ACM.

Schlotter, I., Elkind, E., & Faliszewski, P. (2011). Campaign management under approval-driven voting rules. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 726–731. AAAI Press.

Schrijver, A. (2003). *Combinatorial Optimization: Polyhedra and Efficiency*, Vol. A. Springer.

Szeider, S. (2011). Limits of preprocessing. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 93–98. AAAI Press.

Walsh, T. (2011). Is computational complexity a barrier to manipulation?. *Annals of Mathematics and Artificial Intelligence*, *62*(1-2), 7–26.

Yap, C.-K. (1983). Some consequences of non-uniform conditions on uniform classes. *Theoretical Computer Science*, *26*(3), 287–300.