Mixed Integer Programming with Convex/Concave Constraints: Fixed-Parameter Tractability and Applications to Multicovering and Voting*

Robert Bredereck
TU Berlin
Berlin, Germany

Piotr Faliszewski
AGH University
Krakow, Poland

Rolf Niedermeier
TU Berlin
Berlin, Germany

Piotr Skowron
TU Berlin
Berlin, Germany

Nimrod Talmon
Weizmann Institute of Science
Rehovot, Israel

Abstract

A classic result of Lenstra [Math. Oper. Res. 1983] says that an integer linear program can be solved in fixed-parameter tractable (FPT) time for the parameterization by the number of variables. We extend this result by incorporating piecewise linear convex or concave functions to our (mixed) integer programs. This general technique allows us to analyze the parameterized complexity of a number of classic NP-hard computational problems. In particular, we prove that WEIGHTED SET MULTICOVER is in FPT when parameterized by the number of elements to cover, and that there exists an FPT-time approximation scheme for MULTISET MULTICOVER for the same parameter. Further, we use our general technique to prove that a number of problems from computational social choice (e.g., problems related to bribery and control in elections) are in FPT when parameterized by the number of candidates. For bribery, this resolves a nearly 10-year old family of open problems, and for weighted electoral control of Approval voting, this improves some previously known XP-memberships to FPT-memberships.

1 Introduction

The idea of parameterized complexity theory is to measure the difficulty of computational problems with respect to both the length of their encodings (as in standard complexity theory) and additional parameters (e.g., pertaining to the structure of the input). For example, in the SET MULTICOVER problem we are given a set of elements $U = \{x_1, \ldots, x_m\}$, a multiset $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets over U, integer covering requirements $r_1, \ldots, r_m$ for the elements of U, and a budget B. The question is whether it is possible to pick a collection of at most B sets from $\mathcal{S}$ so that each element

*A preliminary version of this paper appeared under the title "Elections with Few Candidates: Prices, Weights, and Covering Problems" in *Proceedings of the 4th International Conference on Algorithmic Decision Theory, ADT 2015* [7]. This journal version focuses on the theoretical part which is substantially revised and improved.

$x_i \in U$ belongs to at least r_i of them. This problem is a simple extension of the SET COVER problem and, thus, is NP-hard. There exists, however, an efficient algorithm for this problem when the cardinality of U is small. In fact, SET MULTICOVER is fixed-parameter tractable (is in FPT) with respect to the number of elements to cover: We have an algorithm that solves it in time $f(|U|) \cdot |I|^{O(1)}$, where $|I|$ is the length of the encoding of the given instance and f is a computable function (which depends only on the parameter; i.e., $|U|$ in our case).

This FPT algorithm for SET MULTICOVER proceeds by expressing the problem as an integer linear program (ILP) and solving it using the classic algorithm of Lenstra [39]. For each subset A of U, let $\mathcal{S}(A)$ be the subfamily of $\mathcal{S}$ that contains sets equal to A; use a variable x_A, intended to hold the number of sets from $\mathcal{S}(A)$ that we include in the solution; and define a constant $b_A = |\mathcal{S}(A)|$. Then, we introduce the following constraints:

$$x_A \leq b_A \quad \text{for each } A \subseteq U, \tag{1}$$

$$\sum_{A \subseteq U} x_A \leq B, \tag{2}$$

$$\sum_{A \subseteq U \colon x_i \in A} x_A \geq r_i \quad \text{for each } x_i \in U. \tag{3}$$

Constraints of the form (1) ensure that the solution is possible (i.e., we never use more sets of a given type than there are in the input), constraint (2) ensures that we use at most B sets, and constraints of the form (3) ensure that each element from U is covered a required number of times. We solve this ILP using the algorithm of Lenstra [39], which decides feasibility of ILPs in FPT time with respect to the number of integer variables.

Unfortunately, the above approach seems to fail for the case of WEIGHTED SET MULTICOVER, a variant of SET MULTICOVER where each set from $\mathcal{S}$ also has a weight and we can only choose sets of total weight at most B (SET MULTICOVER is the special case of WEIGHTED SET MULTICOVER where all the weights are equal to one). The reason for this apparent failure is that, for each $A \subseteq U$, the ILP for SET MULTICOVER treats all sets from $\mathcal{S}(A)$ as indistinguishable, but in WEIGHTED SET MULTICOVER they have weights, which give them "identities." Specifically, for the case of WEIGHTED SET MULTICOVER we would have to replace constraint (2) with one of the form:

$$\sum_{A \subseteq U} \mathrm{weight}_A(x_A) \leq B,$$

where $\mathrm{weight}_A(x)$ is a (convex) function that gives the sum of the lowest x weights of the sets from $\mathcal{S}(A)$. The main contribution of this paper is in showing a technique that allows us to include constraints of this type, where some variables are replaced by their "piecewise linear convex or concave functions", and still solve the resulting programs in FPT time using Lenstra's algorithm.

In Section 1.1.1 we compare our technique to similar tools known in the literature. We argue that one of the greatest benefits of our technique is that it is very simple to use. Indeed, to obtain an FPT algorithm for WEIGHTED SET MULTICOVER it suffices to make only very small changes to the algorithm for SET MULTICOVER and invoke our machinery. Second, our technique proceeds by constructing a mixed integer program from an integer linear program with aforementioned convex

constraints—this makes it easy to feed it into existing commercial solvers and to obtain an efficient practical algorithm with a relatively low effort. Finally, we consider that the main contribution of this work is that we illustrate the simplicity and usefulness of the discussed technique by applying it to a number of set-covering problems and by resolving the computational complexity status of a number of election-related problems parameterized by the number of candidates. These problems include, for example, various bribery problems [16, 18] and priced control problems [43] that were known to be in XP for nearly ten years, but were neither known to be fixed-parameter tractable, nor to be W[1]-hard. Indeed, in all these problems the voters had prices, which gave them "identities" in the same way as weights gave "identities" to sets in the WEIGHTED SET MULTICOVER problem. Nonetheless, our technique is not limited to the case of election problems with prices. For example, we show that it also applies to weighted voter control for Approval voting, improving results of Faliszewski et al. [20], and Faliszewski et al. [21] apply our technique to problems pertaining to finding winners according to several multiwinner election rules (Peters [44] and Caragiannis et al. [9] use a very similar technical trick).

We also demonstrate the usefulness of our technique in more technically demanding scenarios. In particular, we consider the MULTISET MULTICOVER problem, which generalizes SET MULTICOVER to the case of multisets, and we show that our general technique can be used as a component in the construction of an FPT-time approximation scheme for the problem (parameterized by the number of elements, there is no hope for an FPT exact algorithm as the problem is NP-hard even for two elements). In this case, our analysis combines combinatorial arguments with the technique of handling ILPs with piecewise-linear convex/concave transformations.

1.1 Related Work

Our work is related to three main research lines, regarding algorithms for integer linear programming, regarding various types of covering problems, and regarding algorithms for and complexity of voting-related problems. Below we review the main points of intersection with this literature.

1.1.1 Integer Linear Programming

Solving integer linear programs is one of the major approaches for tackling NP-hard problems [48]. Indeed, modern ILP solvers such as CPLEX or Gurobi can effectively deal with ILPs with thousands of variables and constraints; this makes them practical tools to solve even fairly large instances of computationally difficult problems. Yet, from our point of view, the most important aspect of integer linear programming is that it provides a powerful tool for designing FPT algorithms. The first major breakthrough in this direction was achieved by Lenstra [39], who showed that MIXED INTEGER LINEAR PROGRAMMING is fixed-parameter tractable with respect to the number n of the variables. Then Frank and Tardos [24] and Kannan [31] improved the corresponding running time bounds.

The algorithm of Lenstra is very useful, but for some problems its parameterization by the number of integer variables seems insufficient to obtain FPT algorithms (the SWAP BRIBERY problem [16] parameterized by the number of candidates is one particular example where this seems to be the case). Fortunately, one may use other approaches, such as the recently popularized n-fold

integer programming technique (n-fold IP). The main idea of n-fold integer programming is as follows: Our goal is to find a feasible solution of an inequality of the form $E \cdot x \leq b$, where E is a matrix, b is a vector of constants, and x is a vector of integer variables (all as in the classic ILP problem), but where E is restricted to be of a very special form (A and D are matrices):

$$E = \begin{pmatrix} D & D & D & \cdots & D \\ A & 0 & 0 & \cdots & 0 \\ 0 & A & 0 & \cdots & 0 \\ 0 & 0 & A & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & A \end{pmatrix}.$$

There is an algorithm that solves ILPs of this form in FPT time parameterized by the dimensions of the matrices A and D [40, 41, 27], without restricting the dimensions of matrix E (i.e., the constraints have a restricted form, but we can use as many integer variables as we like). In spite of this restrictive structure, n-fold integer programming found a number of applications, e.g., in scheduling [34] and voting [38]. See also the works of Dvorák et al. [15] and Knop et al. [35] for very recent generalizations and extensions of the this technique.

It is quite non-obvious how the technique of n-fold integer programming compares to ours. On the one hand, using n-fold IP it is possible to find FPT algorithms for problems for which our approach seems not to be applicable (such as SWAP BRIBERY parameterized by the number of candidates [38]). On the other hand, it is not clear if n-fold IP can be used in all cases where our method works (but we also cannot provide an obvious example where it fails and our approach works; as a side note, it seems as if proving its failure is an interesting hard task). While we stress that our approach is easier to use than that of n-fold integer programming, we also point out that Koutecký et al. [38] made significant progress in making n-fold IP more approachable. In particular, they described how to implement a number of primitives, and showed how using them makes the task of building n-fold programs much easier compared to directly designing the corresponding A and D matrices.

Further, we note that more general techniques than the one presented in this paper are known in the literature[1]. For instance, Dadush et al. [11] proved that an ILP can be solved in FPT time with respect to the number of integer variables, even if the constraints describe an arbitrary convex polyhedron, and are given through a separation oracle. One particular aspect in which the result of Dadush et al. [11] generalizes our technique is that it allows for using convex *multi-variate* functions in the constraints formulations. (A similar argument has been also given by Hildebrand and Köppe [28]—for the case when constraints are expressed as convex polynomials, and by Khachiyan and Porkolab [33]—for the case when the constraints are expressed as convex semialgebraic sets.[2])

[1] We thank Martin Koutecký for pointing out to us the most relevant works in this area.

[2] Some of these results are not commonly known among the community centered around parameterized complexity theory. For instance, Khachiyan and Porkolab claim to show a polynomial time algorithm for solving ILPs provided that the number of integer variables is constant; thus, this result appears to show an XP membership only. However, a more careful look at the paper allows to realize that Khachiyan and Porkolab in fact give an FPT algorithm. Again, we thank Martin Koutecký for fruitful discussions on clarifying these issues.

Gavenčiak et al. [25] give a comprehensive review of the advances in solving convex integer programs from the last two decades.

Yet, we believe that our approach has two advantages that make it preferable whenever it can be used. First, the technique explained by our proof is simpler, uses only elementary methods, and can be easily understood without the necessity of learning quite advanced tools. Second, our technique reduces ILP programs with convex constraints to standard MILP programs, which allows to easily use cutting edge off-the-shelf MILP solvers with the more general types of constraints. Thus, due to our results one may obtain both a theoretical FPT guarantee and a practical algorithm.

1.1.2 Covering

The class of covering problems is of fundamental importance in theoretical computer science because, on the one hand, covering problems are abstractions of many real-life issues, and, on the other hand, they exhibit very interesting algorithmic behavior. For example, the SET COVER problem, arguably the best known representative of the class, was among the first problems shown to be NP-complete (in Karp's seminal paper [32]), and later it was thoroughly studied from the points of view of (in)approximability [52] and parameterized complexity [14]. For example, it is known to be W[2]-hard for the parameterization by the solution size (indeed, it is one of the classic W[2]-hard problems), but it is fixed-parameter tractable with respect to the number of elements. Covering problems have various applications in domains such as software engineering (e.g., covering scenarios by few test cases), antivirus development (looking for a set of suspicious byte strings which covers all known viruses), databases (finding a set of labels which covers all data items), to name just a few.

There is a vast literature on the SET COVER problem and, thus, we only briefly point out selected results. It is known that a simple greedy algorithm gives a $\log(m)$ approximation guarantee, where m is the size of the set to be covered (this algorithm was given, e.g., by Johnson [30], but see the textbook of Vazirani [51] for further references). It is also known that unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time algorithm can approximate the problem with a better ratio [23, 12]. The variant of the problem where each element appears in at most f sets can be approximated with the ratio f [51]. Parameterized approximation algorithms for the problem were considered by Bonnet et al. [4], Skowron and Faliszewski [50] and Skowron [49].

SET MULTICOVER and MULTISET MULTICOVER also received some attention in the literature. For example, Berman et al. [3] considered approximability of the former problem and Rajagopalan and Vazirani [45] studied the same issue for the latter (and, in general, for various covering problems). Exact algorithms for these problems were studied by Hua et al. [29]. Approximation algorithms for covering integer programs were considered by Kolliopoulos [36] and Kolliopoulos and Young [37].

1.1.3 Voting

Computational social choice (COMSOC) is an interdisciplinary area that spans computer science, economics, and operations research, and whose goal is to (computationally) analyze group decision-making processes [46, 5]. From our point of view, the most relevant part of COMSOC regards the

complexity of various election-related problems. We mostly focus on the problems of manipulating election outcomes through bribery and control [17].

In the simplest variant of the bribery problem [18], we are given a description of an election (i.e., a set of candidates and a collection of voters, with their votes represented in some way) and an integer k. We ask if it is possible to ensure that a given candidate becomes a winner by bribing at most some k voters (i.e., by changing the votes of at most k voters). However, there are many other variants where, for example, each voter may have a different price for being bribed [18], the prices may depend on the extent to which we change given votes [16], or where the votes are represented using some involved language [42]. Initially, bribery problems were supposed to model undesirable behavior, but later researchers realized that they also capture perfectly legal actions, such as campaigning [2, 8, 13, 47], fraud detection [53], or analysis of candidate performance [22]. Control problems are similar in spirit to the bribery ones, but instead of modifying votes, we are allowed to add/delete either candidates or voters [1, 26].

We focus on the case where we have a few candidates but (possibly) many voters. This is a very natural setting and it models many real-life scenarios such as political elections or elections among company stockholders. The complexity of manipulating elections with few candidates is, by now, very well understood. On the one hand, if the elections are weighted (as is the case for the elections held by company stockholders), then our problems are typically NP-hard even if the number of candidates is a small fixed constant [10, 18, 20]; these results typically follow by reductions from the well-known NP-hard PARTITION problem. One particular example where we did not have NP-hardness for the setting with the fixed number of candidates was control by adding/deleting voters under the Approval and k-Approval voting rules. For this problem Faliszewski at al. [20] have shown XP membership, but could neither prove W[1]-hardness nor give an FPT-algorithm; in this paper we resolve this problem by proving fixed-parameter tractability.

If we consider parameterization by the number of candidates but the elections are unweighted (as is the case for political elections) and no prices are involved, then we typically get FPT results. These results are often obtained by expressing the respective problems as integer linear programs (ILPs) and then applying Lenstra's algorithm [39]. In essence, these results are of the same nature as the FPT algorithm for the SET MULTICOVER problem given at the beginning of the introduction. The main missing piece in our understanding of the complexity of manipulating elections with few candidates regards those unweighted-election problems where each voter has some sort of price (for example, as in the bribery problems). In this paper we almost completely fill this gap by showing a general approach for proving FPT membership for a class of bribery-like problems parameterized by the number of candidates, for unweighted elections.

1.2 Organization

The paper is organized as follows. First, in Section 2, we describe our technique of handling ILPs with piecewise-linear convex/concave transformations. Then, in Section 3, we show examples of applying it to covering problems and to problems regarding Approval voting (which are very close in spirit to covering problems). In Section 4 we show how further election-related problems, beyond Approval voting, can be solved using our approach, and we conclude in Section 5.

2 Mixed Integer Linear Programming with Piecewise Linear Convex/Concave Functions

We now describe our technique of solving integer linear programs with piecewise linear convex/concave transformations. The technique is based on using rational-valued variables to simulate the behavior of convex/concave transformations and it uses as a subroutine an algorithm for solving the MIXED INTEGER LINEAR PROGRAMMING problem, as defined below.

> MIXED INTEGER LINEAR PROGRAMMING (MIP)
> **Input:** An $m \times (n+r)$ matrix A with integer elements and a length-m integer vector b of rational numbers.
> **Question:** Is there a length $(n+r)$ vector $x = (x_1, \ldots, x_{n+r})$ such that $A \cdot x \leq b$, where values $x_1, \ldots, x_n$ are required to be integers, but values $x_{n+1}, \ldots, x_{n+r}$ can be rational?

We use standard terminology and syntax from linear programming. In particular, we interpret the entries of vector x as *variables* and the rows of matrix A as *constraints*. Formally, we require matrix A and vector b to have integer entries, but it would be straightforward to allow them to have rational values (indeed, we implicitly assume that whenever we obtain matrix A or vector b with rational values, they are transformed to the integer form by appropriate multiplication).

Formally, the MIP problem captures the issue of testing whether *some* feasible solution exists for a given mixed integer program. If we want a solution that maximizes a certain objective function of the form $c_1x_1 + c_2x_2 + \cdots + c_{n+r}x_{n+r}$ (where $c_1, \ldots, c_{n+r}$ are integers), then we can use the standard trick of including the constraint:

$$c_1x_1 + c_2x_2 + \cdots + c_{n+r}x_{n+r} \geq T$$

in the program and performing a binary search for the largest value of T for which a feasible solution exists. If M is the value of the objective function for the optimal solution, then this requires solving $\log M$ programs instead of one. Typically this is a perfectly acceptable price to pay.

The following result, due to Frank and Tardos [24] and Kannan [31], gives the FPT algorithm for the MIP problem parameterized by the number of integer variables (the result is built on top of Lenstra's original algorithm [39]).

Theorem 1 (Lenstra [39], Frank and Tardos [24], and Kannan [31])**.** *There is an algorithm that solves* MIXED INTEGER LINEAR PROGRAMMING *in* $O(n^{2.5n+o(n)} \cdot |I|)$ *time, where* $|I|$ *is the number of bits encoding the input and* n *is the number of integer variables.*

Below we describe how the MIP problem and the above theorem can be extended to the case where each variable is replaced by some piecewise linear convex/concave transformation, while still maintaining the FPT running time.

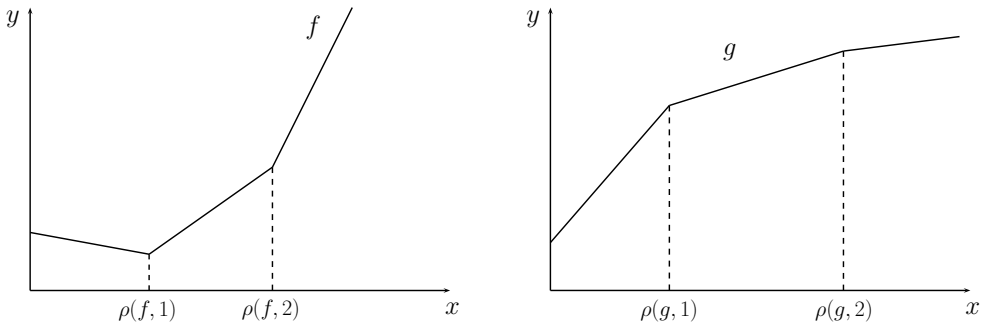

Figure 1: An example of a piecewise linear convex function f (left plot) and a piecewise linear concave function g (right plot) with three pieces.

Piecewise Linear Convex/Concave Transformations. We consider two simple types of piecewise linear transformations, *piecewise linear convex functions* and *piecewise linear concave functions*. A piecewise linear convex function is a continuous convex function, defined on a finite sequence of intervals (that together give the set of all real numbers) so that for each of the intervals, the function restricted to this interval is linear. Piecewise linear concave functions are defined analogously, except that they are concave instead of convex. We present examples of functions of this type in Figure 1.

For a piecewise linear convex/concave function $f : \mathbb{R} \to \mathbb{R}$, we denote the decomposition of its domain into a minimal number of disjoint intervals on which it is linear as follows:

$$\mathbb{R} = (-\infty, \rho(f,1)] \cup (\rho(f,1), \rho(f,2)] \cup (\rho(f,2), \rho(f,3)] \cup \ldots \cup (\rho(f,\ell), \infty).$$

We say that such a function f consists of $\ell + 1$ pieces and we define $\mathrm{pieces}(f)$ to be $\ell + 1$. We refer to the function f restricted to interval $(-\infty, \rho(f,1)]$ as the zeroth piece of f, to the function f restricted to interval $[\rho(f,1), \rho(f,2)]$ as to the first piece of f, to the function f restricted to interval $[\rho(f,2), \rho(f,3)]$ as to the second piece of f, and so on. By $\mathrm{der}(f,i)$ we denote the (constant) derivative of the i-th piece of f.

For technical reasons we also require that for each piecewise linear convex/concave function we have that (a) $f(0)$ is integer, (b) for each piece i, $0 \leq i \leq \mathrm{pieces}(f)$, the derivative $\mathrm{der}(f,i)$ is integer, and (c) for each i, $1 \leq i \leq \mathrm{pieces}(f) - 1$, $\rho(f,i)$ is an integer. These requirements are technical only and, for example, we could replace all the occurrences of the word "integer" with "rational value" and all our results would still hold (this is analogous to the fact that we could define the MIP problem to work with rational values in the matrix A and vector b).

Mixed Integer Programs with Simple Linear Convex/Concave Transformations. The following problem captures our extension of the MIXED INTEGER LINEAR PROGRAMMING problem and will be our central technical tool in the following sections.

MIXED INTEGER PROGRAMMING WITH SIMPLE PIECEWISE LINEAR TRANSFORMATIONS (MIP/SPLIT)

Input: A collection of $(n+r)m$ piecewise linear convex functions $F = \{f_{i,j} \colon 1 \leq i \leq (n+r), 1 \leq j \leq m\}$, a collection of $(n+r)m$ piecewise linear concave functions $G = \{g_{i,j} \colon 1 \leq i \leq (n+r), 1 \leq j \leq m\}$, and a vector $b \in \mathbb{Z}^m$.
Question: Is there a vector x such that

$$\begin{aligned}
\sum_{i=1}^{n+r} f_{i,1}(x_i) &\leq \sum_{i=1}^{n+r} g_{i,1}(x_i) \ + b_1, \\
\sum_{i=1}^{n+r} f_{i,2}(x_i) &\leq \sum_{i=1}^{n+r} g_{i,2}(x_i) \ + b_2, \\
&\vdots \\
\sum_{i=1}^{n+r} f_{i,m}(x_i) &\leq \sum_{i=1}^{n+r} g_{i,m}(x_i) + b_m, \\
x_i &\in \mathbb{N} \qquad \text{for } 1 \leq i \leq n, \\
x_i &\in \mathbb{R}^+ \qquad \text{for } n+1 \leq i \leq n+r?
\end{aligned}$$

Below we describe how to use Theorem 1 to solve MIP/SPLIT with up to polynomial factors the same asymptotic time complexity as MIP.

Theorem 2. *There is an algorithm that solves* MIXED INTEGER PROGRAMMING WITH SIMPLE PIECEWISE LINEAR TRANSFORMATIONS *in $O(n^{2.5n+o(n)} \cdot (|I| + p_{\max})^{O(1)})$ time, where n is the number of integer variables, $p_{\max}$ is the maximum number of pieces per function, and $|I|$ is the number of bits encoding the input.*

Proof. To prove the theorem, we will reduce MIP/SPLIT to MIP, by replacing each non-linear constraint with a polynomial number of linear ones, using polynomially many additional real-valued variables (but without changing the number of integer ones). This will allow us to invoke Theorem 1 to solve the resulting program.

Note that in the MIP/SPLIT problem (as in the MIP one), we consider the canonical form, where all variables are nonnegative. Hence, we can assume that the zeroth piece of each function $f_{i,j}$ and each function $g_{i,j}$ includes point 0 (pieces covering only negative points are irrelevant). Furthermore, by appropriately setting the b_j coefficients, we can also assume that for each i and j we have $f_{i,j}(0) = 0$ and $g_{i,j}(0) = 0$.

Our reduction starts with the input MIP/SPLIT instance and successively transforms it into an ordinary MIP instance. We keep all integer variables $x_1, \ldots, x_n$ and all real-valued variables $x_{n+1}, \ldots, x_{n+r}$ of the original MIP/SPLIT instance, but we introduce additional real-valued variables and we replace non-linear constraints with a number of linear ones.

Replacing a Non-Linear Constraint. Let j be an integer such that the non-linear constraint

$$\sum_{i=1}^{n+r} f_{i,j}(x_i) \leq \sum_{i=1}^{n+r} g_{i,j}(x_i) + b_j \tag{4}$$

has not yet been replaced with linear ones. We remove it from the program and in its place we include constraint (where $w_{i,j}$ and $u_{i,j}$ are new real-valued variables):

$$\sum_{i=1}^{n+r} w_{i,j} \leq \sum_{i=1}^{n+r} u_{i,j} + b_j. \tag{5}$$

We will include additional constraints so that variables $w_{i,j}$ will upper-bound values $f_{i,j}(x_i)$ and variables $u_{i,j}$ will lower-bound values $g_{i,j}(x_i)$ (and, indeed, we will be able to assume that these variables have exactly the values $f_{i,j}(x_i)$ and $g_{i,j}(x_i)$, respectively).

Additional Variables. To ensure that variables $w_{i,j}$ and $u_{i,j}$ have correct values (in the feasible solution for our program), we need additional variables. For each $i \in [n+r]$, we introduce $\mathrm{pieces}(g_{i,j})$ real-valued variables $y_{i,j,1}, \ldots, y_{i,j,\mathrm{pieces}(g_{i,j})}$ and $\mathrm{pieces}(f_{i,j})$ real-valued variables $z_{i,j,1}, \ldots, z_{i,j,\mathrm{pieces}(f_{i,j})}$. We will ensure that if there is a feasible solution to our program then there is one where $y_{i,j,\ell} = \max(0, x_i - \rho(g_{i,j}, \ell))$ and $z_{i,j,\ell} = \max(0, x_i - \rho(f_{i,j}, \ell))$. In other words, for each ℓ, the variable $y_{i,j,\ell}$ (resp. $z_{i,j,\ell}$) measures how far the variable x_i is beyond the beginning of the ℓ-th piece of function $g_{i,j}$ (resp. of function $f_{i,j}$).

Constraining Variables $w_{i,j}$ (Convex Case). First, for each variable $z_{i,j,\ell}$ we introduce two constraints:

$$\begin{aligned} z_{i,j,\ell} &\geq 0, \\ z_{i,j,\ell} &\geq x_i - \rho(f_{i,j}, \ell). \end{aligned} \tag{6}$$

Second, for each $i \in [n+r]$ we introduce the constraint:

$$x_i \cdot \mathrm{der}(f_{i,j}, 0) + \sum_{\ell=1}^{\mathrm{pieces}(f_{i,j})} z_{i,j,\ell} \cdot (\mathrm{der}(f_{i,j}, \ell) - \mathrm{der}(f_{i,j}, \ell - 1)) \leq w_{i,j}. \tag{7}$$

Constraining Variables $u_{i,j}$ (Concave Case). Analogously to the convex case, we first introduce two constraints for each variable $y_{i,j,\ell}$ (these two constraints are almost identical to the convex case):

$$\begin{aligned} y_{i,j,\ell} &\geq 0, \\ y_{i,j,\ell} &\geq x_i - \rho(g_{i,j}, \ell). \end{aligned} \tag{8}$$

Second, for each $i \in [n+r]$ we introduce the constraint:

$$u_{i,j} \leq x_i \cdot \mathrm{der}(g_{i,j}, 0) + \sum_{\ell=1}^{\mathrm{pieces}(g_{i,j})} y_{i,j,\ell} \Big(\mathrm{der}(g_{i,j}, \ell) - \mathrm{der}(g_{i,j}, \ell - 1) \Big). \tag{9}$$

Correctness. We now argue that if there is a feasible solution for our transformed MIP instance, then there also is one for the original MIP/SPLIT instance. Let us assume some feasible solution for the transformed instance and focus on some arbitrary constraint number j (recall Equations (4) and (5)).

In order to satisfy Constraint (5), smaller values of $w_{i,j}$ are clearly more desirable. Since each $w_{i,j}$ only occurs once on the left-hand side in Constraint (5) and once on the right-hand side in one of the constraints of the from (7), we can assume that each constraint of the form (7) is satisfied with equality (given a feasible solution, we can keep on decreasing the values $w_{i,j}$ until we hit equalities in these constraints). Further, together with the fact that $f_{i,j}$ is convex, and consequently $\mathrm{der}(f_{i,j}, \ell) > \mathrm{der}(f_{i,j}, \ell - 1)$ for each ℓ, we infer that the values $z_{i,j,\ell}$ can be as small as possible. Formally, similarly as above, using constraints of the form (6), we can assume that in our feasible solution for each variable $z_{i,j,\ell}$ it holds that $z_{i,j,\ell} = \max(0, x_i - \rho(f_{i,j}, \ell))$. Consequently, we conclude that there is a feasible solution where:

$$w_{i,j} = x_i \cdot \mathrm{der}(f_{i,j}, 0) + \sum_{\ell=1}^{\mathrm{pieces}(f_{i,j})} \max(0, x_i - \rho(f_{i,j}, \ell)) \cdot (\mathrm{der}(f_{i,j}, \ell) - \mathrm{der}(f_{i,j}, \ell - 1)).$$

Let us now analyze the value $w_{i,j}$ provided by this equality. If $x_i = 0$, then we surely have $w_{i,j} = 0 = f_{i,j}(0)$. Next, we analyze how the value of $w_{i,j}$ changes when we increase x_i to Δ. If x_i does not exceed $\rho(f_{i,j}, 0)$, then $w_{i,j}$ has value $\Delta \cdot \mathrm{der}(f_{i,j}, 0)$ and we have $w_{i,j} = f(\Delta)$. If x_i is greater than $\rho(f_{i,j}, 0)$ but smaller than $\rho(f_{i,j}, 1)$, then $w_{i,j}$ has value $\Delta \cdot \mathrm{der}(f_{i,j}, 0) + (\Delta - \rho(f_{i,j}, 0)) \cdot (\mathrm{der}(f_{i,j}, 1) - \mathrm{der}(f_{i,j}, 0)) = f_{i,j}(\Delta)$. By analogous reasoning, we obtain $w_{i,j} = f_{i,j}(x_i)$ for every value of x_i.

Analogous reasoning shows that we can also assume that under our feasible solution it holds that $u_{i,j} = g_{i,j}(x_i)$. Specifically, we note that to satisfy Constraint (5), larger values of $u_{i,j}$ are clearly more desirable. Since each $u_{i,j}$ only occurs once on the right-hand side in Constraint (5) and once on the left-hand side in one constraint of the form (9), we infer that each constraint of the form (9) can be satisfied with equality. Further, together with the fact that $g_{i,j}$ is concave, and consequently $\mathrm{der}(g_{i,j}, \ell) < \mathrm{der}(g_{i,j}, \ell - 1)$ for each ℓ, we infer that the values $y_{i,j,\ell}$ can be as small as possible. Formally, similarly as above, we infer from constraints of the form (8) that we can assume that in our feasible solution for each variable $y_{i,j,\ell}$ it holds that $y_{i,j,\ell} = \max(0, x_i - \rho(g_{i,j}, \ell))$. Consequently, we have that:

$$u_{i,j} = x_i \cdot \mathrm{der}(g_{i,j}, 0) + \sum_{\ell=1}^{\mathrm{pieces}(g_{i,j})} \max(0, x_i - \rho(g_{i,j}, \ell)) \cdot \Big(\mathrm{der}(g_{i,j}, \ell) - \mathrm{der}(g_{i,j}, \ell - 1)\Big).$$

Analysis analogous to that for the case of variables $w_{i,j}$ shows that, indeed, we have $u_{i,j} = g_{i,j}(x_i)$.

The above reasoning, together with the fact that constraints of the form (5) are satisfied and clearly correspond to the original constraints in the MIP/SPLIT problem, proves that if there is a feasible solution for the transformed instance, then there also is one for the original instance. From our reasoning it is also apparent that the other implication holds (given variables x_i, it suffices that for each i, j, and ℓ we set $w_{i,j} = f_{i,j}(x_i)$, $u_{i,j} = g_{i,j}(x_i)$, $z_{i,j,\ell} = \max(0, x_i - \rho(f_{i,j}, \ell))$, and $y_{i,j,\ell} = \max(0, x_i - \rho(g_{i,j}, \ell))$).

Running Time. The running-time of the algorithm from Theorem 1 invoked on our transformed instance is upper-bounded by $n^{2.5 \cdot n + o(n)} \cdot |I^*|^{O(1)}$, where n denotes the number of integer variables and $|I^*|$ is the number of bits needed to encode our MIP. Finally, $|I^*|^{O(1)}$ can be upper-bounded by $(|I| + p_{\max})^{O(1)}$ since we introduced at most $O(p_{\max} \cdot (n+r))$ additional constraints and variables. This completes the proof. □

We conclude this section with two observations regarding the generality of Theorem 2. First, note that in this section we used the canonical form of MIP/SPLIT, requiring all the variables to be non-negative. Yet, as long as we do not actually use the piecewise linear transformations on a variable x_i (that is, as long as for each j, functions $f_{i,j}$ and $g_{i,j}$ are linear) we can use the standard technique of replacing each occurrence of x_i with $x_i^+ - x_i^-$, where x_i^+ and x_i^- are two nonnegative variables denoting, respectively, the positive and the negative part of x_i. In other words, we may allow negative values for each variable x_i whose associated functions $f_{i,j}$ and $g_{i,j}$ are simply linear.

Second, note that if a certain function $f_{i,j}$ is applied to an integer variable x_i and we know that the value of x_i must come from some set $\{0, \ldots, t_i\}$, then it suffices that $f_{i,j}$ is convex/concave on integer arguments, and we do not have to worry about it being piecewise linear. This last condition is vacuously satisfied: It suffices to consider pieces $[0,1), [1,2), [2,3), \ldots, [t_i, t_i+1)$, and derivatives $\mathrm{der}(f,0) = f(1) - f(0)$, $\mathrm{der}(f,1) = f(2) - f(1)$, $\mathrm{der}(f,2) = f(3) - f(2)$, and so on. It turns out that in many applications we apply convex/concave functions to integer variables only and this observation comes handy.

3 Covering and Approval Voting: Showcases of the Technique

In this section we demonstrate how to apply Theorem 2 to resolve the complexity of a number of problems related to covering and approval voting. These are interesting because the complexity of some of the problems we consider was open for the last ten years or so. We also show that Theorem 2 is useful in designing an FPT approximation scheme for the MULTISET MULTICOVER problem. This result is interesting because it is somewhat involved technically and illustrates mixing of our MIP/SPLIT-based approach with combinatorial arguments.

3.1 Weighted Multiset Multicover with Small Universe

We start by focusing on the complexity of a few generalizations of the MAX COVER problem. If A is a multiset and x is some element, then we write $A(x)$ to denote the number of times x occurs in A (that is, $A(x)$ is x's multiplicity in A). If x is not a member of A, then $A(x) = 0$.

Definition 1. *In the* WEIGHTED MULTISET MULTICOVER *(WMM) problem we are given a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of multisets over the universe $U = \{x_1, \ldots, x_m\}$, integer weights $w_1, \ldots, w_n$ for the multisets, integer covering requirements $r_1, \ldots, r_m$ for the elements of the universe, and an integer budget B. We ask whether there exists a subfamily $\mathcal{S}' \subseteq \mathcal{S}$ of multisets from $\mathcal{S}$ such that:*

1. *for each $x_i \in U$ it holds that $\sum_{S_j \in \mathcal{S}'} S_j(x_i) \geq r_i$ (that is, each element x_i is covered at least the required number of times), and*

2. $\sum_{S_j \in \mathcal{S}'} w_j \leq B$ *(the budget is not exceeded).*

We will show how the complexity of WMM (parameterized with respect to the universe size) changes as we keep on adding restrictions. First, we observe that a straightforward polynomial-time reduction from PARTITION proves that WMM is NP-hard even for the case of a single-element universe. Clearly, this also means that the problem is Para-NP-hard with respect to the number of elements in the universe as the parameter.

Proposition 1. WMM *is* NP*-complete even for the case of a single-element universe.*

Proof. Membership in NP is clear. We show NP-hardness by a reduction from the PARTITION problem. An instance of PARTITION consists of a sequence of nonnegative integers $k_1, \ldots, k_n$. We ask if there is a set $I \subseteq [n]$ such that $\sum_{i \in I} k_i = \frac{1}{2} \sum_{i=1}^{n} k_i = \sum_{i \notin I} k_i$.

We form an instance of WEIGHTED MULTISET MULTICOVER as follows. The universe contains a single element x with covering requirement equal to $\frac{1}{2} \sum_{i=1}^{n} k_i$. For each i, $1 \leq i \leq n$, there is a single multiset S_i containing k_i occurrences of x, with weight k_i. We set the budget to be $\frac{1}{2} \sum_{i=1}^{n} k_i$. Clearly, it is possible to cover x sufficiently many times if and only if our input instance of PARTITION is a "yes"-instance. □

Another variant of WMM is MULTISET MULTICOVER, where we assume each set to have unit weight. By generalizing the proof for Proposition 1, we show that this problem is NP-hard already for two-element universes, which again implies Para-NP-hardness with respect to the number of elements in the universe.

Proposition 2. MULTISET MULTICOVER *is* NP*-complete even for universes of size two.*

Proof. Membership in NP is clear. To show NP-hardness, we give a reduction from a variant of the SUBSET SUM problem. We are given a sequence $k_1, \ldots, k_{2n}$ of positive integers, a target value T, and we ask if there is a set $I \subseteq \{1, \ldots, 2n\}$ such that (a) $\sum_{i \in I} k_i = T$, and (b) $\|I\| = n$.

Let K be $\max(k_1, \ldots, k_{2n})$. We form an instance of MULTISET MULTICOVER that contains two elements, x_1 and x_2. For each i, $1 \leq i \leq 2n$, we form a set S_i that contains x_1 with multiplicity k_i, and x_2 with multiplicity $nKT - k_i$. We set the covering requirement r_1 of x_1 to be T, and the covering requirement r_2 of x_2 to be $n^2KT - T$. We ask if there is a multiset multicover of size at most n.

Clearly, if there is a solution for our SUBSET SUM instance, then the sets that correspond to this solution form a multiset multicover of x_1 and x_2. On the contrary, assume that there is a collection of at most n sets that form a multiset multicover of x_1 and x_2. There must be exactly n of these sets. Otherwise, the sum of their multiplicities for x_2 would be smaller than $n^2KT - T$. Due to the covering requirement of x_1, these sets correspond to the numbers from $\{x_1, \ldots, x_{2n}\}$ that sum up to at least T, and due to covering requirement of x_2, these sets correspond to numbers that sum up to at most T. This completes the proof. □

Often we do not need the full flexibility of WMM. For instance, in the next section we will describe several problems from computational social choice that can be reduced to more specific

variants of WMM, such as the WEIGHTED SET MULTICOVER and UNIFORM MULTISET MULTICOVER problems. WEIGHTED SET MULTICOVER is a variant of WMM where each input multiset has elements with multiplicities 0 or 1 (in other words, the family $\mathcal{S}$ contains sets without multiplicities, but the union operation takes multiplicities into account). UNIFORM MULTISET MULTICOVER is a variant of MULTISET MULTICOVER (and, thus, of WMM), where for each multiset S_i in the input instance there is a number t_i such that for each element x we have $S_i(x) \in \{0, t_i\}$ (in other words, elements within a single multiset have the same multiplicities).

As the first application of our new framework, we show that WEIGHTED SET MULTICOVER is fixed-parameter tractable when parameterized by the universe size. Notably, we only use convex constraints in the construction of the MIP/SPLIT instance.

Theorem 3. WEIGHTED SET MULTICOVER *is fixed-parameter tractable when parameterized by the universe size.*

Proof. Consider an instance of WEIGHTED SET MULTICOVER with universe $U = \{x_1, \ldots, x_m\}$, family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets, weights $w_1, \ldots, w_n$ for the sets, covering requirements $r_1, \ldots, r_m$ for the elements, and budget B. Our algorithm proceeds by solving an appropriate MIP/SPLIT instance.

First, we form a family $U_1, \ldots, U_{2^m}$ of all subsets of U. For each i, $1 \leq i \leq 2^m$, let $\mathcal{S}(U_i) := \{S_j \in \mathcal{S} \mid S_j = U_i\}$. For each i and j, $1 \leq i \leq 2^m$ we define a convex function f_i so that for each integer j, $1 \leq j \leq |\mathcal{S}(U_i)|$, $f_i(j)$ is the sum of the j lowest weights of the sets from $\mathcal{S}(U_i)$. We have 2^m integer variables z_i, $1 \leq i \leq 2^m$. Intuitively, these variables describe how many sets we take from each type (i.e., how many sets we take from each family $\mathcal{S}(U_i)$). We introduce the following constraints: For each i, $1 \leq i \leq 2^m$, we have constraints $z_i \geq 0$ and $z_i \leq |\mathcal{S}(U_i)|$. For each element x_ℓ of the universe, we also have constraint $\sum_{U_i \colon x_\ell \in U_i} z_i \geq r_\ell$. These constraints ensure that the variables z_i describe a possible solution for the problem (disregarding the budget). Our last constraint uses variables z_i to express the requirement that the solution has cost at most B:

$$\sum_{i=1}^{2^m} f_i(z_i) \leq B.$$

Finally, we use Theorem 2 to get the statement of the theorem. □

As a second application of our new framework, we show that UNIFORM MULTISET MULTICOVER is fixed-parameter tractable when parameterized by the universe size. Notably, we only use concave constraints in the corresponding MIP/SPLIT instance.

Theorem 4. UNIFORM MULTISET MULTICOVER *is fixed-parameter tractable when parameterized by the universe size.*

Proof. Consider an instance of UNIFORM MULTISET MULTICOVER with universe $U = \{x_1, \ldots, x_m\}$, family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets, covering requirements $r_1, \ldots, r_m$ for the elements, and budget B. Our algorithm proceeds by solving an appropriate MIP/SPLIT instance.

Similarly as in the proof of Theorem 3, we form a family $U_1, \ldots, U_{2^m}$ of all the subsets of U (note that these, indeed, are subsets and not multisets). For each i, $1 \leq i \leq 2^m$, let $\mathcal{S}(U_i)$ be a subfamily of S that contains those multisets in which exactly the elements from U_i appear (that

is, their multiplicities are non-zero). For each i, $1 \leq i \leq 2^m$, we define a concave function f_i so that for each j, $1 \leq j \leq |\mathcal{S}(U_i)|$, $f_i(j)$ denotes the maximum sum of multiplicities for each element from U_i using j multisets from $\mathcal{S}(U_i)$. (To compute this function, we simply need to sort the multisets from $\mathcal{S}(U_i)$ in the order of decreasing multiplicities. Then, $f_i(j)$ is the sum of the multiplicities with respect to an arbitrary element from U_i of the first j multisets.)

We have 2^m integer variables z_i, $1 \leq i \leq 2^m$. Intuitively, the z_i variables describe how many multisets we take from each type. Thus, $f_i(z_i)$ describes how much each element from U_i is covered by taking z_i multisets of type U_i. We introduce the following constraints: For each i, $1 \leq i \leq 2^m$, we have constraints $z_i \geq 0$ and $x_i \leq |\mathcal{S}(U_i)|$. For each element x_ℓ of the universe, we also have constraint $\sum_{U_i\colon x_\ell \in U_i} f_i(z_i) \geq r_\ell$. These constraints ensure that the variables z_i describe a possible solution for the problem (disregarding the budget). To express the requirement that the solution has cost at most B, we add the constraint $\sum_{i=1}^{2^m} z_i \leq B$. Finally, we use Theorem 2 to get the statement of the theorem. □

Unfortunately, it is impossible to apply our approach to the more general MULTISET MULTICOVER; by Proposition 2, MULTISET MULTICOVER is already NP-hard for two-element universes. It is, however, possible to obtain a certain form of an FPT approximation scheme.

Definition 2. *Let ϵ be a real number, $\epsilon > 0$. We say that algorithm $\mathcal{A}$ is an* ϵ-almost-cover algorithm *for* MULTISET MULTICOVER *if, given an input instance I with universe $U = \{x_1, \ldots, x_m\}$ and covering requirements $r_1, \ldots, r_m$, it outputs a solution that covers each element x_i with multiplicity r'_i such that $\sum_i \max(0, r_i - r'_i) < \epsilon \sum_i r_i$.*

In other words, on the average an ϵ-almost-cover algorithm can miss each element of the universe by an ϵ-fraction of its covering requirement. For the case where we really need to cover all the elements perfectly, we might first run an ϵ-almost-cover algorithm and then complement its solution, for example, in some greedy way, since the remaining instance might be much easier to solve.

The key idea regarding computing an ϵ-almost-cover is that it suffices to replace each input multiset by several sub-multisets, each with a particular "precision level," so that multiplicities of the elements in each sub-multiset are of a similar order of magnitude. The full argument, however, forms the most technical part of our paper.

Theorem 5. *For every rational $\epsilon > 0$, there is an* FPT*-time ϵ-almost-cover algorithm for* MULTISET MULTICOVER *parameterized by the universe size.*

Proof. We describe our ϵ-almost-cover algorithm for MULTISET MULTICOVER and argue its correctness. We consider an instance I of MULTISET MULTICOVER with a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of multisets over the universe $U = \{x_1, \ldots, x_m\}$, where the covering requirements for the elements of the universe are $r_1, \ldots, r_m$. We associate each set S from the family $\mathcal{S}$ with the vector $v_S = \langle S(x_1), S(x_2), \ldots, S(x_m)\rangle$ of element multiplicities.

Let $\epsilon > 0$ be the desired approximation ratio. We fix $Z = \lceil \frac{4m}{\epsilon} \rceil$ and $Y = Z + \lceil \frac{4Zm^3}{\epsilon} \rceil$. Note that $\frac{m}{Z} \leq \frac{\epsilon}{4}$ and $\frac{Zm^3}{Y-Z} \leq \frac{\epsilon}{4}$. Let $X = \left(\frac{2Y^m}{\epsilon} + 1\right)^m$ and let $V_1, \ldots, V_X$ be a sequence of all m-dimensional vectors whose entries come from the $\left(\frac{2Y^m}{\epsilon} + 1\right)$-element set $\left\{0, \frac{\epsilon}{2}, \epsilon, \frac{3\epsilon}{2}, 2\epsilon, \ldots, Y^m\right\}$. For each j, $1 \leq j \leq X$, we write $V_j = \langle V_j(x_1), V_j(x_2), \ldots, V_j(x_m)\rangle$. Intuitively, these

vectors describe some subset of "shapes" of all possible multisets—interpreted as vectors of multiplicities—over our m-element universe. For each number β, we write βV_i to mean the vector $\langle \lfloor \beta V_{i,1} \rfloor, \lfloor \beta V_{i,2} \rfloor, \ldots, \lfloor \beta V_{i,m} \rfloor \rangle$.

Vectors of the form βV_i are approximations of those multisets for which the positive multiplicities of the elements do not differ too much (formally, for those multisets for which the positive multiplicities differ by at most a factor of Y^m). Indeed, for each such multiset S, we can find a value β and a vector V_j such that for each element x_i it holds that $S(x_i) \geq \beta V_j(x_i) \geq \left(1 - \frac{\epsilon}{2}\right) S(x_i)$. However, this way we cannot easily approximate those sets for which multiplicities differ by large factors. For example, consider a set S represented through the vector $\langle 0, \ldots, 0, 1, Q \rangle$, where $Q \gg Y^m$, in particular where $Q \gg \frac{2Y^m}{\epsilon}$. For each value β and each vector V_j, the vector βV_j will either be inaccurate with respect to the multiplicity of element x_{m-1} or with respect to the multiplicity of element x_m (or with respect to both these multiplicities).

The main step of our algorithm is to modify the instance I so that we replace each multiset S from the family $\mathcal{S}$ with a sequence of vectors of the form βV_j that altogether add to at most the multiset S (each such sequence can contain multiple vectors of different "shapes" V_j and of different scaling factors β). The goal is to obtain an instance that on the one hand consists of "nicely-structured" sets (vectors) only, and on the other hand has the following property: If in the initial instance I there exist K sets that cover elements $x_1, \ldots, x_m$ with multiplicities $r_1, \ldots, r_m$, then in the new instance there exist K sets that cover elements $x_1, \ldots, x_m$ with multiplicities $r'_1, \ldots, r'_m$, such that $\sum_i \max(0, r_i - r'_i) < \epsilon \sum_i r_i$. We refer to this as the *almost-cover approximation property*.

The procedure for replacing a given set S is presented as Algorithm 1. This algorithm calls the `Emit` function with arguments (β, V) for each vector βV that it wants to output (V is always one of the vectors $V_1, \ldots, V_X$). The emitted sets replace the set S from the input. Below we show that if we apply Algorithm 1 to each set from $\mathcal{S}$, then the resulting instance I' has our almost-cover approximation property.

Let us consider how Algorithm 1 proceeds on a given set S. For the sake of clarity, let us assume there is no rounding performed by Algorithm 1 in function `Round_And_Emit` (the loop in line 29). We will come back to this issue later.

The algorithm considers the elements of the universe—indexed by variable i throughout the algorithm—in the order given by the vector "sorted" (formed in line 3 of Algorithm 1). Let $\prec$ be the order in which Algorithm 1 considers the elements (so $x_{i'} \prec x_{i''}$ means that $x_{i'}$ is considered before $x_{i''}$), and let $x'_1, \ldots, x'_m$ be the elements from the universe renamed so that $x'_1 \prec x'_2 \prec \cdots \prec x'_m$. Let r be the number of sets that Algorithm 1 emits on our input set S and let these sets be $S_1, S_2, \ldots, S_r$. (This is depicted on Figure 2, where for the sake of the example we take $m = 6$ and $r = 3$.)

Consider the situation where the algorithm emits the k'th set, S_k, and let i_k be the value of variable i right before the call to `Round_And_Emit` that caused S_k to be emitted. Note that each element x from the universe such that $x'_{i_k} \prec x$ has the same multiplicity in S_k as element x'_{i_k} (lines 19 and 20 of Algorithm 1). Let $t_k = \sum_j S_k(x'_j)$ be the sum of the multiplicities of the elements from S_k. We make the following observations:

Observation 1: $S_k(x'_{i_k}) = Z \cdot S_k(x'_{i_k - 1})$.

Algorithm 1: The transformation algorithm used in the proof of Theorem 5. The algorithm replaces a given set S with a sequence of vectors of the form βV_j.

```
Main(S):
    multip ← ⟨(1, S(x_1)), (2, S(x_2)), ..., (m, S(x_m))⟩;
    sorted ← sort(multip) ;                    // sort in ascending order of multiplicities
    i ← 0 ;
    //  sorted[i].first refers to the i'th item's number
    //  sorted[i].second refers to its multiplicity
    while sorted[i].second = 0 do
        i ← i + 1 ;
    Main_Rec(i, sorted) ;

Main_Rec(i, multip):
    V ← ⟨0, 0, ..., 0⟩ (vector of m zeros). ;
    β ← multip[i].second ;
    V[multip[i].first] ← 1 ;
    i ← i + 1 ;
    while i ≤ m do
        if multip[i].second < Y · multip[i−1].second then
            V[multip[i].first] ← multip[i].second / β ;
            i ← i + 1 ;
        else
            for j ← i to m do
                V[multip[j].first] ← Z·multip[i−1].second / β ;
            Round_And_Emit(β, V) ;
            for j ← 1 to m do
                multip[j].second ← multip[j].second − βV[multip[j].first] ;
            Main_Rec(i, multip) ;
            return
    Round_And_Emit(β, V);

Round_And_Emit(β, V):
    for ℓ ← 1 to m do
        V[ℓ] ← ⌊2V[ℓ] / ε⌋ / (ε/2);
    Emit(β, V);
```

Observation 2: It holds that $S_{k+1}(x'_{i_k}) \geq Y \cdot S_k(x'_{i_k-1}) - Z \cdot S_k(x'_{i_k-1}) = (Y-Z)S_k(x'_{i_k-1}) = \frac{(Y-Z)}{Z} S_k(x'_{i_k})$.

Observation 3: We have that $S_{k+1}(x'_{i_k}) \geq \frac{(Y-Z)}{Z} S_k(x'_{i_k}) \geq \frac{(Y-Z)}{Zm} t_k$. Further, we have that

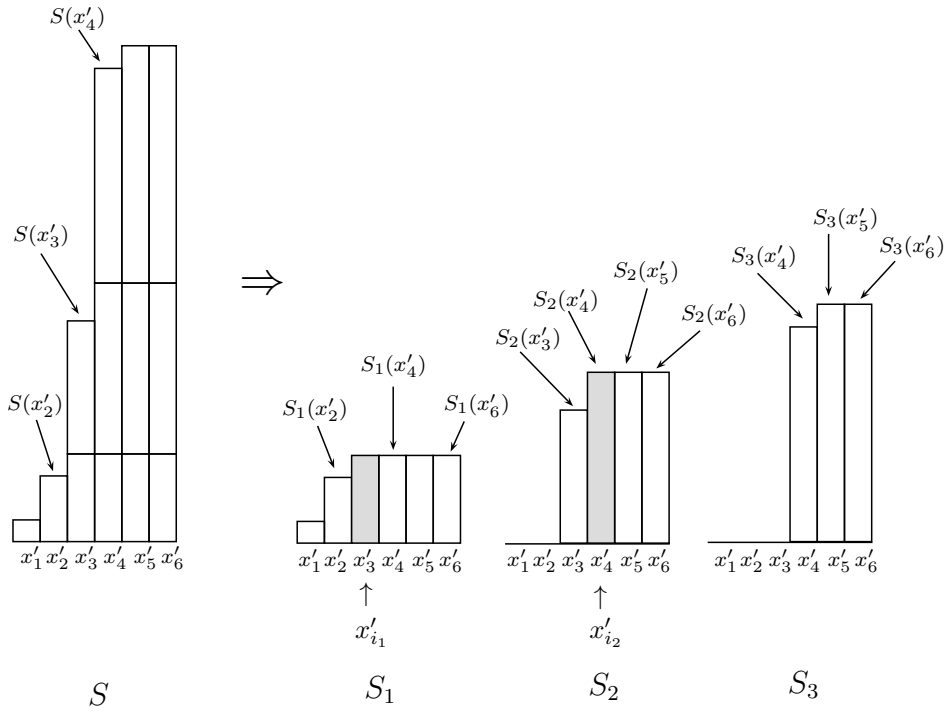

Figure 2: An example for Algorithm 1: The algorithm replaces S with sets S_1, S_2, and S_3.

$S_{k+1}(x'_{i_{k+1}}) \geq S_{k+1}(x'_{i_k}) \geq \frac{(Y-Z)}{Zm^2} \sum_{j \leq k} t_j$. (To see why the last inequality holds, note that $k \leq m$.)

Observation 4: For $i < i_k$ it holds that $\sum_{q \leq k} S_q(x'_i) = S(x'_i)$.

Now let us consider some solution for instance I that consists of K sets, $\mathcal{S}^{\text{opt}} = \{S_1^{\text{opt}}, S_2^{\text{opt}}, \ldots, S_K^{\text{opt}}\} \subseteq \mathcal{S}$. These sets, altogether, cover all elements from the universe with required multiplicities. That is, it holds that for each i we have $\sum_{S \in \mathcal{S}^{\text{opt}}} S(x_i) \geq r_i$. For each set $S \in \mathcal{S}^{\text{opt}}$ and for each element x_i from the universe, we pick an arbitrary number $y_{S,i}$ so that altogether the following conditions hold:

1. For every set $S \in \mathcal{S}^{\text{opt}}$ and every x_i, $y_{S,i} \leq S(x_i)$.
2. For every x_i, $\sum_{S \in \mathcal{S}^{\text{opt}}} y_{S,i} = r_i$.

Intuitively, for a given set S, the values $y_{S,1}, y_{S,2}, \ldots, y_{S,m}$ describe the multiplicities of the elements from S that are *actually used* to cover the elements. Based on these numbers, we will show how to replace each set from $\mathcal{S}^{\text{opt}}$ with one of the sets emitted for it, so that the resulting family of sets has the almost-cover approximation property.

Consider a set $S \in \mathcal{S}^{\text{opt}}$ for which Algorithm 1 emits r sets, $S_1, S_2, \ldots, S_r$. As in the discussion of Algorithm 1, let $x'_1, \ldots, x'_m$ be the elements from the universe in which Algorithm 1 considers

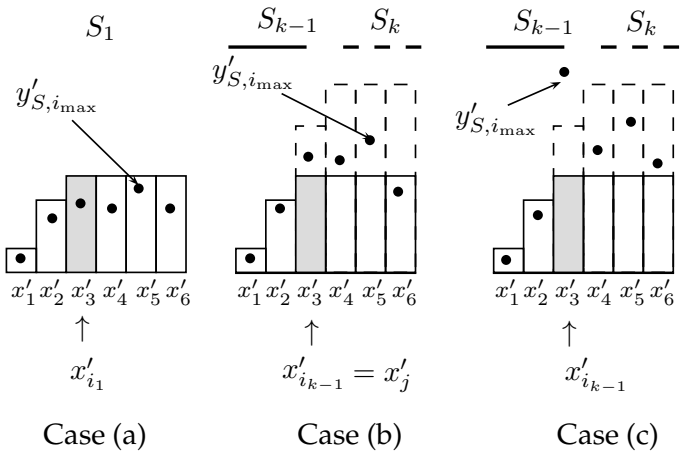

Figure 3: The cases in the proof of Theorem 5. The bullets represent values $y'_{S,1}, \ldots, y'_{S,m}$.

them (when emitting sets for S). We write $y'_{S,i}$ to mean the value $y_{S,j}$ such that $x_j = x'_i$. Let $\mathcal{R} = \{S_1, S_2, \ldots, S_r\}$, let $i_{\max} = \operatorname{argmax}_i y'_{S,i}$, and let S_{repl} be the set from $\mathcal{R}$ defined in the following way:

1. If for every set $S_k \in \mathcal{R}$ we have $S_k(x'_{i_{\max}}) < y'_{S,i_{\max}}$, then S_{repl} is the set $S_k \in \mathcal{R}$ with the greatest value $S_k(x'_{i_{\max}})$ (the set that covers element $x'_{i_{\max}}$ with the greatest multiplicity). This is the case denoted as "Case (c)" in Figure 3.

2. Otherwise S_{repl} is the set $S_k \in \mathcal{R}$ that has the lowest value $S_k(x'_{i_{\max}})$, yet no-lower than $y'_{S,i_{\max}}$. This is the case denoted as either "Case (a)" or "Case (b)" in Figure 3.

We now show that S_{repl} is a good candidate for replacing S, that is, that

$$\sum_i \max(0, y'_{S,i} - S_{\text{repl}}(x'_i)) < \epsilon \sum_i y'_{S,i}.$$

To this end, we consider the three cases depicted in Figure 3:

Case (a) It holds that $y'_{S,i_{\max}} < S_1(x'_{i_{\max}})$ (that is, S_1 already covers the most demanding element of the universe to the same extent as S does). This means that we have $\sum_\ell \max(0, y'_{S,\ell} - S_1(x'_\ell)) = 0$. By the criterion for choosing set S_{repl}, we have that $S_{\text{repl}} = S_1$.

Case (b) There exist sets $S_{k-1}, S_k \in \mathcal{R}$ such that $S_k(x'_{i_{\max}}) \geq y'_{S,i_{\max}} > S_{k-1}(x'_{i_{\max}})$ (and thus, $S_{\text{repl}} = S_k$). Let $x'_j = x'_{i_{k-1}}$ (recall from the discussion of Algorithm 1 that i_{k-1} is the index of the universe element which caused emitting S_{k-1}). Let us consider two subcases:

(i) $y'_{S,i_{\max}} \leq S_k(x'_j)$: We first note that for each $i \geq j$ it holds that $y'_{S,i} \leq S_k(x'_i)$. Further, for each $i < j$, we have $y'_{S,i} \leq \sum_{\ell \leq k-1} S_\ell(x'_i)$ (this follows from Observation 4 and the

fact that $y'_{S,i} \leq S(x'_i)$). Based on this inequality, we get:

$$\begin{aligned}\sum_{i<j} y'_{S,i} &\leq \sum_{i<j} \sum_{\ell \leq k-1} S_\ell(x'_i) \leq \sum_{\ell \leq k-2} t_\ell + \sum_{i<j} S_{k-1}(x'_i) \\ &\leq \frac{Zm^2}{(Y-Z)} S_{k-1}(x'_j) + \frac{m}{Z} S_{k-1}(x'_j) \qquad \text{(Observations 3 and 1)} \\ &\leq \frac{\epsilon}{2} S_{k-1}(x'_j) \leq \frac{\epsilon}{2} y_{S,i_{\max}}.\end{aligned}$$

In consequence, it holds that $\sum_\ell \max(0, y'_{S,\ell} - S_k(x'_\ell)) < \frac{\epsilon}{2} \sum_\ell y'_{S,\ell}$.

(ii) $y'_{S,i_{\max}} > S_k(x'_j)$: For $\ell \geq i_k$, we have $S_k(x'_\ell) \geq y'_{S,\ell}$ (this follows from the fact that $S_k(x'_{i_{\max}}) \geq y'_{S,i_{\max}}$ and the definition of i_k). For $\ell < i_k$, by Observation 4 and the fact that $y'_{S,\ell} \leq S(x'_\ell)$, we have $y'_{S,\ell} - S_k(x'_\ell) \leq \sum_{q \leq k-1} S_q(x'_\ell)$. Thus we get:

$$\begin{aligned}\sum_\ell \max(0, y'_{S,\ell} - S_k(x'_\ell)) &\leq \sum_{\ell < i_k} \max(0, y'_{S,\ell} - S_k(x'_\ell)) \\ &\leq \sum_{\ell < i_k} \sum_{q \leq k-1} S_q(x'_\ell) \leq \sum_{q \leq k-1} t_q \\ &\leq \frac{Zm^2}{(Y-Z)} S_k(x'_j) \qquad \text{(Observation 3)} \\ &\leq \frac{Zm^2}{(Y-Z)} y'_{S,i_{\max}} \leq \frac{\epsilon}{2} \sum_\ell y'_{S,\ell}.\end{aligned}$$

Case (c) Every set $S_k \in \mathcal{R}$ has $S_k(x'_{i_{\max}}) < y'_{S,i_{\max}}$.

By the choice of S_k (the set from $\mathcal{R}$ that has highest multiplicity of $x'_{i_{\max}}$), we infer that $\sum_{q \leq k} S_q(x'_{i_{\max}}) = S(x'_{i_{\max}})$. Also, for every $\ell < i_{\max}$, we have $\sum_{q \leq k} S_q(x'_\ell) = S(x'_\ell)$. Consequently, for every $\ell \leq i_{\max}$ we have

$$y_{S,\ell} - S_k(x'_\ell) \leq \sum_{q \leq k-1} S_q(x'_\ell) \leq \sum_{q \leq k-1} S_q(x'_{i_{\max}}).$$

Further, for every $\ell > i_{\max}$, we have

$$y'_{S,\ell} - S_k(x'_\ell) \leq y'_{S,i_{\max}} - S_k(x'_{i_{\max}}) \leq S(x'_{i_{\max}}) - S_k(x'_{i_{\max}}) \leq \sum_{q \leq k-1} S_q(x'_{i_{\max}}).$$

Based on these observations, we get the following:

$$\begin{aligned}\sum_\ell \max(0, y_{S,\ell} - S_k(x'_\ell)) &\leq m \sum_{q \leq k-1} S_q(x'_{i_{\max}}) \leq m \sum_{q \leq k-1} t_q \\ &\leq m \frac{Zm^2}{(Y-Z)} S_k(x'_{i_k}) \qquad \text{(Observation 3)}\end{aligned}$$

$$\leq \frac{Zm^3}{(Y-Z)} y'_{S,i_{\max}} \leq \frac{\epsilon}{2} y'_{S,i_{\max}} \leq \frac{\epsilon}{2} \sum_{\ell} y'_{S,\ell}.$$

Thus we obtain the desired bound.

The above case analysis almost shows that we indeed have the almost-cover approximation property. It remains to consider the issue of rounding (line 29 of Algorithm 1). This rounding introduces inaccuracy that is bounded by factor $\frac{\epsilon}{2}$ and thus, indeed, we do have the almost-cover approximation property.

Now, given the new instance I', it suffices to find a solution for I' that satisfies the desired approximation guarantee (that is, a collection $\mathcal{S}'$ of at most K sets that form an ϵ-almost-cover). It is possible to do so using our technique from Section 2.

Let us recall that the new instance consists of the sets that are of the form βV_j (recall the discussion at the beginning of the proof). For each vector V_j, $1 \leq j \leq X$, we introduce an integer variable v_j, which, intuitively, gives the number of sets with shape V_j taken into the solution. Further, for each v_j, $1 \leq j \leq X$, and each x_i, $1 \leq i \leq m$, we introduce a concave function $f_{i,j}$, so that $f_{i,j}(v_j)$ is the maximum multiplicity with which element x_i is covered by some v_j "best" sets of the shape V_j (these are the v_j sets which have been emitted with the highest values of β). Finally, we introduce variables $\mathrm{miss}_1, \ldots, \mathrm{miss}_m$, responsible for measuring the inaccuracy levels (in other words, miss_i gives the missing multiplicity for element x_i). The constraints for our mixed integer linear program are given below:

1. $\sum_{j=1}^{X} v_j \leq B$.
2. For each j, $1 \leq j \leq X$: $v_j \geq 0$.
3. $\sum_{i=1}^{m} \mathrm{miss}_i \leq \epsilon \sum_i r_i$.
4. For each i, $1 \leq i \leq m$: $\sum_{j=1}^{X} f_{i,j}(v_j) \geq r_i - \mathrm{miss}_i$.

One can verify that solutions to this program directly correspond to ϵ-almost-covers for instance I. This completes the proof. □

3.2 From Covering Problems to Approval Voting

In this section we show a relation between several problems regarding Approval voting and the covering problems studied above. In consequence, we will explain how our technique can be used for obtaining fixed-parameter tractability results for these voting problems.

We model an election as a pair $E = (C, V)$, where $C = \{c_1, \ldots, c_m\}$ is a set of candidates and $V = (v_1, \ldots, v_n)$ is a collection of voters. Each voter is represented through his or her preferences. For the case of Approval voting, each voter's preferences take the form of a set of candidates approved by this voter. The candidate(s) receiving the most approvals are the winner(s). In other words, we assume the nonunique-winner model (if several candidates have the same number of approvals, then we view each of them as winning). We write $\mathrm{score}_E(c_i)$ to denote the number of voters approving c_i in election E. We refer to elections that use Approval voting and represent

voter preferences in this way as approval elections. In a weighted election, voters also have integer weights in addition to their preferences. A voter v with weight $\omega(v)$ counts as $\omega(v)$ copies of an unweighted voter.[3]

We are interested in the following three problems.

Definition 3 (Bartholdi et al. [1], Faliszewski et al. [18, 43])**.** *In each of the problems Approval-*$\$$BRIBERY *(priced bribery), Approval-*$\$$CCAV *(priced control by adding voters), and Approval-*$\$$CCDV *(priced control by deleting voters), we are given an approval election $E = (C, V)$ with $C = \{p, c_1, \ldots, c_m\}$ and $V = (v_1, \ldots, v_n)$, and an integer budget B. In each of the problems the goal is to decide whether it is possible to ensure that p is a winner, at a cost of at most B. The problems differ in the allowed actions and possibly in some additional parts of the input:*

1. *In Approval-*$\$$BRIBERY*, for each voter v_i, $1 \leq i \leq n$, we are given a nonnegative integer price π_i; for this price we can change v_i's approval set in any way we choose.*
2. *In Approval-*$\$$CCAV *(CCAV stands for "Constructive Control by Adding Voters") we are given a collection $Q = (q_1, \ldots, q_{n'})$ of additional voters. For each additional voter q_i, $1 \leq i \leq n'$, we also have a nonnegative integer price π_i for adding q_i to the original election.*
3. *In Approval-*$\$$CCDV *(CCDV stands for "Constructive Control by Deleting Voters"), we have a nonnegative integer price π_i for removing each voter v_i from the election.*

*In the weighted variants of these problems (which we denote by putting "*WEIGHTED*" after "Approval"), the input elections (and all the voters) are weighted; in particular, each voter v has an integer weight $\omega(v)$. The unpriced variants of these problems (denoted by omitting the dollar sign from their names) are defined identically, except that all prices have the same unit value.*

The above problems are, in essence, equivalent to certain covering problems. Briefly put, the relation between WMM and various election problems (as those defined above) is that the universe corresponds to the candidates in the election, the multisets correspond to the voters, and the covering requirements depend on particular actions that we are allowed to perform.

Construction 1. *Consider an instance of Approval-*$\$$CCDV *with election $E = (C, V)$, where $C = \{p, c_1, \ldots, c_m\}$ and $V = (v_1, \ldots, v_n)$, with prices $\pi_1, \ldots, \pi_n$ that one needs to pay to the respective voters in order to convince them not to participate in the election, and with budget B. We can express this instance as an instance of* WEIGHTED MULTISET MULTICOVER *as follows. For each voter v_i not approving p, we form a multiset S_i with weight π_i that includes exactly the candidates approved by v_i, each with multiplicity exactly one. For each candidate c_i, $1 \leq i \leq m$, we set its covering requirement to be $\max(\text{score}_E(c_i) - \text{score}_E(p), 0)$. There is a way to ensure p's victory by deleting voters of total cost at most B if and only if it is possible to solve the presented instance of* WEIGHTED MULTISET MULTICOVER *with budget B.*

[3] There is a name clash between the literature on covering problems and that on elections. In the former, "weights" refer to what the voting literature would call "prices." Weights of the voters are modeled as multiplicities of the elements in the multisets. We kept the naming conventions from the respective parts of the literature to make our results more accessible to researchers from both communities.

Naturally, we do not use the full generality of WMM in Construction 1; in fact, we provide a reduction to WEIGHTED SET MULTICOVER, where the multiplicities of input multisets are either 0 or 1. This is important since Proposition 1 says that WMM is NP-hard even for a single element in the universe. From the viewpoint of voting theory, it is also interesting to consider UNIFORM MULTISET MULTICOVER, where for each multiset S_i in the input instance there is a number t_i such all elements belonging to S_i have multiplicity either equal to zero or to t_i. Using an argument similar to that used in Construction 1, it is easy to show that UNIFORM MULTISET MULTICOVER is, in essence, equivalent to Approval-WEIGHTED-CCDV.

In Construction 1 we have considered Approval-$CCDV because, among our problems, it is the most straightforward one to model via a covering problem. Nonetheless, constructions with similar flavor are possible both for Approval-$CCAC and for Approval-$BRIBERY. Formally, we have the following result.

Proposition 3. APPROVAL-$CCAV, APPROVAL-$CCDV, APPROVAL-$BRIBERY, APPROVAL-WEIGHTED-CCAV, *and* APPROVAL-WEIGHTED-CCDV *are fixed-parameter tractable when parameterized by the number of candidates.*

Proof. We describe for each voting problem either a reduction to WEIGHTED SET MULTICOVER or to UNIFORM MULTISET MULTICOVER. Formally, we either use standard many-one reductions or very simple special cases of Turing-reductions: In an outer loop, we iterate through certain values, which give an additional hint on how the solution looks like (we refer to it as "guessing"), and then resolve the remaining problem by a transformation to one of the two covering problems. We finally answer yes if one of the covering instances was a yes-instance. In our reductions, the universe set U is always identical to the candidate set C, but the covering requirements, the family $\mathcal{S}$ of the (multi)sets, the weights, and the prices differ.

Approval-$CCDV. See Construction 1.

Approval-$BRIBERY. Consider an instance of Approval-$BRIBERY with election $E = (C, V)$, where $C = \{p, c_1, \ldots, c_m\}$ and $V = (v_1, \ldots, v_n)$, with prices $\pi_1, \ldots, \pi_n$ for changing the voter's approval set, and with budget B. Observe that Approval-$BRIBERY is very similar to Approval-$CCDV, because we can assume without loss of generality that each bribed voter finally approves only candidate p. However, the decisive difference is that we do not know the final number of approvals that p will get because this depends on the given budget B, on the prices of the voters, and on how many bribed voters already approved p (but together with some other candidates). We circumvent this lack of knowledge by guessing the number ℓ of additional approvals p obtains through the bribery process. This also gives us the score $s^* := \mathrm{score}_E(p) + \ell$ of p in the final election (containing the ℓ bribed voters). Now, we have to ensure (i) that p really obtains the guessed score and (ii) that all other candidates which originally have a higher score lose enough approvals through the bribery process. We can express this as an instance of WEIGHTED MULTISET MULTICOVER as follows. For each voter $v_i \in V$, we form a multiset S_i with weight π_i that includes all the candidates approved by v_i, each with multiplicity exactly one, as well as candidate p also with multiplicity one if and only if v_i *does not* approve p. For each candidate $c \in C \setminus \{p\}$, we set its covering requirement

to be $\max(\text{score}_E(c) - s^*, 0)$. For p we set the covering requirement to ℓ. It is easy to see that there is a way to ensure p's victory by adding voters of total cost at most B if and only if it is possible to solve the presented instance of WEIGHTED MULTISET MULTICOVER with budget B. Since the constructed instance is, in fact, an instance of WEIGHTED SET MULTICOVER, we obtain an FPT algorithm.

Approval-$CCAV. Consider an instance of Approval-$CCAV with election $E = (C, V)$, where $C = \{p, c_1, \ldots, c_m\}$, $V = (v_1, \ldots, v_n)$, and $Q = (q_1, \ldots, q_{n'})$, with prices $\pi_1, \ldots, \pi_{n'}$ that one needs to pay to the respective voters from Q in order to convince them to participate in the election, and with budget B. It is never useful to add a voter that does not approve candidate p. Adding a voter w (who approves p) to the election has one decisive effect: it decreases the score difference between candidate p and each candidate that is not approved by w. Hence, we can express this instance as an instance of WEIGHTED MULTISET MULTICOVER as follows. For each voter $q_i \in Q$ approving p, we form a multiset S_i with weight π_i that includes exactly the candidates not approved by q_i, each with multiplicity exactly one. For each candidate $c \in C$, we set its covering requirement to be $\max(\text{score}_E(c) - \text{score}_E(p), 0)$. It is easy to see that there is a way to ensure p's victory by adding voters of total cost at most B if and only if it is possible to solve the presented instance of WEIGHTED MULTISET MULTICOVER with budget B. Since the constructed instance is, in fact, an instance of WEIGHTED SET MULTICOVER, we obtain an FPT algorithm.

APPROVAL-WEIGHTED-CCDV and APPROVAL-WEIGHTED-CCAV. By analogous arguments as above we do the same construction as for Approval-$CCDV (resp. Approval-$CCAV) except that (i) we omit the weights of the multisets, and (ii) we set the multiplicity for each element in the multiset to the weight of the corresponding voter. In consequence, we obtain instances of UNIFORM MULTISET MULTICOVER, which can be solved in FPT time. □

On the other hand, it is either shown explicitly by Faliszewski et al. [18] or follows trivially that when the problems from the above proposition have both prices and weights, then they are NP-hard already for two candidates (that is, they are Para-NP-hard with respect to the number of candidates).

4 Further Generalizations of the Results Related to Voting

We now consider the ordinal model of elections, where each voter's preferences are represented as an order, ranking the candidates from the most preferred one to the least preferred one. For example, for $C = \{c_1, c_2, c_3\}$, vote $c_1 \succ c_3 \succ c_2$ means that the voter likes c_1 best, then c_3, and then c_2.

There are many different voting rules for the ordinal election model. Here we concentrate only on scoring rules. A scoring rule for m candidates is a nondecreasing vector $\alpha = (\alpha_1, \ldots, \alpha_m)$ of integers. Each voter gives α_1 points to his or her most preferred candidate, α_2 points to the second most preferred candidate, and so on. Examples of scoring rules include the Plurality rule, defined through vectors of the form $(1, 0, \ldots, 0)$, k-Approval, defined through vectors with k ones followed by $m - k$ zeroes, and Borda count, defined through vectors of the form $(m - 1, m - 2, \ldots, 0)$.

For each voting rule $\mathcal{R}$ in the ordinal model, it is straightforward to define $\mathcal{R}$-\$CCAV, $\mathcal{R}$-\$CCDV, and $\mathcal{R}$-\$BRIBERY. Using our new framework, we obtain the following result.

Theorem 6. *For every voting rule $\mathcal{R}$ for which winner determination can be expressed through a set of integer linear inequalities over variables that indicate how many voters with each given preference order are in the election, $\mathcal{R}$-\$CCAV, $\mathcal{R}$-\$CCDV, and $\mathcal{R}$-\$BRIBERY are fixed-parameter tractable when parameterized by the number of candidates.*

Proof. The proof follows the same structure as that of Theorem 3. We will present the proof only for $\mathcal{R}$-\$CCDV; the other cases follow by applying the same approach. Let us consider an instance of $\mathcal{R}$-\$CCDV with the set of candidates $C = \{p, c_1, \ldots, c_m\}$, the collection of voters $V = (v_1, \ldots, v_n)$, prices $\pi_1, \ldots, \pi_n$, and with budget B.

Let X be the set of integer variables which indicate how many voters with each given preference order are in the election. By x_σ we denote the variable from X which corresponds to the preference ranking σ. Clearly, the size of X is upper-bounded by $m!$, i.e., by a function of the number of the candidates. Let S be the set of inequalities over variables from X that encode that p is a winner in the election.

We construct an integer program with convex transformations as follows. For each preference order σ we introduce one integer variable c_σ. Intuitively, this variable indicates how many voters with the preference order σ we need to remove from the election. Additionally, we introduce a function f_σ such that $f_\sigma(c_\sigma)$ is the total price of c_σ least expensive voters whose preference order is σ. For each $x_\sigma \in X$ we replace x_σ in S with the number of voters from V whose preference ranking is σ minus c_σ, and we add a constraint enforcing that this difference is greater or equal to zero. Finally, we add the budget constraint $\sum_\sigma f_\sigma(c_\sigma) \leq B$. It is apparent that our ILP is feasible if and only if the answer to the original instance is "yes". We solve it in FPT time via Theorem 2. □

For a more detailed description of the class of voting rules where "winner determination can be expressed through integer linear inequalities," we point the reader to the works of Dorn and Schlotter [13] or of Faliszewski et al. [19]. In particular, Theorem 6 applies to all scoring rules. This, and the results from the previous section, resolves an issue dating back to the work of Faliszewski et al. [18, Theorem 4.4, Theorem 4.13; the conference version of their work was published in 2006], who have shown that \$BRIBERY is in XP for approval voting and for scoring protocols (for the parameterization by the number of candidates).[4] Until our work, the exact parameterized complexity of these problems was unknown.

Our framework also allows to partially resolve an open problem posed by Bredereck et al. [8] regarding SHIFT BRIBERY. In this problem we are given an election and a preferred candidate p, and the goal is to ensure p's victory by shifting p forward in some of the votes (the cost of each shift depends on the number of positions by which we shift p). Under the "sortable prices assumption", voters with the same preference orders can be sorted so that if voter v' precedes voter v'', then we know that shifting p by each given number of positions i in the vote of v' costs at most as much

[4] They did not speak of XP-membership explicitly, but this is exactly what they have shown. Bredereck et al. [6] popularized the issue of resolving if \$BRIBERY problems parameterized by the number of candidates are in FPT or are W[·] hard, leading in particular to our solution and to the slightly later work of Koutecký et al. [38].

as doing the same in the vote of v''. Using this assumption, we obtain the following result (all-or-nothing prices are a special case of sortable prices where we always shift p to the top of a given vote or we leave the vote unchanged).

Theorem 7. *For Borda (and for Maximin and Copeland voting rules),* SHIFT BRIBERY *for sortable price functions and for all-or-nothing price functions is fixed-parameter tractable when parameterized by the number of candidates.*

Bredereck et al. [8] gave an FPT approximation scheme for the problems from Theorem 7; we can use part of their algorithm and apply our new framework in order to derive an exact and not only approximate solution. Their algorithm rephrases the problem and then applies a bounded search through the solution space. We can use their rephrasing but replace the search by solving a MIP/SPLIT instance. We omit technical details since recently Koutecký et al. [38] showed fixed-parameter tractability of the SWAP BRIBERY problem parameterized by the number of candidates, as part of a very general result using the n-fold IP technique. SWAP BRIBERY is a generalization of SHIFT BRIBERY, so the result of Koutecký et al. [38] is stronger than the one given above.

5 Discussion & Outlook

We have proposed an extension of Lenstra's famous result for solving ILPs. In our extended formulation, one can replace any integer variable with its simple piecewise linear transformation—this transformation needs to be either convex or concave, depending on the position of the variable in the ILP. We have shown that such extended ILPs can still be solved in FPT time with respect to the number of integer variables, as long as there are at most polynomially many pieces.

We have demonstrated several applications of our general result which relate to classic covering problems and to selected voting problems. Most notably, we have proven that WEIGHTED SET MULTICOVER is fixed-parameter tractable when parameterized by the number of elements to cover. Further, building upon our general result, but using a more technically involved argument, we have proved the existence of an FPT approximation scheme for MULTISET MULTICOVER, also for the parameterization by the number of elements. We have also explained how our general results can be used in studies on control and bribery in elections—we have shown that certain variants of these problems are in FPT when parameterized by the number of candidates. In particular, we have resolved the parameterized complexity of some problems that were open for the last ten years or so.

Our paper leads to several possible directions for future work. First, unfortunately, while Lenstra's algorithm is a very powerful tool for proving FPT membership, it might be too slow in practice. Thus, as pointed out by Bredereck et al. [6], each time an FPT result is achieved through an application of Lenstra's result, it is natural to ask whether one can derive the same result through a direct, combinatorial algorithm. Coming up with such a direct algorithm usually seems very difficult. In practice, one would probably not use Lenstra's algorithm for solving MIPs, but, instead, one of the off-the-shelf optimized heuristics. In the conference version of this paper [7] we provided a preliminary empirical comparison of the running times of the MIP-based algorithm (using an off-the-shelf MIP solver instead of Lenstra's algorithm) and an ILP-based algorithm that reduces our problems directly to integer linear programming (basically without "exploiting" the parameter). Our

results suggest that FPT algorithms based on solving MIPs can be very efficient in practice. A more thorough experimental analysis of these and similar questions would help in understanding the real power and limitations of the techniques based on MIPs, thus we believe it is an important research direction. Second, our work advances a fairly modest literature on FPT approximation schemes. It would be very interesting to further explore the (practical) relevance of such algorithms.

Acknowledgments

We thank Martin Koutecký for his extremely helpful input and feedback concerning related work.

Robert Bredereck was from September 2016 to September 2017 on postdoctoral leave at the University of Oxford, supported by the DFG fellowship BR 5207/2 and at project start in early 2015 supported by DFG project PAWS (NI 369/10). Piotr Skowron was supported by a Humboldt Research Fellowship for Postdoctoral Researchers. Piotr Faliszewski was supported by DFG project PAWS (NI 369/10) during his stay at TU Berlin and by AGH University grant 11.11.230.337 (statutory research) afterward. Nimrod Talmon was supported by DFG, Research Training Group "Methods for Discrete Structures" (GRK 1408), while the author was affiliated with TU Berlin, where most of the work was done. This work was also partly supported by COST Action IC1205 on Computational Social Choice.

References

[1] J. J. Bartholdi, III, C. A. Tovey, and M. A. Trick. How hard is it to control an election. *Mathematical and Computer Modelling*, 16(8–9):27–40, 1992.

[2] D. Baumeister, P. Faliszewski, J. Lang, and J. Rothe. Campaigns for lazy voters: Truncated ballots. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*, pages 577–584, June 2012.

[3] P. Berman, B. DasGupta, and E. Sontag. Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene networks. *Discrete Applied Mathematics*, 155(6–7):739–749, 2007.

[4] E. Bonnet, V. Paschos, and F. Sikora. Parameterized exact and approximation algorithms for maximum k-set cover and related satisfiability problems. *RAIRO-Theoretical Informatics and Applications*, 50(3):227–240, 2016.

[5] F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

[6] R. Bredereck, J. Chen, P. Faliszewski, J. Guo, R. Niedermeier, and G. J. Woeginger. Parameterized algorithmics for computational social choice: Nine research challenges. *Tsinghua Science and Technology*, 19(4):358–373, 2014.

[7] R. Bredereck, P. Faliszewski, R. Niedermeier, P. Skowron, and N. Talmon. Elections with few candidates: Prices, weights, and covering problems. In *Proceedings of the 4th International*

Conference on Algorithmic Decision Theory (ADT '15), volume 9346 of *LNCS*, pages 414–431. Springer, 2015.

[8] R. Bredereck, J. Chen, P. Faliszewski, A. Nichterlein, and R. Niedermeier. Prices matter for the parameterized complexity of shift bribery. *Information and Computation*, 251:140–164, 2016.

[9] I. Caragiannis, D. Kurokawa, H. Moulin, A. D. Procaccia, N. Shah, and J. Wang. The unreasonable fairness of maximum Nash welfare. In *Proceedings of the 2016 ACM Conference on Economics and Computation (EC '16)*, pages 305–322, 2016.

[10] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54(3):1–33, 2007.

[11] D. Dadush, C. Peikert, and S. Vempala. Enumerative lattice algorithms in any norm via M-ellipsoid coverings. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '11)*, pages 580–589, 2011.

[12] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceeedings of the 46th Symposium on Theory of Computing (STOC '14)*, pages 624–633. ACM Press, 2014.

[13] B. Dorn and I. Schlotter. Multivariate complexity analysis of swap bribery. *Algorithmica*, 64 (1):126–151, 2012.

[14] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[15] P. Dvorák, E. Eiben, R. Ganian, D. Knop, and S. Ordyniak. Solving integer linear programs with a small number of global variables and constraints. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI '17)*, pages 607–613. AAAI Press, 2017.

[16] E. Elkind, P. Faliszewski, and A. Slinko. Swap bribery. In *Proceedings of the 2nd International Symposium on Algorithmic Game Theory (SAGT '15)*, volume 5814 of *LNCS*, pages 299–310. Springer, Oct. 2009.

[17] P. Faliszewski and J. Rothe. Control and bribery in voting. In F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors, *Handbook of Computational Social Choice*, chapter 7. Cambridge University Press, 2016.

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

[19] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Multimode control attacks on elections. *Journal of Artificial Intelligence Research*, 40:305–351, 2011.

[20] P. Faliszewski, E. Hemaspaandra, and L. A. Hemaspaandra. Weighted electoral control. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '13)*, pages 367–374, 2013.

[21] P. Faliszewski, P. Skowron, A. Slinko, and N. Talmon. Multiwinner analogues of the plurality rule: Axiomatic and algorithmic views. In *Proceedings of the 30th Conference on Artificial Intelligence (AAAI '16)*, pages 482–488, 2016.

[22] P. Faliszewski, P. Skowron, and N. Talmon. Bribery as a measure of candidate success: Complexity results for approval-based multiwinner rules. pages 6–14, May 2017.

[23] U. Feige. A threshold of ln *n* for approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998.

[24] A. Frank and É. Tardos. An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65, 1987.

[25] T. Gavenčiak, D. Knop, and M. Koutecký. Applying convex integer programming: Sum multicoloring and bounded neighborhood diversity. *CoRR*, abs/1711.02032, 2017.

[26] E. Hemaspaandra, L. A. Hemaspaandra, and J. Rothe. Anyone but him: The complexity of precluding an alternative. *Artificial Intelligence*, 171(5–6):255–285, 2007.

[27] R. Hemmecke, S. Onn, and L. Romanchuk. n-fold integer programming in cubic time. *Mathematical Programming*, 137(1–2):325–341, 2013.

[28] R. Hildebrand and M. Köppe. A new lenstra-type algorithm for quasiconvex polynomial integer minimization with complexity $2^{o(n\log n)}$. *Discrete Optimization*, 10(1):69–84, 2013.

[29] Q. Hua, D. Yu, F. C. M. Lau, and Y. Wang. Exact algorithms for set multicover and multiset multicover problems. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC '09)*, pages 34–44, 2009.

[30] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974.

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

[32] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[33] L. Khachiyan and L. Porkolab. Integer optimization on convex semialgebraic sets. *Discrete & Computational Geometry*, 23(2):207–224, 2000.

[34] D. Knop and M. Koutecký. Scheduling meets n-fold integer programming. Technical Report arXiv:1603.02611, arxiv.org, 2016.

[35] D. Knop, M. Koutecký, and M. Mnich. Combinatorial n-fold Integer Programming and Applications. In *Proceedings of the 25th Annual European Symposium on Algorithms (ESA '17)*, volume 87 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 54:1–54:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[36] S. G. Kolliopoulos. Approximating covering integer programs with multiplicity constraints. *Discrete Applied Mathematics*, 129(2):461–473, 2003.

[37] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, 71(4):495–505, 2005.

[38] M. Koutecký, D. Knop, and M. Mnich. Voting and bribing in single-exponential time. In *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science (STACS '17)*, pages 46:1–46:14. IBFI Dagstuhl, Germany, 2017.

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

[40] J. D. Loera, R. Hemmecke, S. Onn, and R. Weismantel. n-fold integer programming. *Discrete Optimization*, 5(2):231–241, 2008.

[41] J. D. Loera, R. Hemmecke, and M. Köppe. *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. Society for Industrial and Applied Mathematics, 2012.

[42] N. Mattei, M. S. Pini, F. Rossi, and K. B. Venable. Bribery in voting over combinatorial domains is easy. In *Proc. International Symposium on Artificial Intelligence and Mathematics (ISAIM'12)*, 2012.

[43] T. Miąsko and P. Faliszewski. The complexity of priced control in elections. *Annals of Mathematics and Artificial Intelligence*, 77(3-4):225–250, 2016.

[44] D. Peters. Single-peakedness and total unimodularity: Efficiently solve voting problems without even trying. Technical Report 1609.03537, arXiv.org, 2017.

[45] S. Rajagopalan and V. Vazirani. Primal-dual RNC approximation algorithm for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):526–541, 1999.

[46] J. Rothe, editor. *Economics and Computation, An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division*. Springer, 2016.

[47] I. Schlotter, P. Faliszewski, and E. Elkind. Campaign management under approval-driven voting rules. *Algorithmica*, 77(1):84–115, 2017.

[48] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1998.

[49] P. Skowron. FPT approximation schemes for maximizing submodular functions. In *Proceedings of the 12th Conference on Web and Internet Economics (WINE '15)*, pages 324–338, 2016.

[50] P. Skowron and P. Faliszewski. Fully proportional representation with approval ballots: Approximating the MaxCover problem with bounded frequencies in FPT time. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI '15)*, pages 2124–2130, 2015.

[51] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[52] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[53] L. Xia. Computing the margin of victory for various voting rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)*, pages 982–999, June 2012.