A Multivariate Complexity Analysis of Lobbying in Multiple Referenda

Robert Bredereck[*], **Jiehua Chen**[†], **Sepp Hartung**, **Rolf Niedermeier** and **Ondřej Suchý**[‡]
TU Berlin, Berlin, Germany
{robert.bredereck, jiehua.chen, sepp.hartung, rolf.niedermeier}@tu-berlin.de
suchy@kam.mff.cuni.cz

Stefan Kratsch
Universiteit Utrecht, Utrecht, The Netherlands
s.kratsch@uu.nl

Abstract

We extend work by Christian et al. [Review of Economic Design 2007] on lobbying in multiple referenda by first providing a more fine-grained analysis of the computational complexity of the NP-complete LOBBYING problem. Herein, given a binary matrix, the columns represent issues to vote on and the rows correspond to voters making a binary vote on each issue. An issue is approved if a majority of votes has a 1 in the corresponding column. The goal is to get all issues approved by modifying a minimum number of rows to all-1-rows. In our multivariate complexity analysis, we present a more holistic view on the nature of the computational complexity of LOBBYING, providing both (parameterized) tractability and intractability results, depending on various problem parameterizations to be adopted. Moreover, we show non-existence results concerning efficient and effective preprocessing for LOBBYING and introduce natural variants such as RESTRICTED LOBBYING and PARTIAL LOBBYING.

1 Introduction

Campaign management, roughly speaking, comprises all sorts of activities for influencing the outcome of an election, including well-known scenarios such as bribery (Faliszewski, Hemaspaandra, and Hemaspaandra 2009; Dorn and Schlotter 2012; Schlotter, Elkind, and Faliszewski 2011) and control (Bartholdi III, Tovey, and Trick 1992; Elkind, Faliszewski, and Slinko 2010; Erdélyi, Piras, and 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 *not* win, Christian et al. (2007) introduced the scenario of lobbying in multiple referenda by considering the following combinatorial problem:

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

Herein, *modifying* a row allows to flip all 0s to 1s. Then, modifying a minimum number of rows can be interpreted as a *lobby* (that is, a set of external agents) influencing a minimum number of voters to get all issues approved.

In difference to Christian et al. (2007), we assume that the "desired outcome" of each column is 1. 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.

As a special case of *reverse combinatorial auction* (Sandholm et al. 2002), LOBBYING is relevant to combinatorial markets in multi-agent systems. It is also closely related to *bribery in judgment aggregations* (Baumeister, Erdélyi, and 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.

LOBBYING is NP-complete and Christian et al. (2007) even showed that the problem is W[2]-complete with respect to the parameter k, that is, even if only a small number of voters shall be influenced the problem is computationally (worst-case) hard. In the first part of this work, we provide a much more refined view on the multivariate computational complexity (Fellows 2009; Niedermeier 2010) of LOBBYING. To this end, we identify the following parameters naturally occurring in LOBBYING and analyze their influence on its computational complexity. Herein, the *gap* of a column is the number of additional 1s this column needs in order to have a majority of 1s.

- n: number of rows;
- m: number of columns[1];
- k: number of rows to modify;
- g: maximum gap over all columns;
- s: maximum number of 1s per row;
- t: maximum number of 0s per row.

Note that the various parameters are partially related to each other; for instance, for every non-trivial instance of LOBBYING it holds that $k < \lceil (n+1)/2 \rceil$, $m \leq n \cdot s$, $t \leq m \leq 2t$, and $g \leq k$.

LOBBYING as defined above is a first step towards modeling the phenomena in the real world. There are many possible refinements such as variating the prices of the voters or using

[*]Supported by the DFG project PAWS (NI 369/10).
[†]Supported by the Studienstiftung des Deutschen Volkes.
[‡]Main work done while with the Universität des Saarlandes, Saarbrücken, Germany. Supported by the DFG Cluster of Excellence on Multimodal Computing and Interaction (MMCI) and the DFG project DARE (GU 1023/1-2).

[1]Christian et al. (2007) argued that m seldom exceeds 20.

	m	k	g	s
m	ILP-FPT (Theorem 8)	FPT (Theorem 9)	FPT (Theorem 9)	ILP-FPT (Theorem 8)
k		W[2]-complete [a]	W[2]-complete [a]	FPT (Lemma 1)
g			W[2]-hard [a]	FPT (Lemma 1)
s				para-NP-hard ($s \geq 3$) (Theorem 1)

[a](Christian et al. 2007)

Table 1: Summary of our results. For each parameter combination the entries indicate whether LOBBYING is fixed-parameter tractable (FPT), FPT based on a formulation as integer linear program (ILP-FPT), W[2]-hard, W[2]-complete (meaning that it is W[2]-hard and contained in W[2] $\subseteq$XP), or NP-hard even for a constant parameter value (para-NP-hard). Entries on the main diagonal represent results for the single parameters. Furthermore, for each of the parameter combinations above, under some reasonable complexity-theoretic assumptions (see Section 3), there cannot be a polynomial-size problem kernel.

probability values instead of 0-1 values in the matrix. Furthermore, a voter may not want to change his entire preference, more closely resembling the real-world scenarios. In the second part of this work, we introduce some more fine-grained models of control over the voters and the lobbyist's intentions. We hope that our extensions will be useful for future research.

Preliminaries on Parameterized Algorithmics. The concept of parameterized complexity was pioneered by Downey and Fellows (1999) (see also (Flum and Grohe 2006; Niedermeier 2006) for more recent textbooks). 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. Formally, we call LOBBYING *fixed-parameter tractable* with respect to a parameter p (equivalently, contained in the parameterized complexity class FPT) if it can be solved in $f(p) \cdot |A|^{O(1)}$ time, where f solely depends on p. Downey and Fellows (1999) 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. The difference between a parameterized reduction and a classical many-to-one reduction (used to prove NP-hardness) is that for the former it is allowed to take FPT-time but the parameter in the instance one reduces to has to be bounded by some function solely depending on the parameter in the problem instance one reduces from. A further parameterized complexity class of interest is XP, containing all problems that can be solved in $n^{f(p)}$ time for input size n and a function f solely depending on the parameter p. 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.

Related Work. The central point of reference is the work by Christian et al. (2007), whose result is the proof of W[2]-completeness of LOBBYING with respect to k. Erdélyi et al. (2007) generalized LOBBYING to a weighted scenario and showed that an efficient greedy algorithm achieves a logarithmic approximation ratio. Moreover, they showed that essentially no better approximation ratio can be proven for their algorithm. Later, Erdélyi et al. (2009) and Binkele-Raible et al. (2009) proposed models for lobbying in a probabilistic environment, providing (parameterized) hardness and tractability results.

Our Contributions. We provide a multivariate complexity analysis of LOBBYING and look into two practical variants of LOBBYING. Most of our theoretical results are summarized in Table 1. We subsequently sketch a few highlights. In Section 2, we show that LOBBYING remains NP-hard for $s = 3$. Thus, there is no hope for fixed-parameter tractability with respect to s. In Section 3, we reveal limitations of effective polynomial-time preprocessing for LOBBYING. In Section 4, we show that LOBBYING is fixed-parameter tractable for parameter m. From a practical point of view, we provide two efficient algorithms yielding optimal results for input matrices with up to 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 5 we introduce two more fine-grained models of lobbying: RESTRICTED LOBBYING (modeling restricted influence on a voter) and PARTIAL LOBBYING (not the full list of issues needs to be approved).

2 Computational Intractability

In this paper, we provide several reductions from variants of the SET COVER (SC) problem. Given a family of sets $\mathcal{S} = \{S_1, \ldots, S_l\}$ over a universe of elements $\mathcal{U} = \{e_1, \ldots, e_u\}$ and a positive integer k, SC asks whether there is a size-k *set cover*, that is, a subset of $\mathcal{S}$ 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). A common starting point for each of our reductions is to transform $\mathcal{S}$ (and $\mathcal{U}$) into a binary matrix in a similar way as Christian et al. (2007) transform DOMINATING SET instances into binary matrices. The corresponding $|\mathcal{S}| \times |\mathcal{U}|$-matrix is called SC-*matrix* and is defined as

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

We will make use of the SC-matrix in Section 3, in Section 5, and in the following reduction which is used in several hardness proofs in this work.

Reduction 1 (from 3-SC)**.** Let $(\mathcal{S}, \mathcal{U}, k)$ denote a 3-SC-instance. First, compute $M(\mathcal{U}, \mathcal{S})$, which we refer to as *original rows* and *original columns* in the following. Second, add $|\mathcal{S}| - 2k + 1$ additional *dummy rows* and $|\mathcal{S}| - 2k + 1$ additional *dummy columns* containing only 0s. Finally, for $1 \leq i \leq |\mathcal{S}| - 2k + 1$, change the entry in the ith dummy column of the ith dummy row to 1.

To modify at most $|\mathcal{S}| - k$ rows from the matrix such that in the resulting matrix every column has more 1s than 0s requires to select a set cover for $(\mathcal{S}, \mathcal{U}, k)$: The gap of each dummy column is exactly $|\mathcal{S}| - k$ and since each dummy row has a 1 in a dummy column one never modifies any dummy row. Hence, all dummy rows and k original rows remain unmodified in every solution. Since no dummy row contains a 1 in an original column and the solution flips at most $|\mathcal{S}| - k$ entries per column, each original column must contain a 1 in at least one unmodified original row to obtain a majority of 1s. Thus, for every LOBBYING-solution, the sets corresponding to the k unmodified rows must form a set cover because every uncovered element corresponds to an original column containing only 0s in the unmodified rows. □

Theorem 1. LOBBYING *restricted to input matrices with at least three ones per row, that is, $s \geq 3$, is NP-hard.*

Proof. (Sketch.) Reduction 1 constructs LOBBYING instances with at most three 1s per row: By definition of 3-SC every original row has at most three 1s in the original columns and only 0s in the dummy columns. Dummy rows contain exactly one 1. □

An instance of LOBBYING with $k \geq \lceil (n+1)/2 \rceil$ is a yes-instance. Following general ideas of Mahajan and Raman (1999), we investigate the complexity of LOBBYING in case k is slightly below the bound. Such a parameterization is called *below guarantee*.

Theorem 2. LOBBYING *is NP-hard even if the below-guarantee parameter $(\lceil (n+1)/2 \rceil - k)$ takes any positive constant value.*

Proof. (Sketch.) In Reduction 1, we reduce the question whether there is a set cover of size k to the question whether it is possible to modify at most $(|\mathcal{S}| - k)$ rows in a $(2|\mathcal{S}| - 2k + 1)$-rows matrix such that in the resulting matrix every column has more 1s than 0s. Since a trivial solution consists of $(|\mathcal{S}| - k + 1)$ arbitrary rows, NP-hardness holds even if we ask for a solution of size one less than guaranteed. □

3 Limits of Preprocessing

In this section, we prove that LOBBYING does not admit, under a reasonable complexity-theoretic assumption, an efficient preprocessing algorithm formalized as a *polynomial kernelization* in parameterized algorithmics: 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) (I, p) is a yes-instance if and only if (I', p') is a yes-instance and there are two functions f_1 and f_2 such that ii) $p' \leq f_1(p)$ and iii) $|I'| \leq f_2(p)$. The function f_2 measures the size of the *kernel* I' and the kernel is said to be a *polynomial kernel* if f_2 is polynomial.

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 not to admit a polynomial kernel (Bodlaender, Thomassé, and 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 where one reduces from. Also refer to Szeider (2011) for a general account on showing limits of preprocessing for AI problems.

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 relation between parameters, these two results imply the non-existence of polynomial kernels for all the parameter combinations listed in Table 1. Specifically, we use the following reduction to obtain our results.

Reduction 2 (from SC)**.** Let $(\mathcal{S}, \mathcal{U}, k)$ be an SC-instance. First, take $M(\mathcal{U}, \mathcal{S})$ and invert 0s and 1s. Second, add $|\mathcal{S}| - 2k + 1$ *dummy rows* containing only 0s. In every column j, flip $|\mathcal{S}| - |\{S_i : e_j \in S_i\}| - k + 1$ of the 0s from arbitrary dummy rows to 1s.[2] Finally, add one additional *dummy column* containing 1s in the dummy rows, and 0s elsewhere.

The number of 0s in the dummy column is $|\mathcal{S}|$, the number of 0s in any other column is $|\mathcal{S}| - k + 1$, and the total number of rows is $2|\mathcal{S}| - 2k + 1$. We claim that to modify at most k rows from the matrix such that in the resulting matrix every column has more 1s than 0s means to select a set cover for $(\mathcal{S}, \mathcal{U}, k)$: Since the gap in the dummy column is k and no dummy row contains a 0 in the dummy column, a solution cannot modify any dummy row. The gap of each original column is exactly one. Hence, the sets corresponding to the modified rows of any LOBBYING solution must form a set cover. □

Theorem 3. *Unless NP ⊆ coNP/poly,* LOBBYING *does not admit a polynomial kernel with respect to the combined parameter (m, k).*

Proof. (Sketch.) Dom et al. (2009) showed that, unless NP ⊆ coNP/poly, SC does not admit a polynomial kernel with respect to the combined parameter $(|\mathcal{U}|, k)$. Reduction 2 is a polynomial time and parameter transformation: The matrix in the resulting LOBBYING instance has $|\mathcal{U}| + 1$ columns and we are asked to modify at most k columns. Since LOBBYING is (trivially) 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 $(|\mathcal{U}|, k)$. □

Whereas LOBBYING is trivially fixed-parameter tractable with respect to the parameter n, there is no effective preprocessing in the following sense:

[2] We can assume without loss of generality that each element occurs in at least k and in at most $|\mathcal{S}|$-k sets.

Theorem 4. *Unless NP $\subseteq$ coNP/poly,* LOBBYING *does not admit a polynomial kernel with respect to n.*

Proof. (Sketch.) Observe that in Reduction 2 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 NP $\subseteq$ coNP/poly), then LOBBYING also does not admit a polynomial kernel with respect to n. One can show that, indeed, SC does not admit a polynomial kernel with respect to the number of sets. □

4 Tractable Cases

In this section, we demonstrate that LOBBYING is efficiently solvable in practically relevant cases.

Matching-Based Techniques for Parameter $s \leq 2$. The following result complements Theorem 1, thus providing a complexity dichotomy.

Theorem 5. 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. Let $b = \lceil (n+1)/2 \rceil - k$, that is, the number of 1s that each column requires in addition to the k modified rows of a solution. Clearly, we can answer no if the number of 1s in any column is less than b (we answer yes if $k \geq \lceil (n+1)/2 \rceil$). Consider a solution R^* of k rows (to be modified). Let R' be the multiset of remaining rows that are not contained in R^*. We will show that R' contains a special type of matching for a certain auxiliary graph G. Computing a maximum matching of this type will then lead to a solution.

For the auxiliary graph G we use one vertex for each column, and add an edge between two vertices for each row which contains 1s in the corresponding two columns (multiple edges are allowed). For each column, we arbitrarily mark b rows in R' which contain 1 in this column (so each row is marked at most twice). Let $M \subseteq R'$ denote the marked rows, and let $M_2 \subseteq M$ denote the 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 M_2 it is easy to see that each vertex is incident to at most b of them (we only marked b per vertex, and only the twice marked rows are considered); a set of edges having this property is called *a b-matching*.

Let us consider any maximum b-matching M_2^* of G, that is, each vertex is incident to at most b edges of M_2^*; clearly $|M_2^*| \geq |M_2|$. According to Gabow (1983), such a b-matching can be computed in time $O(nm \log m)$ (see also Schrijver (2003, Chapter 31) for a more general notion of b-matchings). We know that the number of 1s in each column is at least b, so we can greedily add rows to M_2^*, obtaining M^* such that the number of 1s in rows of M^* in each column is at least b. We observe that $|M^*| \leq b \cdot m - |M_2^*|$, since we add $b \cdot m - 2|M_2^*|$ additional rows to M_2^* (each one contributes 1 for some column). It follows that

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

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. □

Few Columns and an ILP for the Parameter m. We first provide a simple and efficient greedy algorithm yielding optimal results for input matrices with at most four columns. The algorithm gd() utilizes the gap values of the columns. During each iteration, it modifies a row $r \in \{0,1\}^m$ with minimum *value* $v(r) := \sum_{j=1}^{m} r_j \cdot 2^{m-j}$. In other words, $v(r)$ is derived by interpreting r as a binary number. Algorithm gd() may change the column order. Hence, the value of each row may also change.

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

```
compute the gap values of all columns in A
while matrix A is nonempty do
    reorder columns in descending order of gap values
    modify an arbitrary row with minimum value
    update the gap values of all columns
    remove columns with gap value zero
return modified rows
```

Lemma 1. *Algorithm gd() runs in $O(mn + n(m \log m + mn + m))$ time.*

Proof. Computing the gap values of all columns takes $O(mn)$ time and gd() performs at most $\lceil (n+1)/2 \rceil$ iterations. During each iteration, reordering the columns takes $O(m \log m)$ time, finding a row with minimum value takes $O(mn)$ time, and updating gap values and deleting columns takes $O(m)$ time. □

Theorem 6. *In polynomial time, Algorithm gd() provides optimal solutions for* LOBBYING *when restricted to input matrices with at most four columns, that is, $m \leq 4$.*

Proof. (Sketch.) We need to show that for every two rows r, r' with $v(r) = v(r') - 1$, modifying row r first is not worse than the other way round if r has the smallest value among all rows. We consider only rows with values other than 0 or $2^m - 1$, since it is always best to modify a row of value 0 and never good to modify a row with value $2^m - 1$.

We call row r *better* (or *not worse*) than r' if modifying r before r' is better (or not worse) than the other way round (assuming r has the smallest value among all rows). A column is *over-satisfied* if the number of 1s in this column is more than $\lceil (n+1)/2 \rceil$.

Obviously, gd() works for $m = 1$. For $m = 2$, the essential part is to show that a row of type 01 is not worse than a row of type 10: Suppose that we modify 10 first, then we still need rows whose first column is zero since $g_1 > 0$. However, these can only be rows of type 01, since 01 has currently the smallest value which means the possibly better choice 00 does not exist in the current matrix. Hence, it is not worse to modify 01 first.

For $m = 3$, we consider six types of rows, from 001 to 110. Since 010 is better than 011 and 100 is better than 101, we only need to show that 1) 001 is not worse than 010; 2) 011 is not worse than 100; and 3) 101 is not worse than 110.

Case 1) Suppose that we modify 010 before 001. Then we still need at least g_2 rows whose second column is zero. These can only be of type 100 since 001 is better than 101. If we modify g_2 rows of type 100 (and 010), the third column will be over-satisfied, so we can replace at least one row of type 010 with 001. This means that if there is an optimal solution modifying row 010, then it can be assumed to contain 001, so it is not worse to modify 001 first.

Case 2) Since $g_1 > 0$ and 011 is the only remaining type of row whose first column is zero, modifying 011 is necessary.

Case 3) Having only rows with 1 in the first column means that the first column is satisfied and, thus, all columns are satisfied. Hence, no modifications are needed.

We omit the proof for $m = 4$. □

There are five-column input matrices where the above greedy heuristic may not provide an optimal solution. However, we conjecture that our greedy algorithm can be extended to also provide at least a logarithmic approximation ratio in the general case (also cf. (Erdélyi et al. 2007)).

Exploiting Theorem 5 for parameter $s \leq 2$ (at most two 1s per row) there is an alternative algorithm specifically tailored towards the cases $m \leq 4$ with better running time bound than our greedy algorithm.

Theorem 7. LOBBYING *is solvable in linear time if* $m \leq 4$.

Proof (Sketch.) We modify the matching-based algorithm from Theorem 5 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 ones 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 on the rest and ask it to achieve $\lceil (n+1)/2 \rceil - c$ ones 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 such 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 the row to be modified is of type 0111. Then there is a positive gap in the first column and whenever there is 0 in this column, then it is a 0111 row. Hence, the number of 0s in this column is at least $n/2$ as it is not yet satisfied. But the number is at most $n/2$, as otherwise we would have more than $n/2$ rows of type 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 one row of type 0111 to satisfy the first column. If there was another row with three 1s to be modified, say of type 1011, then this one together with all 0111 rows would satisfy columns 3 and 4 — a contradiction.

We can solve the case $m = 4$ by branching into at most 5 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, counting for each i the number c_i of 1s in them in the column i. Finally, we ask the matching-based algorithm to modify at most $k - 1$ (or at most k, according to the branch followed) of the remaining rows such that the column i contains $\lceil (n+1)/2 \rceil - c_i$ 1-entries. The running time follows from Theorem 5. □

Theorems 6 and 7 show that in case of at most four issues LOBBYING can be solved very efficiently. On the contrary, parameterizing 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. However, the (worst-case) running time of the corresponding algorithm is impractical and of classification nature only.

Theorem 8. LOBBYING *is in* FPT *with respect to the parameter* m.

Proof (Sketch.) We describe an integer linear program (ILP) with no more than 2^m variables that solves LOBBYING.[3] Then, by a result of Lenstra (1983), improved by Kannan (1987), stating that any ILP can be solved in $O(p^{9p/2}L)$ time with p denoting the number of variables and L denoting the number of input bits, it follows that LOBBYING is fixed-parameter tractable with respect to m.

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 all $1 \leq i \leq l$ we introduce the integer variable b_i, where $0 \leq b_i \leq c(r_i)$, whose solution value indicates how often one has to modify a row of type r_i. It is straightforward to argue that a solution for the following ILP provides a solution for LOBBYING in which at most k rows have been modified.

$$\sum_{i=1}^{l} b_i \leq k, \qquad \forall 1 \leq j \leq m : \sum_{i=1}^{l} b_i \cdot B_j(r_i) \geq g_j \quad \square$$

Dynamic Programming for the Parameter (m, g). Theorem 8 provides a purely theoretical result with respect to the parameter m. Combining m with the "gap parameter" g, however, may lead to a practically feasible algorithm.

Theorem 9. 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 gaps 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.

[3] Dorn and Schlotter (2012) already mentioned that their ILP for the SWAP BRIBERY problem could be adapted to LOBBYING.

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 range $1, \ldots, i$. Otherwise it is False. Clearly, the LOBBYING instance (A, k) is a yes-instance if $T[n, k, g_1, \ldots, g_m] =$ 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 calculate 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 $T[i-1, l, \tilde{g}_1, \ldots, \tilde{g}_m] =$ 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 $T[i-1, l-1, g'_1, \ldots, g'_m] =$ True where $g'_j = \tilde{g}_j$ if row i has 1 in the column j and $g'_j = \max\{0, \tilde{g}_j - 1\}$ otherwise. If none of the two cases applies, then we set $T[i, l, \tilde{g}_1, \ldots, \tilde{g}_m] =$ False.

The table has $(g+1)^m kn$ entries and each table entry can be computed in $O(m)$ time, resulting in a running time of $O((g+1)^m knm)$. □

Results Based on Relating Parameters.

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

Proof. Count the number of 1s in the given matrix. Since there are at most s of them in each row, there are at most ns of them in total. In addition, there are more than $n/2 - g$ of them in each column and, hence, the total number of 1s is more than $(n/2 - g)m$. It follows that $(n/2 - g)m < ns$.

If $g \leq n/4$, then $nm/4 \leq (n/2 - g)m < ns$, hence $m < 4s$ 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 9. Otherwise $n < 4g$ and we can solve the problem by brute-force, testing all possible subsets of rows to be modified in $O(2^n n \cdot m) = O(2^{4g} g \cdot m)$ time. □

Since $m \leq 2t$, Theorem 8 implies the following.

Corollary 1. LOBBYING *is in* FPT *with respect to the parameter* t.

About "Non-Hardness" with Respect to the Parameter g. We could not show containment of LOBBYING in XP with respect to the parameter g (max. gap of a column). However, there is "no hope" for showing para-NP-hardness for parameter g as the following theorem shows.[4]

Theorem 10. LOBBYING *is not* NP*-hard for constant* g *unless all problems in* NP *can be solved in time* $|x|^{O(\log |x|)}$*, where* $|x|$ *is the size of the input.*

Proof. We show that LOBBYING is solvable in time $O(n^{g \cdot \log m + 1} \cdot m)$. To this end we claim that if $k \geq g \cdot \log m$, then the answer is yes. To see this, observe first that there is a row containing 0s in at least half of columns. Modify this row and concentrate on the columns in which nothing was modified so far. Again there must be a row containing 0s in at least half of these columns, which one can modify. Continue this way and after a logarithmic number of such steps one has touched each column at least once and g decreases by at least one. Repeating this g times, it is possible to satisfy the instance by modifying at most $g \cdot \log m$ rows.

If $k \geq g \cdot \log m$, then answer yes, and otherwise try all combinations of at most k rows to modify and check for each of them whether all columns get satisfied. This shows the claimed running time. Now, if there was a polynomial time reduction from some NP-complete problem to LOBBYING with constant g, then we would have $n = |x|^{O(1)}$ and $m = |x|^{O(1)}$ for any instance of LOBBYING, and LOBBYING would be solvable in time $|x|^{O(\log |x|)}$, implying the claim. □

[4] Ongoing research in cooperation with Gerhard J. Woeginger (TU Eindhoven) shows that LOBBYING is LOGSNP-complete (Papadimitriou and Yannakakis 1996) for g=1.

5 Variants of LOBBYING

Restricted Lobbying. In the standard formulation of LOBBYING, the lobby may completely change the opinion of the voters for every single referendum. It is plausible to also investigate scenarios where the lobby is only able to influence a limited amount of referenda per voter. Formally, in RESTRICTED LOBBYING one is additionally given the maximum number t' of allowed flips per row. Notably, a simple greedy strategy makes the problem polynomial-time solvable for $t' = 1$. However, the following theorem destroys any hope for fixed-parameter tractability with respect to the parameter t'. The idea of proof for the following theorem is to use Reduction 2, but reduce from 3-SC.

Theorem 11. RESTRICTED LOBBYING *is* NP*-hard for constant* $t' \geq 4$.

Theorem 11 implies that we need to combine parameter t' with a further parameter in order to search for fixed-parameter tractability; this is what we do next. Extending the ideas of the proof of Theorem 9, the following can be shown.

Theorem 12. RESTRICTED LOBBYING *is solvable in time* $O((k+1)^{t'k} \cdot \binom{t'k}{t'} \cdot t' m n^2)$*, and it is in* FPT *with respect to the combined parameter* (s, t').

Partial Lobbying. Consider the situation where it is sufficient to approve a certain number of issues instead of the all issues. This leads to PARTIAL LOBBYING: Is it possible to "satisfy" at least r columns by modifying at most k rows?

Adapting the ILP formulation from Theorem 8, one can show that PARTIAL LOBBYING is in FPT for the parameter m.

Proposition 2. PARTIAL LOBBYING *is in* FPT *with respect to the parameter* m.

Now, we take a closer look at parameter r.

Proposition 3. PARTIAL LOBBYING *is in* FPT *with respect to the combined parameter* (g, r).

Proof. Remove all columns with gap greater than k such that $k \geq g$ in the following. If $m \leq 2^g r$, we use Proposition 2 to solve the problem. Next, we prove by induction on g that an instance with $m \geq 2^g r$ is a yes-instance. If $g = 0$, the claim is obvious. Otherwise suppose that $g \geq 1$ and every column's gap is g (this can be achieved by changing some arbitrary 1s to 0s). Now, since there are more 0s than 1s in the

matrix, there is a row having 0s in at least half of the columns. Changing these 0s to 1s and removing all other columns, we get an instance with $k' = k-1$, $g' = g-1$, and $m' \geq m/2 \geq 2^{g-1}r$, which is a yes-instance by induction hypothesis. This shows that our instance is also a yes-instance. □

Proposition 3 implies that PARTIAL LOBBYING can be solved efficiently if the gap and the number of issues to be approved are small. By similar technique, it can be solved efficiently if the columns do not differ too much:

Proposition 4. PARTIAL LOBBYING *is in* FPT *with respect to* (g', r) *with* g' *being the maximum Hamming distance of two columns.*

6 Conclusion

Some of our fixed-parameter tractability results are of pure classification nature. Hence, improvement on the time complexities of the corresponding fixed-parameter algorithms would be highly desirable. Whether PARTIAL LOBBYING is W[2]-hard for the parameter r, as well as the complexity of RESTRICTED LOBBYING in case of parameter values $t' = 2$ or $t' = 3$ are further open questions. Finally, we mention in passing that many of our results on LOBBYING can be adapted for the variants of LOBBYING where one either may add or delete votes.

References

Bartholdi III, J. J.; Tovey, C. A.; and 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.; and Rothe, J. 2011. How hard is it to bribe the judges? A study of the complexity of bribery in judgment aggregation. In *Proc. of 2nd ADT'11*, 1–15. Springer.

Binkele-Raible, D.; Erdélyi, G.; Fernau, H.; Goldsmith, J.; Mattei, N.; and Rothe, J. 2011. The complexity of probabilistic lobbying. Technical report, arXiv:0906.4431.

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

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

Dom, M.; Lokshtanov, D.; and Saurabh, S. 2009. Incompressibility through colors and IDs. In *Proc. of 36th ICALP'09*, volume 5555 of *LNCS*, 378–389. Springer.

Dorn, B., and Schlotter, I. 2012. Multivariate complexity analysis of swap bribery. *Algorithmica*. Available electronically.

Downey, R. G., and Fellows, M. R. 1999. *Parameterized Complexity*. Springer.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2010. Cloning in elections. In *Proc. of 24th AAAI'10*, 768–773. AAAI Press.

Erdélyi, G.; Hemaspaandra, L. A.; Rothe, J.; and Spakowski, H. 2007. On approximating optimal weighted lobbying, and frequency of correctness versus average-case polynomial time. In *Proc. of 16th FCT'07*, volume 4639 of *LNCS*, 300–311. Springer.

Erdélyi, G.; Fernau, H.; Goldsmith, J.; Mattei, N.; Raible, D.; and Rothe, J. 2009. The complexity of probabilistic lobbying. In *Proc. of 1st ADT'09*, volume 5783 of *LNCS*, 86–97. Springer.

Erdélyi, G.; Piras, L.; and Rothe, J. 2011. The complexity of voter partition in Bucklin and fallback voting: Solving three open problems. In *Proc. of 10th AAMAS'11*, 837–844.

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

Fellows, M. R. 2009. Towards fully multivariate algorithmics: Some new results and directions in parameter ecology. In *Proc. of 20th IWOCA'09*, volume 5874 of *LNCS*, 2–10. Springer.

Flum, J., and Grohe, M. 2006. *Parameterized Complexity Theory*. Springer.

Gabow, H. N. 1983. An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In *Proc. of the 15th STOC'83*, 448–456. ACM.

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

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

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

Mahajan, M., and 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 *Proc. of 27th STACS'10*, volume 5 of *Leibniz International Proceedings in Informatics*, 17–32.

Papadimitriou, C. H., and 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.; and Levine, D. 2002. Winner determination in combinatorial auction generalizations. In *Proc. of 1st AAMAS'02*, 69–76. ACM.

Schlotter, I.; Elkind, E.; and Faliszewski, P. 2011. Campaign management under approval-driven voting rules. In *Proc. of 25th AAAI'11*, 726–731. AAAI Press.

Schrijver, A. 2003. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer.

Szeider, S. 2011. Limits of preprocessing. In *Proc. of 25th AAAI'11*, 93–98. AAAI Press.