Almost Consistent Systems of Linear Equations

KONRAD K. DABROWSKI, Newcastle University, UK
PETER JONSSON, Linköping University, Sweden
SEBASTIAN ORDYNIAK, University of Leeds, UK
GEORGE OSIPOV, Linköping University, Sweden
MAGNUS WAHLSTRÖM, Royal Holloway, University of London, UK

Checking whether a system of linear equations is consistent is a basic computational problem with ubiquitous applications. When dealing with inconsistent systems, one may seek an assignment that minimises the number of unsatisfied equations. This problem is NP-hard and UGC-hard to approximate within any constant even for two-variable equations over the two-element field. We study this problem from the point of view of parameterized complexity, with the parameter being the number of unsatisfied equations. We consider equations defined over a family of commutative domains (i.e. rings without zero divisors) with a particular Helly property. This set contains, for instance, finite and infinite fields, the ring of integers, and univariate polynomial rings with coefficients from a field; more generally, it contains the important class of Prüfer domains. We show that if every equation contains at most two variables, the problem is fixed-parameter tractable. This generalises many eminent graph separation problems such as Bipartization, Multiway Cut and Multicut parameterized by the size of the cutset. To complement this, we show that the problem is W[1]-hard when three or more variables are allowed in an equation, as well as for many commutative rings that are not covered by our fpt result. On the technical side, we introduce the notion of important balanced subgraphs, generalising the important separators of Marx [Theoretical Computer Science, 351:3, 2006] to the setting of biased graphs. Furthermore, we use recent results on parameterized MinCSP [Kim et al., SODA-2021] to efficiently solve a generalisation of Multicut with disjunctive cut requests.

CCS Concepts: • **Theory of computation → Computational complexity and cryptography**.

Additional Key Words and Phrases: parameterized complexity, linear equations, biased graphs, minimum constraint satisfaction (MinCSP), graph separation

ACM Reference Format:
Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, and Magnus Wahlström. 2025. Almost Consistent Systems of Linear Equations. *ACM Trans. Algor.* 1, 1 (February 2025), 55 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Konrad K. Dabrowski, Newcastle University, Newcastle upon Tyne, UK, konrad.dabrowski@newcastle.ac.uk; Peter Jonsson, Linköping University, Linköping, Sweden, peter.jonsson@liu.se; Sebastian Ordyniak, University of Leeds, Leeds, UK, sordyniak@gmail.com; George Osipov, Linköping University, Linköping, Sweden, george.osipov@pm.me; Magnus Wahlström, Royal Holloway, University of London, London, UK, Magnus.Wahlstrom@rhul.ac.uk.

Contents

1 INTRODUCTION

Algorithms for systems of linear equations have been studied since ancient times [25]. As Håstad [27] aptly remarks, for computer science "[t]his problem is in many regards as basic as satisfiability".

******Suggestion

Håstad [27, p. 799] aptly remarks the following concerning equation solving in a computer science context:

> A problem that in many respects is as basic as satisfiability is that of solv- ing a system of linear equations over a field.

******

Well-known methods like Gaussian elimination can recognise and solve consistent systems of equations. However, these methods are not well suited for dealing with inconsistent systems. In one optimisation version of the problem, called MaxLin, one seeks an assignment maximising the number of satisfied equations. In its dual, called MinLin, the objective is to minimise the number of unsatisfied equations. Both MaxLin and MinLin remain NP-hard in severely restricted settings, which has motivated an extensive study of approximation algorithms for these problems. However, the problems resist approximation within reasonable bounds: in particular, MinLin over the two-element field restricted to equations with at most two variables is not approximable within any constant factor under the Unique Games Conjecture (UGC)—in fact, it has been suggested that constant-factor inappoximability of this version of MinLin may be equivalent to UGC (see Definition 3 in [36] and the discussion that follows it). This motivates exploring other approaches to resolving inconsistent systems.

Crowston et al. [13] initiated the study of the parameterized complexity of MinLin with the parameter k being the number of unsatisfied equations. They focused on systems over the two-element field and proved that when every equation has at most two variables, the problem can be solved in $O^*(2^k)$ time[1] by a reduction to Edge Bipartization. Furthermore, they proved that allowing three or more variables in an equation leads to W[1]-hardness. This rules out the existence of an algorithm for this problem running in $O^*(f(k))$ time for any computable function f under the standard assumption FPT $\neq$ W[1]. Göke et al. [24] considered a closely related problem: the parameterized complexity of resolving linear systems of equations and inequalities under a mix of parameters, showing W[1]-hardness and FPT results (see also Bérczi et al. [5]). We emphasise that we exclusively consider equations in this article.

We substantially extend the study of the parameterized complexity of MinLin by considering equations over commutative rings. Thus, we study a particular class H_2 of commutative domains (i.e. rings without zero divisors) that has a natural Helly property defined via ring ideals. The set H_2 includes the finite fields $\mathbb{F}_q$[2], infinite fields such as the rationals $\mathbb{Q}$, the ring of integers $\mathbb{Z}$, the Gaussian integers $\mathbb{Z}[i]$, the ring of univariate polynomials $\mathbb{F}[x]$ over a field $\mathbb{F}$ and many more structures. In particular, we prove that every *Prüfer* domain has the required Helly property. Prüfer domains and related rings are of central importance in commutative algebra (see e.g. [4] and [22]) and have been intensively studied since their introduction [50]. Perhaps unsurprisingly, we show that with three or more variables per equation, the problem is W[1]-hard for every commutative ring (in fact, the hardness proof only uses the coefficients 0, 1 and -1, so the result holds for equations over any non-trivial Abelian group). On the other hand, Min-2-Lin, where each equation has two variables, turns out to be much more interesting: the problem is fixed-parameter tractable for the members of H_2.

[1] O^* hides polynomial factors in the bit-size of the instance.
[2] In fact, finite fields are the only finite domains by Wedderburn's Little Theorem (see Section 5.4).

Problem	Solution	Method	Reduces to
Bipartization	[51] in 2004	Iterative compression	Min-2-Lin($\mathbb{F}_2$)
q-Multiway Cut	[45] in 2006	Important separators (IS)	Min-2-Lin($\mathbb{F}_q$)
Multiway Cut	[45] in 2006	Important separators	Min-2-Lin($\mathbb{Q}$)
Multicut	[6], [47]† in 2011	†Random sampling of IS	Min-2-Lin($\mathbb{Z}$)

Table 1. Graph separation problems related to Min-2-Lin.

1.1 Background

This background section is divided into two parts, in which we describe the MinLin problem in connection with graph and group problems, respectively. We start off with a few basic definitions. Let $\mathbb{D} = (D; +, \cdot)$ denote a commutative ring. An expression $c_1 \cdot x_1 + \cdots + c_r \cdot x_r = c$ is a *(linear) equation over* $\mathbb{D}$ if $c_1, \ldots, c_r, c \in D$ and $x_1, \ldots, x_r$ are variables with domain D. Let S denote a set (or equivalently a system) of equations over $\mathbb{D}$. We let $V(S)$ denote the variables appearing in S and say that S is *consistent* if there is an assignment $\varphi : V(S) \to D$ that satisfies all equations in S. An instance of the computational problem r-Lin($\mathbb{D}$) is a system S of equations in at most r variables over $\mathbb{D}$, and the question is whether S is consistent. To assign positive integer weights to the elements of any set Y, we use a weight function $w : Y \to \mathbb{N}^+$ and write $w(X)$ for any subset $X \subseteq Y$ as a shorthand for $\sum_{e \in X} w(e)$. The following is the main computational problem that we consider in this paper.

Min-r-Lin($\mathbb{D}$)

Instance:	An instance S of r-Lin($\mathbb{D}$), a weight function $w : S \to \mathbb{N}^+$ and an integer k.
Parameter:	k.
Question:	Is there a set $Z \subseteq S$ such that $S - Z$ is consistent and $w(Z) \leq k$?

Background via graphs. Crowston et al. [13] studied the problem Min-r-Lin($\mathbb{F}_2$) and proved that Min-2-Lin($\mathbb{F}_2$) is in FPT. However, their methods do not seem sufficient to solve Min-2-Lin over structures larger than $\mathbb{F}_2$. As a possible explanation and additional motivation, we note that Min-2-Lin over $\mathbb{F}_2$, $\mathbb{F}_q$, $\mathbb{Q}$ and $\mathbb{Z}$ generalize well-known graph separation problems that have served as milestones for the development of parameterized algorithms: these are Bipartization, q-Terminal Multiway Cut, (General) Multiway Cut and Multicut, respectively (see Table 1 for a short summary).

In Bipartization, given a graph G and an integer k, the goal is to remove at most k edges from the graph to make it bipartite. To reduce to Min-2-Lin($\mathbb{F}_2$), create a variable for every vertex and add an equation $x - y = 1$ for every edge $\{x, y\}$ in G. The parameterized complexity status of this problem was resolved by Reed et al. [51] using *iterative compression*, which has since become a common opening of fpt-algorithms, including those presented in this paper (see [14, Chapter 4] for many more examples).

In q-Terminal Multiway Cut, given a graph G, a set of q vertices $t_1, \ldots, t_q$ called *terminals*, and an integer k, the goal is to remove edges of weight at most k from G to separate the terminals into distinct connected components. The problem is in P for $q = 2$ and NP-hard for $q \geq 3$. The reduction to Min-2-Lin($\mathbb{F}_q$) works by introducing an equality $x = y$ for every edge $\{x, y\}$ in G, and assigning a distinct field element α_i to every terminal t_i by adding the equation $t_i = \alpha_i$ with weight $k + 1$ (prohibiting its deletion). Note that the construction above does not work if there are more than q terminals. This limitation does not arise over infinite fields, so Min-2-Lin($\mathbb{Q}$) generalizes Multiway Cut with arbitrarily many terminals. Marx [45] presented the first fpt-algorithm for Multiway Cut, which is based on *important separators*.

This work was followed by a string of improvements [9, 55] including an approach based on linear programming [16, 26] that is especially relevant to our work.

In Multicut, given an graph G, a set of m cut requests $(s_1, t_1), \ldots, (s_m, t_m)$, and an integer k, the goal is to remove edges of weight at most k from G to separate s_i from t_i for all i. This problem clearly generalizes Multiway Cut: a reduction can introduce a request for every pair of terminals. In turn, Min-2-Lin($\mathbb{Z}$) generalizes it as follows: add an equation $x = y$ for every edge $\{x, y\}$ in G; then, for every pair of terminals (s_i, t_i), introduce two new variables s_i' and t_i', and add two equations $s_i = p_i s_i'$ and $t_i = p_i t_i' + 1$ with weight $k + 1$, where p_i is the ith prime number. Clearly, no path connecting s_i and t_i may exist in a consistent subset of equations, since this would imply a contradiction (different remainders modulo p_i). Moreover, if all cut requests are fulfilled, a satisfying assignment can be obtained by applying the Chinese Remainder Theorem in each component. The parameterized complexity status of Multicut was resolved simultaneously by Bousquet et al. [6] and Marx and Razgon [47]. The latter introduced the method of *random sampling of important separators*, also known as *shadow removal*. More recently, the technique of *flow augmentation* has been used to solve the weighted version of the problem [40].

Background via groups. We continue by comparing Min-2-Lin($\mathbb{D}$) to two well-studied problems based on groups: Group Feedback Edge Set (GFES) and Unique Label Cover (ULC).

GFES is a problem that is usually defined in terms of labelled graphs, but can also be defined in terms of equations as follows. The input is a system of group equations of the form $x \cdot \gamma = y$ over a group Γ, where x and y are variables, $\gamma \in \Gamma$ is a constant and $\cdot$ is the composition operation in Γ. The variables take values from Γ, and the question is whether the system can be made consistent by removing at most k equations. More commonly, the input is represented as a *group-labelled graph*, i.e. an oriented graph G where every oriented edge $xy \in E(G)$ is assigned a group label $\gamma = \gamma(xy) \in \Gamma$, and where the reverse direction of the edge has the label $\gamma(yx) = \gamma(xy)^{-1}$. Such a system of equations is consistent if and only if every cycle of the underlying undirected graph of G is consistent. This implies that GFES can equivalently be defined as finding a set of at most k edges which intersects every inconsistent cycle in G. The vertex-deletion variant, Group Feedback Vertex Set (GFVS), is defined accordingly and generalizes GFES. GFES and GFVS parameterized by k generalize many well-studied problems in parameterized complexity such as Feedback Vertex Set, Subset Feedback Vertex/Edge Set, Multiway Cut and Odd Cycle Transversal. Guillemot [26] was the first to study GFVS in terms of parameterized complexity, and showed fpt-algorithms parameterized by $k + |\Gamma|$. Cygan et al. [15] demonstrated GFVS to be fpt parameterized by k alone, even when the group is given only implicitly by an oracle supporting group operations. Iwata et al. [30] showed a faster algorithm, solving GFVS in time $O^*(4^k)$ using an LP-branching approach, also in the oracle model. The latter was generalised by Wahlström [54] to a setting of *balanced* (or *biased) subgraphs*, a connection that we use in our algorithms.

For comparison with Min-2-Lin, consider an Abelian group, e.g. $\mathbb{Z}$ with operation $+$. Group equations $x + \gamma = y$ are equivalent to equations $y - x = \gamma$ in Min-2-Lin($\mathbb{Z}$). Thus Min-2-Lin generalises GFES for all Abelian groups that are the additive group of a ring (which, for example, includes all finite Abelian groups). However, additive equations ignore the multiplicative structure of the ring, which leads to significant algorithmic complications in the Min-2-Lin problem. Indeed, as we have seen above, Min-2-Lin($\mathbb{Z}$) is complex enough to generalise Multicut, which does not seem possible for the simpler problem of GFES. Similarly, if Γ is the multiplicative group of a field $\mathbb{F}$, then group equations $x \cdot \gamma = y$ over Γ have similarities to homogeneous equations $\gamma \cdot x - y = 0$ in Min-2-Lin($\mathbb{Z}$). In fact, for such groups GFES is equivalent to Min-2-Lin with only homogeneous equations, with the additional constraint that no variable can take the zero value.

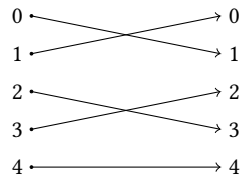


Fig. 1. A permutation of $\{0, 1, 2, 3, 4\}$ not expressible as a linear equation.

Another problem related to Min-2-Lin is Unique Label Cover (ULC) [11, 30, 35]. In ULC over an alphabet Σ, the input is a set of constraints of the form $\pi(x) = y$, where x and y are variables and π is a permutation of Σ—the connection to groups follows from the fact that we can view the permutations as members of the symmetric group on Σ. Constraints are consistent if there is an assignment of values from Σ to the variables that satisfy all constraints. The question is whether the input set can be made consistent by removing at most k constraints. ULC lies at the heart of the Unique Games Conjecture [35]. In the realm of parameterized complexity, it is known that ULC is fixed-parameter tractable when parameterized by $k + |\Sigma|$, but W[1]-hard when parameterized by k alone [11].

To connect this problem with Min-2-Lin, consider for example a field $\mathbb{F}$ and an equation $ax + by = c$ with $a, b, c \in \mathbb{F}$ and $a \neq 0$, $b \neq 0$. For every value of y there is exactly one value of x that satisfies this equation. Thus, any equation is equivalent to a permutation constraint over $\mathbb{F}$, and ULC generalizes Min-2-Lin($\mathbb{F}$). As a consequence, observe that Min-2-Lin($\mathbb{F}_q$) is in FPT for all finite fields $\mathbb{F}_q$, and can be solved in $O^*(q^{2k})$-time using the best known algorithm for ULC [30, 31]. On the other hand, ULC is strictly more general than Min-2-Lin($\mathbb{F}$): Consider $\Sigma = \{0, 1, 2, 3, 4\}$ and a permutation π mapping $(0, 1, 2, 3, 4)$ to $(1, 0, 3, 2, 4)$ (see Figure 1). It is easy to see that no linear equation over $\mathbb{F}_5$ defines this permutation.

1.2 Our Results

We introduce the *Helly dimension* of a ring. In brief, the Helly dimension of $\mathbb{D}$ denoted by $\kappa(\mathbb{D})$ is the minimum number in $\mathbb{N} \cup \{\infty\}$ with the following property: if a family of principal ideal cosets in $\mathbb{D}$ has empty intersection, then it contains a subfamily of $\leq \kappa(\mathbb{D})$ cosets that have empty intersection. We let H_m, $m \in \mathbb{N} \cup \{\infty\}$, contain all commutative domains $\mathbb{D}$ that have Helly dimension at most m and satisfy some mild computational properties. An in-depth treatment of the Helly dimension together with a formal definition of H_m can be found in Section 4.2.

We prove that Min-2-Lin($\mathbb{D}$), where $\mathbb{D} \in \mathsf{H}_2$, is fixed-parameter tractable. For the special case when $\mathbb{D}$ is a field, we provide a faster $O^*(2^{O(k \log k)})$ algorithm. Furthermore, if $\mathbb{D}$ is a finite, q-element field, we provide a $O^*((2q)^k)$ algorithm improving upon the $O^*(q^{2k})$ upper bound obtained by reduction to ULC. To complement the results, we show that Min-3-Lin($\cdot$) is W[1]-hard for every non-trivial ring and thus rule out the existence of fpt-algorithms for Min-r-Lin($\cdot$) when $r \geq 3$ under standard complexity assumptions. Furthermore, we show that Min-2-Lin($\mathbb{D}$) is W[1]-hard for many commutative rings $\mathbb{D}$ that are not members of H_2. For example, the hardness result holds if $\mathbb{D}$ is isomorphic to the direct product of nontrivial rings (such as the ring $\mathbb{Z}/6\mathbb{Z}$ of integers modulo 6).

The Min-2-Lin algorithm consists of three steps: compression, cleaning and cutting. The compression step is based on a standard technique for FPT algorithms. An important technical issue here is that we need a polynomial-time algorithm for 2-Lin($\mathbb{D}$) when $\mathbb{D} \in \mathsf{H}_2$ (Theorem 18). The cleaning step is the most involved, and is based upon our main technical contribution – *important balanced subgraphs*, which is a substantial generalisation of the *important separators* introduced by Marx [45]. In the cutting step, we solve the remaining problem by reducing it to a carefully

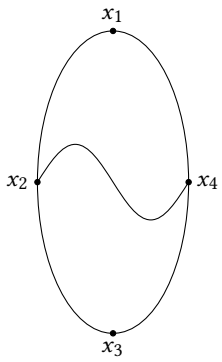


Fig. 2. An example of a theta graph.

chosen cutting problem – Pair Partition Cut. The choice here is delicate: the problem must be solvable in FPT time, but it also has to be sufficiently powerful so that it can handle the cleaned instance. Finally, we solve Pair Partition Cut by a reduction to Boolean MinCSP, for which an FPT dichotomy was recently given by Kim et al. [39].

We now introduce our main novel technical contribution required for the Min-2-Lin algorithm, i.e. important balanced subgraphs.

Important balanced subgraphs. Consider a parameterized deletion problem where the input consists of an edge-weighted graph G, an integer k, and a polynomial-time membership oracle to a family $\mathcal{F}$ of minimal forbidden subgraphs of G that we call *obstructions*. A (sub)graph of G is *balanced* if it does not contain any obstructions. The goal is to find a set of edges of total weight at most k that intersects all obstructions in $\mathcal{F}$. This objective is dual to finding a maximum-weight balanced subgraph of G. For example, in Bipartization a graph is balanced if it is bipartite, and the set of obstructions consists of all odd cycles. Wahlström [54] presented a general method based on LP-branching for solving this problem in fpt time when the obstructions $\mathcal{F}$ are a family of cycles with the *theta property*. This property can roughly be defined as follows: if a chordal path P is added to a cycle C from $\mathcal{F}$, then at least one of the smaller cycles formed by P and C is also in $\mathcal{F}$. For illustration, consider the theta graph in Figure 2: here $C = x_1x_2x_3x_4$ is a cycle and x_2x_4 is a chordal path that cuts it into two smaller cycles $C_1 = x_2x_1x_4$ and $C_2 = x_2x_3x_4$. If C is in the family, then C_1 or C_2 is in the family. For instance, the set of all odd cycles in a graph has the theta property since any chordal path added to an odd cycle forms an odd and an even cycle. Alternatively, the problem can be defined in terms of biased graphs. A *biased graph* is a pair $(G, \mathcal{B})$ where G is a graph and $\mathcal{B}$ is a set of simple cycles in G such that the complement of $\mathcal{B}$ has the theta property; cycles outside $\mathcal{B}$ are referred to as the *unbalanced* cycles in $(G, \mathcal{B})$. Biased graphs are encountered, for instance, in matroid theory [56]. The problem is then to find a set of k edges that intersects every unbalanced cycle in $(G, \mathcal{B})$. In the case of Graph Bipartization, the set $\mathcal{B}$ contains all even cycles.

It is instructive to approach this global problem by instead considering a local version where a single root vertex x is distinguished, and the goal is to remove edges of total weight at most k to make the connected component of x balanced. For a balanced subgraph H of G, define $c(H)$ as the cost of carving H out of G i.e. the sum of weights on all edges between $V(H)$ and $V(G) \setminus V(H)$ plus the weights of edges in the subgraph of G induced by $V(H)$ that are not in H. To solve the global problem, we can choose a root, enumerate solutions to the local problem i.e. balanced subgraphs of cost at most k that include x, and solve the remaining part recursively (possibly with some branching to

guess the intersection of the local and global solutions). The caveat is that the number of balanced subgraphs of cost at most k does not have to be bounded by any function of k. To overcome this obstacle, we need another observation: if there are two balanced subgraphs H_1 and H_2 such that $c(H_2) \geq c(H_1)$ and $V(H_2) \subseteq V(H_1)$, then H_1 is clearly a better choice than H_2 both in terms of cost and in the amount of "work" left in the remaining graph. If the conditions above hold for H_1 and H_2, we say that H_1 *dominates* H_2. See Figure 3 for an illustration. Formally, we want to compute a set $\mathcal{H}$ of *important balanced subgraphs* defined analogously to the important separators: for any balanced subgraph H' including x of cost at most k, there is a subgraph $H \in \mathcal{H}$ such that $V(H') \subseteq V(H)$ and $c(H') \geq c(H)$. In other words, the balanced subgraphs in $\mathcal{H}$ are Pareto efficient balanced subgraphs in terms of maximising the set of covered vertices and minimising the weight. Note here a subtlety in the definition: the total number of incomparable solutions is not bounded in k. For example, if the input consists of a single, unbalanced cycle C_n on n vertices, i.e. $G = C_n$ and $\mathcal{B} = \emptyset$, then we may output a single important balanced subgraph H, with $V(H) = V(C_n)$ and $c(H) = 1$, but there are n incomparable solutions with these parameters, produced by deleting any one edge of the cycle. However, all these solutions belong to the same "equivalence class" of solutions, and the key to applications (in previous work [54] as well as in the present work) is that a global solution does not need to distinguish between them.

We prove (in Theorem 5) that there is a *dominating family* $\mathcal{H}$ of important balanced subgraphs such that $|\mathcal{H}| \leq 4^k$ and every connected balanced subgraph of G with cost at most k is dominated by some member of $\mathcal{H}$. Moreover, there is an fpt-algorithm that computes $\mathcal{H}$ by branching based on the optimum of the half-integral LP-relaxation of the local problem.

We show that important balanced subgraphs strictly generalize important separators: given a graph and two subsets of vertices, one can recover important separators by computing a dominating family of important balanced subgraphs—see Example 6 for further details. In fact, the bounds achieved are identical: using the important balanced subgraph framework to enumerate important separators yields at most 4^k important separators of cost at most k, and they can be enumerated in $O^*(4^k)$ time, both of which match the bounds for important separators [9, 45] (although the polynomial factor incurred in the running time by the method above is significantly higher). Moreover, our algorithms crucially build on the increased generality of the important balanced subgraphs setting, where the cost of carving out a subgraph includes both the cost of a separator and a transversal of unbalanced cycles reachable from the root after separation. This way, important balanced subgraphs can be used for graph separation problems in a more general sense than simply enumerating graph cuts, e.g. for computing transversals of obstruction families with the theta property. As we show in what follows, removing a family of obstructions is a key step in our Min-2-Lin algorithms.

We note that other generalizations of important separators have appeared in the literature. A recent example is reported by Jansen, de Kroon and Włodarczyk [32]. Given an undirected graph G, two vertex sets S and T, and a *finite* set of forbidden subgraphs $\mathcal{F}$, they show that there are $2^{O(k)}$ maximal vertex sets C in G such that C contains S and avoids T, the neighbourhood of C contains at most k vertices, and the subgraph induced by C does not contain any subgraph in $\mathcal{F}$. Furthermore, they provide an fpt algorithm for listing such subgraphs. The obstruction set $\mathcal{F}$ being finite is crucial for their approach, and the $O(k)$ factor in the exponent hides a dependency on $|\mathcal{F}|$. We, on the other hand, deal with infinite sets of obstructions so our results and theirs are incomparable: our obstructions are cycles with theta property, while theirs can be arbitrary connected subgraphs. Another example was presented by Lokshtanov and Ramanujan [43] who introduced a generalisation of important separators to develop an fpt-algorithm for the Parity Multiway Cut problem, where given an undirected graph G, two sets T_e and T_o of even and odd terminals, respectively, and an integer k, the task is to decide whether there is a set S of at most k vertices that hits every even (odd) path with one endpoint v in T_e (T_o) and the other endpoint in $(T_e \cup T_o) \setminus \{v\}$. Given vertex sets X and Y as well as a minimal

X-Y-separator S, their notion of an important separator is defined as follows: S' is an important separator if it is an important separator with respect to X in the original sense and the size of a minimum vertex set that hits all even length paths with endpoints in X in $G - S$ does not increase in $G - S'$. Their notion of important separators is, however, clearly not related to ours.

Let us now illustrate important balanced subgraphs using a few examples.

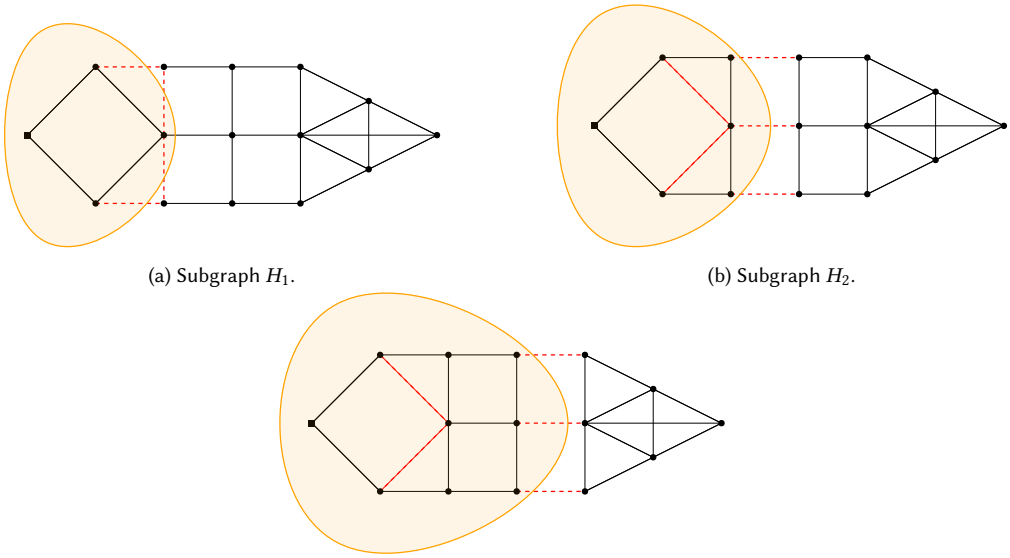
(1). Consider the biased graph $(G, \mathcal{B})$ defined above, where $\mathcal{B}$ contains the set of even cycles. Then a balanced subgraph of $(G, \mathcal{B})$ is precisely a bipartite graph, and our result outputs connected bipartite subgraphs containing the root vertex x.

(2). The previous example can be generalised as follows. Recall that a group-labelled graph (that we discussed in connection with the GFES problems) is a graph G in which every oriented edge of G is assigned a label by γ from a group Γ so that for any edge $\{u, v\} \in E(G)$, the labelling satisfies $\gamma(uv) = \gamma(vu)^{-1}$. Let a cycle $C = (v_1, \ldots, v_n)$ be *balanced* if the product $\gamma(v_1v_2)\gamma(v_2v_3)\ldots\gamma(v_nv_1)$ of the group labels of the edges of C is the identity element of Γ. This forms a biased graph $(G, \mathcal{B})$ where $\mathcal{B}$ is the class of balanced cycles in G. For example, consider a directed graph where edges are labelled by integers (using $(\mathbb{Z}, +)$ as the group). Then the important balanced subgraph theorem can be used to output weakly connected subgraphs which can be laid out on a line such that all edges get the length and orientation according to the value of their labels.

(3). As a special case of the previous example, consider the Subset Feedback Edge Set problem. In this problem, the input is a graph G with a set of special edges $F \subseteq E(G)$, and the goal is to find a set of edges $X \subseteq E(G)$ such that no edge of F is contained in a cycle in $G - X$. It is easy to observe that the class of cycles intersecting F has the theta property (and in fact, can be captured as the unbalanced cycles in a group-labelled graph). Then a subgraph H of G is balanced if any edge of $F \cap E(H)$ is a bridge in H.

For more examples, see Wahlström [54] and Zaslavsky [56]. From a technical perspective, the result follows from a refined analysis of the LP formulation of Wahlström [54] for the problem of computing a minimum (vertex) transversal for the set of unbalanced cycles. Wahlström showed that this can be solved in $O^*(4^k)$ time, given oracle access to $\mathcal{B}$, where k is the solution size. The result works in two parts. First, consider the rooted case outlined above, where the input additionally distinguishes a root vertex $x \in V(G)$ and the task is to find a balanced subgraph of minimum cost, rooted in x. Wahlström provided a half-integral LP-relaxation for this problem, and showed that it can be used to guide a branching process for an fpt-algorithm computing a min-cost rooted balanced subgraph. Second, the LP-relaxation is shown to have some powerful *persistence properties* (see Section 2), that allow the solution from the rooted case to be reused in solving the general problem. We reformulate and simplify these results for the edge deletion case. We find that the extremal ("furthest", or *important*) optima to the LP are described by a rooted, connected, balanced subgraph H of G, where edges of value 1 in the LP are deleted edges within $V(H)$, and the half-integral edges are the edges leaving H, i.e. with precisely one endpoint in $V(H)$. Furthermore, every balanced subgraph H' of G with $x \in V(H')$ can be "improved" to a subgraph H'' so that $V(H) \cup V(H') \subseteq V(H'')$, $c(H'') \leq c(H')$, and the edges of value 1 in the LP are not contained in $E(H')$. A dominating family of important balanced subgraphs of $(G, \mathcal{B})$ rooted in x can then be obtained by branching over the status of the half-integral edges leaving $V(H)$, in an analysis similar to that of Chen et al. [9] for the bound 4^k on the number of important separators.

Min-2-Lin Algorithm for Fields. We begin by explaining our algorithm when restricted to fields for pedagogical reasons—the basic underlying ideas become clearer in this restricted setting. In short, our fpt-algorithms are based



(a) Subgraph H_1.

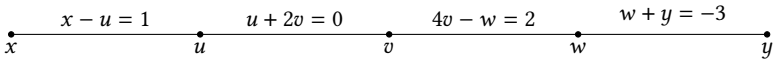
(b) Subgraph H_2.

(c) Subgraph H_3.

Fig. 3. Examples of rooted balanced subgraphs of the same graph. The root is the leftmost vertex, balanced cycles are even cycles, and all edges have unit weight. Red dashed edges are deleted, and the yellow area covers all vertices reachable from the root. The cost of H_1 is 4, while the cost of H_2 and H_3 is 5. Subgraph H_1 is incomparable with H_2 and H_3 since it has lower cost but $V(H_1)$ is a strict subset of $V(H_2)$ and $V(H_3)$. On the other hand, H_2 is dominated by H_3 since $V(H_2) \subsetneq V(H_3)$ while they have the same cost.

on three steps: compression, cleaning, and cutting. Given an instance (S, w_S, k) of Min-2-Lin, we first use iterative compression to compute a slightly suboptimal "solution" X. In the cleaning step we consider the primal graph of S i.e. the graph with vertices for variables of S and edges for equations, and produce a dominating family of important balanced subgraphs around a subset of vertices in $V(X)$. We use the subgraphs for removing rigid cycles that may cause inconsistencies. Finally, the problem reduces to computing a cut in the "cleaned" graph that fulfils certain requirements.

For a basic example, consider Min-2-Lin($\mathbb{Q}$) i.e. Min-2-Lin over the field of rationals. Every instance of this problem can be viewed as a graph where an edge connecting two variables is labelled by an equation from S ranging over these two variables. To gain insight into the problem, first consider a path of equations, e.g.



Equations along the path can be combined to cancel out intermediate variables. For example, combining $x - u = 1$ with $u + 2v = 0$ yields $x + 2v = 1$. In the end, we obtain an equation over the endpoints x and y, namely $2x - y = 3$. Now, if there is another path between x and y in the instance, we are dealing with a cycle. Depending on the equation implied by the other path, the cycle is of one of the following types:

(1) If the other path implies the same equation as the first one (i.e. $2x - y = 3$), then the cycle admits infinitely many solutions.
(2) If the other path implies a linear equation that has one common solution with $2x - y = 3$ (e.g. $2x + y = 0$), then the cycle admits a unique solution.

(3) If the other path implies a linear equation that contradicts $2x - y = 3$ (e.g. $2x - y = 1$), then the cycle admits no solution.

A geometric intuition for this classification comes from viewing linear equations as lines in the xy-plane. The lines defined by the two paths of equations between x and y in the instance either coincide (as in Case 1), intersect (as in Case 2) or are parallel (as in Case 3). We say that the cycles formed by paths of these types are *flexible*, *rigid* and *inconsistent*, respectively.

Observe that any acyclic instance (with respect to the underlying primal graph) of 2-Lin($\mathbb{Q}$) is consistent, since we can pick an arbitrary variable, assign any value to it, and then propagate to the remaining variables according to the equations labelling the edges. Thus, any inconsistency in an instance of Min-2-Lin($\mathbb{Q}$) is due to cycles. If an instance contains only flexible cycles, we call it *flexible*, and observe that, similarly to acyclic instances, all flexible instances are consistent. This follows from propagating a value in the same way as above.

With this in mind, we can now sketch an algorithm. By iterative compression, we may assume that we have an over-sized solution X at our disposal i.e. a set of equations of total weight $k + 1$ such that $S - X$ is consistent. In the special case when $S - X$ is not only consistent but flexible, a solution to the instance is a minimum cut Z in $S - X$ that separates vertices $V(X)$ into distinct connected components according to some partition. Recall the problem q-Terminal Multiway Cut that was introduced earlier in this section. We let Multiway Cut denote the problem with arbitrarily many terminals, and we note that it is in FPT [45]. Since $|V(X)| \leq 2k + 2$, we can enumerate partitions of $V(X)$ in fpt time, and compute a minimum cut Z using an fpt-algorithm for Multiway Cut.

In the general case when $S - X$ is not flexible, our strategy is to reduce the case to the flexible one outlined above. Assuming that X is inclusion-wise minimal, every connected component of $S - X$ must contain a rigid cycle (otherwise, there is an edge in X connecting a flexible component with another component of $S - X$, and the equation labelling that edge can be added back to $S - X$ without causing inconsistency). This implies that $S - X$ admits a unique satisfying assignment φ_X. Let φ_Z be the assignment that satisfies $S - Z$. We guess which variables in $V(X)$ have the same value in φ_X and φ_Z and which do not. Let $T \subseteq V(X)$ be the subset of variables that receive different values under these assignments. The key observation is that the change propagates i.e. every variable reachable from T in $S - Z$ has a different value under φ_X and φ_Z. Since rigid cycles in $S - X$ admit a unique satisfying assignment (which is φ_X), none of them can remain in $S - Z$ and be reachable from T. We show that the set of rigid cycles in $S - X$ has the theta property using the fact that fields have Helly dimension 2. This allows us to use the method of important balanced subgraphs to get rid of rigid cycles reachable from the changing terminals T. More specifically, in one of the branches we obtain a set F of size at most k such that the connected components of T in $S - (X \cup F)$ are free from rigid cycles and thus flexible. Moreover, all variables in the remaining components have the same value in φ_X and φ_Z. Thus, we can concentrate on the flexible part of the cleaned instance and use the partition-guessing and cutting idea outlined above. We remark that the reduction to the flexible case is analogous to the shadow removal process of Marx and Razgon [47], but works in $O^*(4^k)$-time instead of $2^{O(k^3)}n^{O(1)}$ (later improved to $2^{O(k^2)}n^{O(1)}$ in [10]) time required by random sampling of important separators. We also note that essentially the same algorithm works for Min-2-Lin($\mathbb{F}$), where $\mathbb{F}$ is an arbitrary field.

General Min-2-Lin Algorithm. Let us now consider the general Min-2-Lin($\mathbb{D}$) problem where $\mathbb{D} \in \mathsf{H}_2$. Important differences between fields and rings in H_2 become apparent even when considering simple structures such as the ring of integers. While in the case of fields all obstructions to consistency of 2-Lin($\mathbb{F}$) instances are cycles, obstructions may now also be paths. For example, consider the following system of equations over $\mathbb{Z}$: $\{y - 2x = 1, y - 2z = 0\}$. While

both equations have integer satisfying assignments (e.g. $(y, x) \mapsto (1, 0)$ and $(y, z) \mapsto (2, 1)$, respectively), they are not simultaneously satisfiable: the equation obtained by cancelling out y is $2z - 2x = 1$ and it has no integer solutions. The restriction to rings with Helly dimension at most two now becomes important: there are no other obstructions besides cycles and paths. This is a very important property that we utilise in our algorithm.

We show that after taking similar steps to those in the Min-2-Lin($\mathbb{Q}$) algorithm (iterative compression, and cleaning by the method of important balanced subgraphs), the solution is again a certain cut in the cleaned graph. However, this time it is not sufficient to partition T into connected components, but we additionally need to break some paths that have no solutions in $\mathbb{D}$. The latter requirements can be expressed as certain disjunctive cut request for the non-terminals. The requests are of the form $(\{x, s\}, \{y, t\})$, where s and t are terminals, and the cut is required to either separate x from s or y from t. We refer to Section 3.2 for the formal definition of the problem. These cut requests are used to deal with inconsistent paths. More concretely, we can check whether the path going from x to s, then from s to t, and finally from t to y is inconsistent in $\mathbb{D}$, and then add the cut request for it. By iterative compression, the optimal solution is disjoint from X, so the part of this path between s and t cannot be cut. Computing all cut requests requires polynomial time: we consider every pair of terminals s, t and non-terminals x, y, and add a corresponding cut request if necessary. To compute a separator that satisfies such disjunctive requests in fpt time, we reduce the cut problem to a special case of the MinCSP parameterized by the solution cost. Kim et al. [39] solve this MinCSP using the recently introduced technique of flow augmentation.

An additional technical issue is that 2-Lin($\mathbb{D}$) is not well understood when $\mathbb{D} \in \mathsf{H}_2$. If $\mathbb{D}$ is a field, then Gaussian elimination solves r-Lin($\mathbb{F}$) for every r in polynomial time. Polynomial-time algorithms are also known for r-Lin($\mathbb{Z}$) and r-Lin($\mathbb{F}[x]$) [33, 34] for every r, but to the best of our knowledge, there are no general algorithms described in the literature that are applicable in our setting, even for the simpler 2-Lin($\mathbb{D}$) problem. This forces us to develop novel methods for checking consistency of 2-Lin($\mathbb{D}$) instances to be used in our Min-2-Lin algorithms, and we present polynomial-time algorithms for 2-Lin($\mathbb{D}$) when $\mathbb{D} \in \mathsf{H}_m$ for $m \in \mathbb{N}$.

Roadmap. The remainder of the paper is structured as follows. In Section 2 we describe the LP-based approach to parameterized deletion problems, define important balanced subgraphs and develop the fpt-algorithm producing a dominating family of important balanced subgraphs. Section 3 contains fpt-algorithms for the graph separation problems (Partition Cut and Pair Partition Cut) used in the Min-2-Lin algorithms. In Sections 4 and 5 we present the general algorithm for rings in H_2 and faster algorithms for fields, respectively.

Section 4 begins by presenting some background concerning commutative rings (Section 4.1) and formally introducing the Helly dimension together with a polynomial-time algorithm for 2-Lin($\mathbb{D}$) when $\mathbb{D} \in \mathsf{H}_m$ (Section 4.2). When presenting the algorithm, we follow the compression–cleaning–cutting structure outlined earlier. Thus, Section 4.3 describes the compression step and Section 4.4 demonstrates how the machinery of balanced important subgraphs (from Section 2) can be used in the cleaning step. In Section 4.5, we show how the resulting instance is reduced to Pair Partition Cut and thus can be solved in fpt-time (by the results in Section 3). We prove correctness of the algorithm and analyze its time complexity in Section 4.6. The complexity analysis shows that Min-2-Lin($\mathbb{D}$) is solvable in time $O^*(2^{k^{O(1)}})$. Finally, we use Section 4.7 for verifying that H_2 contains the so-called *Prüfer* domains and thus confirming that a wide range of well-studied and interesting rings are members of H_2. Section 5 contains faster algorithms for Min-2-Lin($\mathbb{F}$) when $\mathbb{F}$ is a field. The first algorithm is applicable to all reasonably represented fields $\mathbb{F}$ and it runs in time $O^*(k^{O(k)})$. The second algorithm is only applicable to finite fields but it has much better time complexity: the algorithm runs in $O^*((2p)^k)$ where p is the number of elements in $\mathbb{F}$.

Section 6 is devoted to W[1]-hardness results. We show W[1]-hardness of Min-r-Lin($\mathbb{D}$) when $r \geq 3$ and $\mathbb{D}$ is an arbitrary ring with at least two elements. We consider Min-2-Lin($\mathbb{D}$) for commutative rings $\mathbb{D}$ that contain a zero divisor and prove that Min-2-Lin($\mathbb{D}$) is W[1]-hard for many such structures. We finish off in Section 7, summarising and discussing the results, open questions and possible directions for future work. Compared to the preliminary conference version of this article [18], we have extended the algorithm from Euclidean domains to effective Prüfer domains and the rings in H_2 (Section 4) and sharpened the W[1]-hardness proof to apply to a substantially broader class of rings (Section 6).

Preliminaries. We assume familiarity with the basics of graph theory, linear and abstract algebra, and combinatorial optimisation throughout the article. The necessary material can be found in, for instance, the books by Diestel [19], Artin [1], and Schrijver [52], respectively. We use the following graph-theoretic terminology. Let G be an undirected graph. We write $V(G)$ and $E(G)$ to denote the vertices and edges of G, respectively. For every vertex $v \in V(G)$, let the *neighbourhood of v in G* denoted by $N_G(v)$ be the set $\{u \in V(G) \mid \{u,v\} \in E(G)\}$ and the *closed neighbourhood* $N_G[v] = N_G(v) \cup \{v\}$. We extend this notion to sets of vertices $S \subseteq V(G)$ in a natural way: $N_G(S) = (\bigcup_{v \in S} N_G(v)) \setminus S$. If $U \subseteq V(G)$, then the *subgraph of G induced by U* is the graph G' with $V(G') = U$ and $E(G') = \{\{v,w\} \mid v,w \in U \text{ and } \{v,w\} \in E(G)\}$. We denote this graph by $G[U]$. If Z is a subset of edges in G, we write $G - Z$ to denote the graph G' with $V(G') = V(G)$ and $E(G') = E(G) \setminus Z$. For $X, Y \subseteq V(G)$, an *(X,Y)-cut* is a subset of edges Z such that $G - Z$ does not contain a path with one endpoint in X and another in Y. When X, Y are singleton sets $X = \{x\}$ and $Y = \{y\}$, we simplify the notation and write *xy-cut* instead of (X,Y)-cut. We say that a path P is an *xy-path* if its endpoints are the vertices x and y.

2 GRAPH CLEANING

We will now consider one of the cornerstones in our algorithms for Min-2-Lin: *graph cleaning*. The framework we present is intimately connected with *biased graphs*. These are combinatorial objects of importance especially to matroid theory [56]. To introduce biased graphs, we recall that a *theta graph* is a collection of three vertex-disjoint paths with shared endpoints—see Figure 2 for an illustration. A *biased graph* is a pair $(G, \mathcal{B})$ where G is an undirected graph and $\mathcal{B} \subseteq 2^{E(G)}$ is a set of cycles in G (referred to as the *balanced cycles of G*) with the property that if two cycles $C, C' \in \mathcal{B}$ form a theta graph, then the third cycle of $C \cup C'$ is also in $\mathcal{B}$. A set of cycles $\mathcal{B}$ with this property is referred to as a *linear class*. An example of two cycles forming a theta graph is given in Figure 2 with C following $x_1 \to x_2 \to x_4 \to x_1$ and C' following $x_2 \to x_3 \to x_4 \to x_2$. Given a biased graph $(G, \mathcal{B})$, we always assume that $\mathcal{B}$ is defined via a membership oracle that takes as input a cycle C (provided as an edge set) and tests whether $C \in \mathcal{B}$.

The most basic biased graph cleaning problem is the following.

Biased Graph Cleaning (BGC)	
Instance:	A biased graph $(G, \mathcal{B})$ and an integer k.
Question:	Is there a set $X \subseteq V(G)$ such that $\lvert X\rvert \leq k$ and all cycles in $G - X$ are balanced, i.e. members of $\mathcal{B}$?

We will consider an LP-relaxation of BGC and its rooted variant in Section 2.1. Results based on this LP-relaxation will then be used in Section 2.2 where we present fpt-algorithms for various biased graph cleaning problems. These results are directly used in our single-exponential time algorithm for Min-2-Lin over finite fields (Section 5.4). Inspired by these kinds of problems and their solution structure, we introduce the concept of *important balanced subgraphs* in

Section 2.3. Our main result shows that we can efficiently compute a small family of important balanced subgraphs such that every other balanced subgraph is dominated by a member in the set. This forms an important step of our later fpt-algorithms for Min-2-Lin.

Our algorithms exploit half-integral LP-relaxations, a method that historically has proven to be very powerful for attacking this kind of problems. Guillemot [26] was the first to use half-integral LP-relaxations as a tool for constructing fixed-parameter algorithms. The method was refined by Cygan et al. [16] who used it to solve Multiway Cut in time $O^*(2^k)$ and Almost 2-SAT in time $O^*(4^k)$. The latter was further improved by Lokshtanov et al. [42] to $O^*(2.3146^k)$. Iwata et al. [30] generalised the approach using tools from constraint satisfaction problems to provide $O^*(4^k)$-time algorithms for a range of problems, including GFVS as noted above. Iwata et al. [31] later improved this to *linear time* $O^*(4^k)$-time algorithms for an important subclass of these problems, by providing a fast combinatorial solution to the LP-relaxation. Wahlström [54] generalised the GFVS-results of Iwata et al. [30] further to the setting of biased graphs, via the Biased Graph Cleaning problem.

2.1 LP-relaxation for Rooted Biased Graph Cleaning

Previous work by Wahlström [54] shows that BGC has an fpt-algorithm running in $O^*(4^k)$ time. The algorithm is based around a particular LP-relaxation. The workhorse of this result is the following *rooted* variant of BGC. Note that we have extended the problem with vertex weights.

Rooted Biased Graph Cleaning (RBGC)	
Instance:	A biased graph $(G, \mathcal{B})$, a vertex-weight function $w : V(G) \to \mathbb{N}$, a vertex $v_0 \in V(G)$, and an integer k.
Question:	Is there a set $X \subseteq V(G)$ such that $w(X) \leq k$, $v_0 \notin X$, and all cycles in the connected component of v_0 in $G - X$ are balanced?

We will now review the LP-relaxation that underlies the fpt-algorithms for BGC and RBGC. Note that both BGC and RBGC can be defined in terms of seeking a set of vertices X that intersects a class of obstructions. In BGC, the obstructions are simply all unbalanced cycles of $(G, \mathcal{B})$. In RBGC, we can define a class of obstructions consisting of the combination (P, C) of an unbalanced cycle C and a (possibly empty; in the case that v_0 is on C) path P connecting C to v_0. Such a rooted unbalanced cycle is referred to as a *balloon* [54]. Then we see that a set $X \subseteq V(G)$ is a solution to an instance $I = ((G, \mathcal{B}), w, v_0, k)$ of RBGC if and only if X intersects every balloon (where the balloons are implicitly rooted in v_0).

A balloon (P, C) can clearly be decomposed into two paths: for any $v \in V(C) \setminus V(P)$, (P, C) is the union of two paths P_1, P_2 from v_0 to v. The LP-relaxation of an RBGC instance $I = ((G, \mathcal{B}), v_0, w, k)$ is then defined as follows. Let $x \in [0, 1]^{V(G)}$ be an assignment, and for a path P define $x(P) = \sum_{v \in V(P)} x(v)$. The LP-relaxation of I has objective

$$\min \sum_{v \in V(G)} w(v)x(v)$$

subject to the constraints $x(v_0) = 0$, $x(v) \geq 0$ for every $v \in V(G) \setminus \{v_0\}$ and

$$x(P_1) + x(P_2) \geq 1$$

for every balloon (P, C) decomposed into two paths P_1 and P_2. Wahlström [54] showed several properties of this LP. First, an optimal solution can be found in polynomial time, given access to a membership oracle for $\mathcal{B}$. Second, it is *half-integral* i.e. the LP always has an optimum $x^* \in \{0, \frac{1}{2}, 1\}^{V(G)}$. Finally, it is *persistent* in the sense that there is an

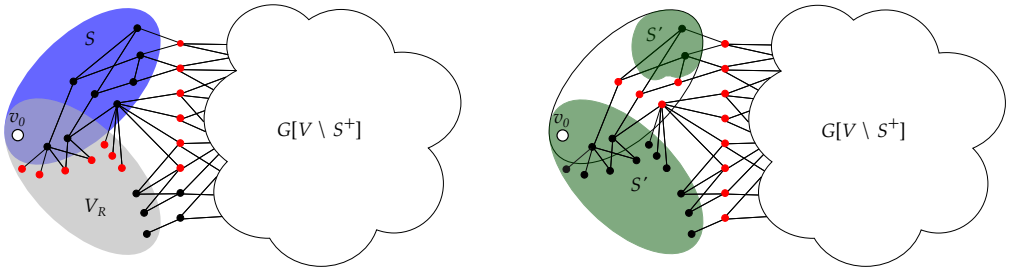


Fig. 4. Two illustrations for Lemma 1. On the left we have a graph with root vertex v_0, the set S such that $G[S]$ is balanced and connected (highlighted in blue) and the set V_R defined by the optimal half-integral LP solution (highlighted in gray). The set S^+ includes S, V_R and all their neighbours. The white cloud corresponds to the remainder of the graph, i.e. $G[V \setminus S^+]$. The blue-gray part is the intersection of S and V_R. On the right we have the same graph with the new set S' (two parts highlighted in green) such that $G[S']$ is balanced and $V_R \subseteq S'$. The deleted vertices in both pictures are indicated in red.

optimal solution $X \subseteq E(G)$ with the following property: if $x^*(v) = 1$, then $v \in X$. Note that for certain vertices v with $x^*(v) = 0$ we can conclude $v \notin X$.

More precisely, we have the following. The *support* of an LP-solution x is the set $\mathrm{supp}(x) = \{v \in V(G) \mid x(v) > 0\}$. Let $I = ((G, \mathcal{B}), v_0, w, k)$ be an instance of RBGC and let $x = V_1 + \frac{1}{2}V_{1/2}$ be a half-integral optimum to the LP-relaxation of I, i.e. $x(v) = 1$ for $v \in V_1$, $x(v) = 1/2$ for $v \in V_{1/2}$ and $x(v) = 0$ otherwise. Let $V_R(x) \subseteq V(G)$ be the set of vertices connected to v_0 in $G - \mathrm{supp}(x)$. Then x is an *extremal LP-optimum* if $V_R(x)$ is maximal among all LP-optima x. If x is a half-integral extremal LP-optimum for I, then there is an optimal solution $X \subseteq V(G)$ to I such that $V_1(x) \subseteq X$ and $V_R(x) \cap X = \emptyset$. Via these properties, we can design an fpt-algorithm for RBGC by a branch-and-bound approach over the LP [54].

In fact, an even stronger, more technical property holds, which we will used in what follows. We give an informal version first. Let $S \subseteq V(G)$ be an arbitrary connected balanced subgraph of G with $v_0 \in S$. The cost of cutting out S from G is $w(N_G(S))$. Now consider the extended set $S^+ = N_G[S \cup V_R(x)]$ which contains both S and $V_R(x)$ as well as their neighbours. One can show that S^+ also contains another set $S' \subseteq V(G)$ such that $G[S']$ is balanced; moreover, $V_R(x) \subseteq S'$, $N_G(S') \subseteq S^+$ and $w(S^+ \setminus S') \leq w(N_G(S))$. Thus, if we are solving BGC problem, then, informally, instead of "covering" $N_G[S]$ by deleting $N_G(S)$ we can cover a possibly larger set S^+ by deleting vertices of at most the same cost, and we can replace $G[S]$ with $G[S']$. In terms of RBGC, this means that we always have a balanced subgraph of minimum cost that includes V_R. See Figure 4 for an illustration. We will use this lemma to prove Lemma 8, which contains a simpler version of the statement for the edge-deletion case, and then to derive Lemma 9, which is a concise and important consequence.

Lemma 1 (Wahlström [54, Lemma 6]). *Let $x = V_1 + \frac{1}{2}V_{1/2}$ be a half-integral extremal LP-optimum for a* RBGC *instance $I = ((G, \mathcal{B}), w, v_0, k)$ and let $V_R(x)$ be defined as above. Let $S \subseteq V(G)$ be a vertex set with $v_0 \in S$ such that $G[S]$ is balanced and connected. Then there is a set of vertices S^+ and a set $S' \subseteq S^+$ such that $G[S']$ is balanced and the following hold.*

(1) $S^+ = N_G[S \cup V_R(x)]$;
(2) $N_G[S'] \subseteq S^+$;
(3) $V_R(x) \subseteq S'$;
(4) $V_1(x) \subseteq (S^+ \setminus S')$;
(5) $w(S^+ \setminus S') \leq w(N_G(S))$.

2.2 Biased Graph Cleaning

In this section we consider further variants of biased graph cleaning problems. Our goal is to show that the vertex-weighted extension of Biased Graph Cleaning and the edge version of Rooted Biased Graph Cleaning (RBGCE) both admit fpt-algorithms. These results are standard, but they are needed for later results in the paper.

We begin by noting that RBGC is in FPT, using the same LP-branching algorithm as in the unweighted case [54].

Proposition 2 (Wahlström [54, Theorem 1])**.** RBGC *admits an fpt-algorithm with running time* $O^*(2^k)$ *assuming that* $\mathcal{B}$ *is given by a polynomial-time membership oracle.*

Proof sketch. Let $I = ((G, \mathcal{B}), w, v_0, k)$ be an instance of RBGC, where $(G, \mathcal{B})$ is a biased graph, $w : V(G) \to \mathbb{N}$ is a weight function, $v_0 \in V(G)$ is the root vertex and $k \in \mathbb{N}$ is the deletion budget. We note that the algorithm for the unweighted case [54, Lemma 8] also applies in the presence of integer weights. We use the slightly more careful version of the extended preprint [41]. Let $W = \sum_{v \in V(G)} w(v)$, and assume $k < W$, as otherwise the instance is trivial. For a vertex $v \in V(G)$, we say *fix* $v = 0$ to refer to setting a weight $w(v) = 2W$, and *fix* $v = 1$ to refer to setting $w(v) = 0$. Let x^* be a half-integral extremal LP-optimum as in Lemma 1; such an optimum can be computed efficiently in a greedy manner [41]. Let λ be the cost of x^*. If $\lambda > k$, then we reject the instance. Otherwise $\lambda < W$ and as x^* is half-integral, if we have fixed $v = 0$, then we must have $x^*(v) = 0$. Furthermore, if $\lambda < k/2$, then $\mathrm{supp}(x^*)$ is an integral solution of cost at most k. Now, as in [41], we either find an integral LP-optimum, or we find a half-integral vertex $v \in \mathrm{supp}(x^*)$ such that $x^*(v) = 1/2$, fixing $v = 0$ increases the LP-optimum cost, and $w(v) > 0$. Then we can recursively branch on fixing $v = 0$, and on fixing $v = 1$ and decreasing k by $w(v)$. In the former case λ increases by at least $1/2$ since the LP is half-integral, and k is unchanged. In the latter case λ decreases by $w(v)/2$, but k decreases by $w(v)$. Hence the value of $k - \lambda$ decreases by at least $1/2$ in both branches. It follows that an exhaustive branching takes $O^*(2^{2(k-\lambda)})$ time, and since $\lambda \geq k/2$ initially, this is $O^*(2^k)$. □

Next, we consider the *edge deletion* version of the problem, Rooted Biased Graph Cleaning by Edge Deletion (RBGCE), which is defined similarly to RBGC except that the solution is an edge set, not a vertex set, and where the input comes with edge weights w instead of vertex weights. By a standard reduction, this problem is also in FPT.

Proposition 3. RBGCE *admits an fpt-algorithm with running time* $O^*(2^k)$ *assuming that* $\mathcal{B}$ *is given by a polynomial-time membership oracle.*

Proof. Let $I = ((G, \mathcal{B}), w, v_0, k)$ be an instance of RBGCE. We provide a polynomial-time and parameter preserving reduction to RBGC, which together with Proposition 2 shows the proposition. The instance $I' = ((G', \mathcal{B}'), w', v_0', k')$ of RBGC is obtained from I as follows. We set $k' = k$. The graph G' is obtained from G by subdividing every edge of G exactly once. We set the weight w' of every original vertex to $k + 1$ and the weight of every new (subdividing) vertex x_e, subdividing an edge e, to $w'(x_e) = w(e)$. Finally, we let $\mathcal{B}'$ be the set of all cycles C in G' such that the cycle obtained from C after reversing the subdivision is in $\mathcal{B}$. This completes the construction of I', which can clearly be achieved in polynomial-time and is parameter preserving. It remains to show that I is a yes-instance if and only if I' is a yes-instance.

Towards showing the forward direction, let $X \subseteq E(G)$ be a solution for I and let $X' \subseteq V(G')$ be the set of vertices used for subdividing the edges in X. We claim that X' is a solution for I'. Suppose for contradiction that this is not the case. Then, there is a cycle C' in $G' - X'$ reachable from v_0 with $C' \notin \mathcal{B}'$. But then, the cycle C obtained from C' after reversing the subdivision of all edges is also in $G - X$ and reachable from v_0. Finally, because $C' \notin \mathcal{B}'$, we obtain that $C \notin \mathcal{B}$, contradicting our assumption that X is a solution of I.

Towards showing the reverse direction, let $X' \subseteq V(G')$ be a solution for I'. Then, because the weight of every original vertex is $k+1$, X' only contains vertices used for the subdivision of edges of G. We claim that the set X containing all edges of G whose corresponding vertex in G' is in X' is a solution for I. Suppose for contradiction that this is not the case and there is a cycle C in $G - X$ reachable from v_0 with $C \notin \mathcal{B}$. Let C' be the corresponding cycle in G', i.e. C' is obtained from C after subdividing each edge of C. Then, C' is also in $G' - X'$ because X' does not contain any (original) vertex of C. Moreover, C' is reachable from v_0 and $C' \notin \mathcal{B}'$, a contradiction to our assumption that X' is a solution for I'. □

On a side note, we observe that the standard, non-rooted (but weighted) problem BGC and its edge-deletion variant BGCE are in FPT. The result follows from same procedure as in the original paper [54], building on Propositions 2 and 3.

Proposition 4. *The integer-weighted variants of* BGC *and* BGCE *both admit an fpt-algorithm with run-time* $O^*(4^k)$ *assuming that* $\mathcal{B}$ *is given by a polynomial-time membership oracle, where* k *is the total weight of a solution.*

2.3 Important Balanced Subgraphs

Important separators are a central concept in fpt-algorithms for graph separation problems that was originally defined by Marx [45]. Let G be an undirected graph, and let $X, Y \subset V(G)$ be disjoint sets of vertices. For an (X, Y)-cut $C \subseteq V(G)$, let $R(X, C)$ be the set of vertices reachable from X in $G - C$. Then C is an *important* (X, Y)*-separator* if, for every (X, Y)-cut C' such that $|C'| \leq |C|$ and $R(X, C) \subseteq R(X, C')$, we have $C' = C$. In other words, for any (X, Y)-cut C' such that $R(X, C) \subsetneq R(X, C')$ we must have $|C'| > |C|$.[3] Marx showed that for any graph G and sets X, Y, there are at most $f(k)$ distinct important separators C with $|C| \leq k$ [45], and this bound was later improved to $f(k) = 4^k$ (see [14]). The same bound applies to both undirected and directed graphs, and by using standard reductions it also applies to edge cuts. Important separators are a key component in many fpt-algorithms, including the algorithms for Multiway Cut [45] and Multicut [47] (see Cygan et al. [14] for more applications).

We show a new result on the solution structure of RBGCE that generalizes important separators for undirected edge cuts. We first note that the number of (in some sense) incomparable solutions to RBGCE is not bounded in k. Indeed, if C is an unbalanced cycle on n vertices, then deleting any one edge of C is a minimal solution, and there is no clear order of preference between these solutions. On the other hand, it turns out that a result in the style of important separators does hold in terms of vertex sets of balanced connected subgraphs of a biased graph.

Let us introduce some terminology. Let $(G, \mathcal{B})$ be a biased graph and let $w\colon E(G) \to \mathbb{Z}$ be a set of edge weights. Let H be a subgraph of G. Let $\delta_G(X)$ for $X \subseteq V(G)$ denote the set of edges in G with precisely one endpoint in X. We then define the *cost of* H as the cost of the edges in G incident with $V(H)$ but not present in H i.e.

$$c_G(H) = w(E(G[V(H)]) \setminus E(H)) + w(\delta_G(V(H))).$$

We refer to $(E(G[V(H)]) \setminus E(H)) \cup \delta_G(V(H))$ as the *deleted edges* of H.

Let H and H' be balanced subgraphs (with respect to $\mathcal{B}$) of G. We say that H' *dominates* H if $V(H) \subseteq V(H')$ and $c_G(H) \geq c_G(H')$, and that H' *strictly dominates* H if at least one of these two inequalities is strict. Analogously to important separators, we refer to H as an *important balanced subgraph* in $(G, \mathcal{B})$ if H is a connected, balanced subgraph of G and no balanced subgraph H' of G strictly dominates H. We refer the reader to Figure 3 for an illustration. We

[3] One can alternatively view important (X, Y)-separators as minimal (X, Y)-cuts with respect to the quasi-order $\preceq$ defined as $C' \preceq C$ if and only if $|C'| \leq |C|$ and $R(X, C) \subseteq R(X, C')$.

may assume here without loss of generality that H' is also connected (see Lemma 7). Importantly, observe that if H is dominated by H', then H might not be a subgraph of H'. In the example of a single unbalanced cycle C, the subgraphs $C - \{e\}$ for $e \in E(C)$ all mutually dominate each other, although not strictly.

Let $\mathcal{G} := \mathcal{G}(G, \mathcal{B}, w, kv_0)$ be the family of connected balanced subgraphs in $(G, \mathcal{B})$ that contain v_0 and have cost at most k under w. A subset $\mathcal{H} \subseteq \mathcal{G}$ is a *dominating family for* $\mathcal{G}$ if for any $H \in \mathcal{G}$ there is a subgraph $H' \in \mathcal{H}$ that dominates H. We show the following result.

Theorem 5 (Dominating family of important balanced subgraphs)**.** *Let $(G, \mathcal{B})$ be a biased graph with positive integer edge weights w, let $v_0 \in V(G)$ and let k be an integer. Let $\mathcal{G} := \mathcal{G}(G, \mathcal{B}, w, k, v_0)$ be the family of connected balanced subgraphs in $(G, \mathcal{B})$ that contain v_0 and have cost at most k. Then, given (G, w, k, v_0) and oracle access to $\mathcal{B}$ we can compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ such that $|\mathcal{H}| \leq 4^k$ in $O^*(4^k)$-time. Furthermore, every member of $\mathcal{H}$ is an important balanced subgraph of $(G, \mathcal{B})$.*

Before we present our proof of Theorem 5, we illustrate that important biased subgraphs are indeed a generalisation of important separators.

Example 6. Let G be an undirected graph and $s, t \in V(G)$ be distinguished vertices. Note that since we are considering edge cuts, the assumption that s and t are single vertices (as opposed to disjoint vertex sets X and Y) can be made without loss of generality. Now, add two vertices z, z' and three edges $e_1 = \{t, z\}$, $e_2 = \{z, z'\}$, $e_3 = \{t, z'\}$. Let G' be the resulting graph, and let $\mathcal{B}$ be the set containing all cycles except $C_t = \{e_1, e_2, e_3\}$. Note that $\mathcal{B}$ trivially defines a linear class since C_t is not part of any theta graph, so $(G', \mathcal{B})$ is a biased graph. Finally, set edge weights $w(e_i) = k + 1$, for $i \in \{1, 2, 3\}$, and $w(e) = 1$ for every other edge $e \in E(G)$, and use $v_0 = s$ as the root vertex. Then a connected subgraph H of G' with $s \in V(H)$ and of cost at most k is balanced if and only if $t \notin V(H)$ (since breaking the unbalanced cycle C_t would exceed the budget). Hence H is a connected, balanced subgraph of G' with $s \in V(H)$ and of cost at most k if and only if the set of deleted edges C of H contains an st-cut in G. Furthermore, in such a case $V(H) = R(\{s\}, C)$. We see that the important (s, t)-separators in G of cost at most k directly correspond to the deleted edges of important balanced subgraphs H of G' with $s \in V(H)$ and of cost at most k.

We proceed to prove Theorem 5; this occupies the rest of the subsection. We first note that we may assume that strictly dominating subgraphs are connected.

Lemma 7. *Assume that G is a connected undirected graph that has no zero-weight edges. Let H be a connected balanced subgraph of G. If there is a balanced graph H' that strictly dominates H, then there is also a connected balanced graph H' that strictly dominates H. Furthermore, if H' is of minimum cost among all balanced graphs that dominate H, then H' is connected.*

Proof. Assume that there exists a balanced subgraph H' of G that strictly dominates H, and let H' be chosen to minimise $c_G(H')$ among all such subgraphs H'. Suppose that H' is not connected, and first suppose that H' contains a connected component C such that $C \cap V(H) = \emptyset$. Then $H' - C$ is balanced, $c_G(H' - C) < c_G(H')$ since G is connected, and $V(H) \subseteq V(H' - C)$. Hence $H' - C$ also dominates H, and would be the preferred choice over H'. Hence we proceed assuming that every connected component of H' intersects $V(H)$. Now let $e \in E(H)$ be an edge connecting distinct connected components in H'. Such an edge clearly exists, e.g. follow a path in H whose endpoints lie in distinct components of H'. Then adding e to H' yields another balanced subgraph H'', since no cycle passes through e in H''. Then $c_G(H'') < c_G(H')$ and H'' dominates H; hence by choice of H', no such edge can exist. We conclude that H' is connected. □

For the proof of Theorem 5, we begin by adapting the LP-relaxation for RBGC and Lemma 1 to the edge-deletion version RBGCE. More precisely, the LP-relaxation for RBGC and Lemma 1 are both defined in terms of a solution space $x \in \{0, 1/2, 1\}^{V(G)}$ of half-integral relaxed solutions over the vertex set of a graph. We use the reduction from RBGCE to RBGC provided in the proof of Proposition 3 to derive half-integral optimal solutions $x^* \in \{0, 1/2, 1\}^{E(G)}$ to the edge-deletion version of the problem. We also observe the following persistence properties of such a solution x^*, as a simplification of Lemma 1. We refer to these solutions as the LP-relaxation of RBGCE (indeed, they are a projection of the solutions to the LP-relaxation of the RBGC-instance resulting from the reduction, so they correspond to an LP over variables $E(G)$).

Lemma 8. *Let $I = ((G, \mathcal{B}), w, v_0, k)$ be an instance of* RBGCE. *In polynomial time, we can compute a half-integral extremal optimum $x^* = X_1 + \frac{1}{2}X_{1/2}$ of the LP-relaxation of I such that the following holds. Let $X = X_1 \cup X_{1/2}$ be the support of x^*, $X \subseteq E(G)$. Let G_R be the subgraph consisting of edges reachable from v_0 in $G - X$. Let H be any connected balanced subgraph of G with $v_0 \in V(H)$. Then there is a balanced subgraph H' of G on vertex set $V(G_R) \cup V(H)$ such that G_R is a subgraph of H', $c_G(H') \leq c_G(H)$, and $X_1 \cap E(H') = \emptyset$.*

In particular, unless $V(G_R) \subseteq V(H)$, there is a graph H' that strictly dominates H and has $V(G_R) \subseteq V(H')$.

Proof. Let $(G', \mathcal{B}')$ be the biased graph obtained from $(G, \mathcal{B})$ by subdividing every edge $e \in E(G)$ by a new vertex z_e. Here, $\mathcal{B}'$ contains a cycle C' if and only if it is a subdivision of a cycle $C \in \mathcal{B}$. Apply Lemma 1 to $(G', \mathcal{B}')$ giving every vertex $v \in V(G)$ weight $w(v) = 2w(E(G)) + 1$ and the subdividing vertices weight 1. Let x be the resulting half-integral LP-optimum, and define $x^* \in \{0, 1/2, 1\}^{E(G)}$ as $x^*_e = x_{z_e}$ for all $e \in E(G)$, and define G_R accordingly. For an edge $e \in E(G)$, we say that the vertex z_e which subdivides e in G' *represents* e in G'. For a subgraph G_0 of G, let $V'(G_0) \subseteq V(G')$ contain the copy in G' of every vertex $v \in V(G_0)$ as well as the vertex z_e subdividing e for every edge $e \in E(G_0)$. Note that V' maps connected, respectively balanced subgraphs of G to vertex sets S such that $G'[S]$ is connected, respectively balanced.

Consider the vertex sets $S = V'(H)$ and $R = V'(G_R)$. Since $G'[S]$ is balanced and connected, Lemma 1 provides sets $S', S^+ \subseteq V(G')$, where $R \subseteq S'$, $N_{G'}[S'] \subseteq S^+$ and $S^+ = N_{G'}[S \cup R]$. Let H' be the subgraph of G defined by S', i.e. $V(H') = S' \cap V(G)$ and $e \in E(H')$ for $e \in E(G)$ if and only if the vertex subdividing e is contained in S'. We claim that $V(H') = V(H) \cup V(G_R)$. In one direction, $V(H') \subseteq V(H) \cup V(G_R)$, since $S' \subseteq S^+ = N_{G'}[S \cup R]$ and every vertex of $N_{G'}(S \cup R)$ represents an edge in G. In the other direction, we claim $V(H) \cup V(G_R) \subseteq S'$. Indeed, $V(G_R) \cup V(H) \subseteq S^+ = N_{G'}[S \cup R]$, and if there were a vertex $v \in V(G) \cap (S^+ \setminus S')$, then the cost of $S^+ \setminus S$ would exceed $w(N_{G'}(S)) = c_G(H)$. Thus $V(H') = V(G_R) \cup V(H)$. Furthermore H' is balanced, since any unbalanced cycle in H' would correspond to an unbalanced cycle in $G'[S']$.

Next, we note that G_R is a subgraph of H' since $R = V'(G_R) \subseteq S'$. Finally,

$$c_G(H') \leq w(S^+ \setminus S') \leq w(N_{G'}(S)) = c_G(H),$$

and $X_1 \cap E(H') = \emptyset$, since the vertices subdividing X_1 are contained in $S^+ \setminus S'$.

For the last part, let H' be the subgraph produced above from G_R and H. Then H' is balanced, $c_G(H') \leq c_G(H)$, and, unless $V(G_R) \subseteq V(H)$, $V(H') = V(G_R) \cup V(H)$ is a strict superset of $V(H)$. □

We show one more property of the LP-relaxation. Recall that the constraints of the LP are written as $x(P_1) + x(P_2) \geq 1$ for every balloon $B = (P, C)$ decomposed into two paths P_1 and P_2. Equivalently, for every balloon $B = (P, C)$, there is a constraint where the edges of P have coefficient 2, and the edges of C have coefficient 1. The constraint for $B = (P, C)$ is *tight* if equality holds in the constraint. By so-called *slackness conditions*, it is known that for any LP optimum x^* and

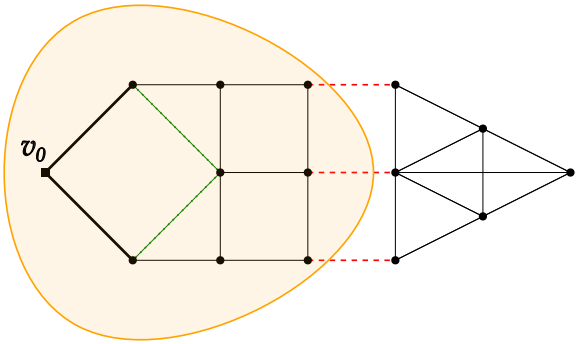


Fig. 5. Illustration of a half-integral extremal optimum $x^* = X_1 + X_{1/2}$ to the LP-relaxation of the RGBCE instance $(G, \mathcal{B}, w, v_0, k)$, where G is the graph in the picture, $\mathcal{B}$ is the set of even cycles in G, the root v_0 is the leftmost vertex, and w assigns weight 2 to the edges incident to v_0 (bold), and unit weight to all other edges. In the picture, the edges of X_1 are in green and dotted, the edges of $X_{1/2}$ are in red and dashed, and the yellow area encloses the subgraph G_R. Here $c_G(G_R) = |X_1| + |X_{1/2}| = 5$, while the LP-objective value of x^* is $|X_1| + \frac{1}{2}|X_{1/2}| = 3\frac{1}{2}$.

any edge e in the support of x^* there is a tight constraint involving e, i.e. a balloon $B = (P, C)$ with $e \in E(B)$ such that the constraint for B is tight.

Lemma 9. *Let $x^* = X_1 + \frac{1}{2}X_{1/2}$ be a half-integral extremal optimum computed in Lemma 8. Let $X = X_1 \cup X_{1/2}$, let G_R be the subgraph corresponding to the connected component of v_0 in $G - X$, and let $V_R = V(G_R)$. Then $X_1 = E(G[V_R]) \setminus E(G_R)$ and $X_{1/2} = \delta_G(V_R)$.*

Proof. For the first item, let $e \in X_1$. By the slackness conditions, there must be a tight constraint in the LP which contains e. By inspection of the constraints, this implies that there is a balloon $B_e = (P, C)$ rooted in v_0 such that e is contained in B_e. Since $x^*(e) = 1$, it may only appear with coefficient 1 in the summation of the constraint, hence $e \in C$. Moreover, we have $x^*(e') = 0$ for every edge $e' \in P \cup C \setminus \{e\}$. Then $B_e - e$ is contained in G_R. Similarly, let $e \in X_{1/2}$ and assume towards a contradiction that e is spanned by G_R, i.e. $e \subseteq V_R$. Let $B_e = (P, C)$ be a balloon with $e \in E(B_e)$ such that the corresponding constraint is tight. There are two cases. First suppose that e occurs in the path P of B_e. Since e occurs with coefficient 2 in the constraint corresponding to B_e, for every other edge $e' \in E(B_e)$ we must have $x^*(e') = 0$. But since $e \subseteq V_R$, this implies that every vertex of B_e is in V_R. In particular, there is a path from v_0 to C entirely contained in G_R, and considering a shortest such path we find a path P' that is internally disjoint from C. This produces a balloon $B'_e = (P', C)$ disjoint from X, which is a contradiction. Next, suppose that $e \in E(C)$. Then by tightness, $V(C) \subseteq V_R$. Indeed, by tightness C intersects precisely two edges of $X_{1/2}$ and none of X_1, and since $e \notin \delta_G(V_R)$ by assumption, it follows that both edges of $E(C) \cap X$ are spanned by G_R, i.e. $V(C) \subseteq V_R$. Then $C \setminus X_{1/2}$ splits into two paths P_1 and P_2, where one of them may be edgeless but both consist entirely of vertices of V_R. Let P' be a shortest path in G_R from P_1 to P_2. Then P' forms a chordal path for the unbalanced cycle C, hence results in at least one new unbalanced cycle C' of weight $1/2$ in x^*. Furthermore, there is a path P'' contained in G_R forming a balloon $B'_e = (P'', C')$ of weight $1/2$ in x^*, which is a contradiction to x^* being an LP solution. Hence $X_{1/2} = \delta_G(V_R)$. □

See Figure 5 for an illustration how a half-integral extremal optimum may look like according to Lemma 9.

We can now show the main result. The proof is based on a branching procedure for the problem, i.e. a recursive enumeration algorithm where at every step the solution space is partitioned by branching (in our case, on whether

certain edges should be included into a solution or not) and a bounding function is used to eliminate recursive calls that will not lead to a solution. The bounding function is based on the LP-relaxation for the edge-deletion version of RBGCE.

Proof of Theorem 5. We assume that G is connected, or otherwise restrict our attention to the connected component of G containing the vertex v_0. Furthermore, by assumption the edge weights of G are positive. Hence Lemma 7 applies. Now, recall that $\mathcal{G}$ denotes the family of all connected, balanced subgraphs in $(G, \mathcal{B})$ that contain v_0 and have cost at most k and let $\mathcal{H}' \subseteq \mathcal{G}$ be all subgraphs $H \in \mathcal{G}$ that are not strictly dominated by any member of $\mathcal{G}$. We observe that every member of $\mathcal{H}'$ is important. Indeed, let $H \in \mathcal{G}$ and assume that there is a balanced subgraph H' of $(G, \mathcal{B})$ that dominates H. Choose H' to minimise $c_G(H')$. Then by Lemma 7 H' is connected. Furthermore $c_G(H') \leq c_G(H) \leq k$ and $v_0 \in V(H) \subseteq V(H')$. Thus $H' \in \mathcal{G}$. Thus any subgraph H of $(G, \mathcal{B})$ that is "domination maximal" within $\mathcal{G}$ is important in $(G, \mathcal{B})$, and we can focus on computing a dominating family $\mathcal{H} \subseteq \mathcal{G}$.

For this, we present a branching procedure over the LP-optimum. Let a *branching state* be defined by a tuple (E_0, E_1) where $E_0, E_1 \subseteq E(G)$ are disjoint edge sets. For a branching state $B = (E_0, E_1)$, we let $LP_e(B)$ denote the LP on the graph $G - E_1$, with edge weights modified so that $w(e) = 2k + 1$ for every $e \in E_0$. Intuitively, edges in E_0 can be thought of as undeletable while edges in E_1 as deleted. We let B^* denote the half-integral solution to $LP_e(B)$ and let G_B denote the corresponding subgraph of G, i.e. G_B is the connected component of $G - (E_1 \cup \operatorname{supp}(B^*))$ containing v_0. Let us consider the following branching procedure.

(1) Let $B = (E_0, E_1)$ be a branching state that initially is set to $(\emptyset, \emptyset)$.
(2) Let X_1 and $X_{1/2}$ be such that $B^* = X_1 + \frac{1}{2}X_{1/2}$ and let $X = X_1 \cup X_{1/2}$ be the support. Let k' be the cost of B^*.
(3) If $|E_1| + k' > k$, then abort the branch without output.
(4) If $X_{1/2} = \emptyset$, output G_B as a potential solution and abort the branch.
(5) Otherwise, let $B' = (E'_0, E'_1)$ with $E'_0 = E_0 \cup E(G_B)$ and $E'_1 = E_1 \cup X_1$.
(6) Let $e \in X_{1/2}$ be an arbitrary half-integral edge and branch recursively on the two states $B_1 = (E'_0 \cup \{e\}, E'_1)$ and $B_2 = (E'_0, E'_1 \cup \{e\})$.

We will show that for any balanced, connected subgraph H of G with $c_G(H) \leq k$, at least one of the produced subgraphs G_B dominates H. Towards this, we need some support claims about the branching process.

Claim 5.1. *In every branching state $B = (E_0, E_1)$ encountered by the algorithm, the edge set E_0 forms a balanced connected subgraph of G rooted in v_0.*

Proof of claim: We choose to interpret the initial empty edge set as the subgraph of G containing the root v_0 and no edges or any further vertices. The claim now holds by induction from the root. Note that there are two places where the E_0-part of a branching state is modified. First, let $B = (E_0, E_1)$ be a branching state and let B^* be the half-integral optimum of $LP_e(B)$ used by the algorithm. Assume that the cost of B^* is at most $k - |E_1|$ as otherwise no further branching state is produced. By assumption, E_0 forms a connected subgraph of G, and every edge of E_0 has cost $2k + 1$ in $LP_e(B)$. Hence $B^*(e) = 0$ for every $e \in E_0$, and $E_0 \subseteq E(G_B)$. Thus in the new branching state $B' = (E'_0, E'_1)$ we in fact have $E'_0 = E(G_B)$ which is a connected, balanced, rooted subgraph of G by construction. Otherwise, assume that a new state is formed as $B' = (E_0 \cup \{e\}, E_1)$ for some edge e that is half integral in $LP_e(B)$. Then by Lemma 9, e is an edge leaving G_B, hence $E'_0 = E(G_B) \cup \{e\}$ forms a connected subgraph. Finally, we note that E'_0 is balanced, since otherwise there would exist an unbalanced cycle C in E'_0 using the edge e, but since e is leaving G_B, e is a pendant edge in E'_0. ⋄

Claim 5.2. *In every branching state $B = (E_0, E_1)$ encountered by the algorithm, every edge of E_1 has at least one endpoint in $V(E_0)$.*

Proof of claim: Shown by induction. In the initial state $(\emptyset, \emptyset)$, it holds vacuously. Thereafter, the E_1-part of a branching state is modified in two ways. First, let $B = (E_0, E_1)$ be a branching state and let $B^* = X_1 + \frac{1}{2}X_{1/2}$ be the optimum of $LP_e(B)$. Let G_B be the corresponding subgraph of G and let $B' = (E'_0, E'_1)$ be the new resulting branching state. Then $E'_1 = E_1 \cup X_1$, edges of E_1 intersect $V(E_0) \subseteq V(E'_0)$ by assumption, and edges of X_1 are spanned by $E(G_B) \subseteq E'_0$ by Lemma 9. Otherwise, we have a modification $E'_1 = E_1 \cup \{e\}$ for some $e \in X_{1/2}$, where e intersects $V(E_0)$ by Lemma 9. $\diamond$

We say that H is *compatible with* a branching state $B = (E_0, E_1)$ if $E_0 \subseteq E(H)$ and $E_1 \cap E(H) = \emptyset$. Note that it follows that every edge of E_1 is deleted in H; indeed, by Claim 5.2 every edge of E_1 intersects $V(H)$, and every edge intersecting $V(H)$ not present in H is deleted in H. Also say that H is *domination compatible with* B if there is a balanced, connected subgraph H' of G rooted in v_0 such that H' dominates H and is compatible with B. Note that if H is domination compatible with a leaf state in the branching tree, then the subgraph G_i produced in this state dominates H. Indeed, let $B(E_0, E_1)$ be the leaf branching state, and $G_i = G_B$. By assumption there is a graph H' dominating H, compatible with B. Then $\delta_G(G_B) \subseteq E_1$, and $E_1 \cap E(H') = \emptyset$. Furthermore $E(G_B) = E_0 \subseteq E(H')$. Hence $V(H') = V(G_B)$, and the cost of G_B is optimal among all such graphs by the integrality of the LP solution $L_e(B)$.

We can now prove by induction that for every balanced, connected subgraph H of G rooted in v_0 with $c_G(H) \leq k$, the branching process will produce at least one balanced subgraph G_i that dominates H. We claim by induction that for every level ℓ of the branching tree, either such a graph G_i has been produced at a preceding level or there is a state on level ℓ domination compatible with H.

In the root node, we have the initial branching state $(\emptyset, \emptyset)$, where we can choose $H' = H$. Inductively, first assume that $B = (E_0, E_1)$ is a branching state domination compatible with H via a graph H' dominating H, and let $B^* = X_1 + \frac{1}{2}X_{1/2}$ be the optimum of $LP_e(B)$. Let $B' = (E'_0, E'_1)$ be the new resulting branching state. We will show that H' is domination compatible with B', hence the same holds for H.

Recall that $LP_e(B)$ is defined in the subgraph $G' := G - E_1$. Let G_B be the subgraph of G' corresponding to the optimum B^*. By Lemma 8 there is a balanced subgraph H'' of G' that dominates H' in G', such that G_B is a subgraph of H'' and $X_1 \cap E(H'') = \emptyset$. We need to show that $E'_0 \subseteq E(H'')$, that $E'_1 \cap E(H'') = \emptyset$, that $c_G(H'') \leq c_G(H')$, and that $V(H'') \supseteq V(H')$. It then follows that H'' dominates H' in G and is compatible with B'.

For the first, as before we have $E_0 \subseteq E(G_B)$ by construction so $E'_0 = E_0 \cup E(G_B) = E(G_B) \subseteq E(H'')$. For the second, since H'' is a subgraph of $G - E_1$ disjoint from X_1, we have $E'_1 \cap E(H'') = \emptyset$. For the cost, we have $c_{G'}(H'') \leq c_{G'}(H')$ by Lemma 8. As noted above, every edge of E_1 is deleted in H'; hence $c_{G'}(H') = c_G(H') - |E_1|$. Similarly, since every edge of E_1 intersects $V(E'_0)$ by Claim 5.2 and $E'_0 \subseteq E(H'')$, every edge of E'_1 is deleted in H'' with respect to G. Thus $c_G(H'') = c_{G'}(H'') - |E_1| \leq c_{G'}(H') - |E_1| = c_G(H')$. Finally, $V(H'') \supseteq V(H')$ by Lemma 8. Thus H' is domination compatible with B'.

The only remaining step to consider is when a branching state is modified as $B = (E_0, E_1) \mapsto (E_0 \cup \{e\}, E_1)$ or $(E_0, E_1) \mapsto (E_0, E_1 \cup \{e\})$ for some edge e that is half-integral in $LP_e(B)$. However, by assumption there exists a subgraph H' that dominates H and is compatible with B. Then either $e \in E(H')$ or $e \notin E(H')$, and precisely one of the two new branching states is compatible with H'. Furthermore, by comparing Lemma 9 to the definition of the cost function $c_G(H')$, it is clear that the cost of the resulting state does not exceed $c_G(H') \leq c_G(H) \leq k$. Hence by induction, there is a leaf in the branching tree which is domination compatible with H.

Finally, we claim that the whole process produces at most 4^k outputs and can consequently be performed in $O^*(4^k)$ time. To see this, we use an approach that is similar to the one used in [54]. Consider the value of the "LP gap" $k - (|E_1| + k')$ computed in some node of the branching tree corresponding to the above computation. Clearly, this

value is initially at most k, and if it is negative in a node, then that branch of the computation is aborted. We claim that furthermore, this gap decreases by at least $1/2$ from a branching state B to both of its children B_1 and B_2. In the branching state B_1, B_1^* is also a valid solution to the state B, and in the branching state B_2, B_2^* becomes a valid solution to the state B if we modify the value of e to $x_e = 1$. In both cases, we get a valid LP solution to the state B. We claim that these solutions cannot be optimal for $LP_e(B)$. On the one hand, if $E_0 \mapsto E_0 \cup \{e\}$ then the set of reachable vertices $V(G_B)$ increases strictly. Since the extremal solution B^* is chosen so that this set is maximal among all LP-optima, the result cannot be an LP-optimum. On the other hand, if $E_1 \mapsto E_1 \cup \{e\}$ and the resulting branching state produces an optimal solution for $LP_e(B)$, then by Lemma 9 the endpoints of e must be spanned by the resulting set $E_0' \supseteq E_0$, which again contradicts the choice of $V(G_B)$ as maximal. Thus, the cost of these solutions is greater than the cost of B^*. Since the cost is half-integral (given integral edge weights), this difference is at least $1/2$. Hence the entire branching process will finish at depth at most $2k$, producing at most 2^{2k} outputs. □

3 GRAPH PARTITIONING

As discussed in the introduction, the general strategy for our fpt-algorithms aims to reduce Min-2-Lin over various domains to graph partitioning problems. In this section we develop algorithms for two problems—Partition Cut and Pair Partition Cut—which arise in the study of Min-2-Lin over fields and the rings in H_2, respectively.

3.1 Partition Cut

A partition $\mathcal{P}$ of a finite set N is a family of pairwise disjoint subsets $B_1, \ldots, B_m$ of N such that $\bigcup_{i=1}^{m} B_i = N$. For any $x, y \in N$, we write $\mathcal{P}(x) = \mathcal{P}(y)$ if x and y appear in the same subset of $\mathcal{P}$, while $\mathcal{P}(x) \neq \mathcal{P}(y)$ if they appear in distinct subsets. If $\mathcal{P}'$ is a partition of N such that $\mathcal{P}'(x) = \mathcal{P}'(y) \implies \mathcal{P}(x) = \mathcal{P}(y)$ for all $x, y \in N$, then we say that $\mathcal{P}'$ *refines* $\mathcal{P}$. All partitions of a finite set can be enumerated in $O(1)$ amortized time per partition [29].

Let G be an undirected graph, T be a subset of its vertices called *terminals*, and $\mathcal{P}$ be a partition of T. A subset of edges X in G is a *$\mathcal{P}$-cut* if no component of $G - X$ contains terminals from more than one subset of $\mathcal{P}$. Consider the following graph separation problem:

Partition Cut	
Instance:	An undirected graph G with positive integer edge weights $w_G : E(G) \to \mathbb{N}^+$, a set of terminals $T \subseteq V(G)$, a partition $\mathcal{P}$ of T, and an integer k.
Parameter:	k.
Question:	Is there a $\mathcal{P}$-cut in G of total weight at most k?

We may view this problem in the light of multiway cuts.

(Edge) Multiway Cut	
Instance:	An undirected graph G with positive integer edge weights $w_G : E(G) \to \mathbb{N}^+$, a set of vertices (terminals) $T \subseteq V(G)$ and an integer k.
Parameter:	k.
Question:	Is there a set of edges $X \subseteq E(G)$ of total weight at most k such that every component of $G - X$ contains at most one vertex from T?

One way to formulate the goal of the solution X in Partition Cut is to ensure the partition of terminals into connected components of $G - X$ refines $\mathcal{P}$. Thus, Multiway Cut is a special case of this problem where every subset

of $\mathcal{P}$ is a singleton i.e. X needs to separate all terminals. In fact, we can reduce from Partition Cut to Multiway Cut and thus show that Partition Cut is in FPT.

Proposition 10 (Cygan et al. [16])**.** *Multiway Cut is solvable in $O^*(2^k)$ time. If a solution exists, then the algorithm computes it in this time.*

Lemma 11. *Partition Cut is solvable in $O^*(2^k)$ time. If a solution exists, then the algorithm computes it in this time.*

Proof. Let $(G, w_G, T, \mathcal{P}, k)$ be an instance of Partition Cut, where $\mathcal{P} = \{B_1, \ldots, B_m\}$. For every $i \in [m]$, introduce a superterminal vertex s_i and connect all terminals in B_i to s_i with edges of weight $k+1$. Let the resulting graph be G', the weight function $w_{G'}$, and the set of superterminals be $S = \{s_1, \ldots, s_m\}$. Then the instance of Multiway Cut is $(G', w_{G'}, S, k)$. Correctness of the reduction follows by noting that a cut in G' is a solution only if it partitions superterminals into distinct connected components. Since edges connecting any $s \in S$ to any $t \in T$ have weight $k+1$, they cannot be included in the solution. Hence, terminals are partitioned according to $\mathcal{P}$ as well. The reduction runs in polynomial time and the parameter is unchanged so we obtain the desired running time via Proposition 10. □

3.2 Pair Partition Cut

For Min-2-Lin over more general rings than fields, our reduction leads to a more general graph separation problem. Given a graph G with a set of terminals $T \subseteq V(G)$, a *(disjunctive) pair cut request* is a tuple $(\{s,u\},\{t,v\})$ where $s,t \in T$ and $u,v \in V(G)$. A cut $X \subseteq E(G)$ *fulfils* $(\{s,u\},\{t,v\})$ if $G-X$ does not contain an su-path or $G-X$ does not contain a tv-path.

Pair Partition Cut	
Instance:	An undirected graph G with positive integer edge weights $w_G : E(G) \to \mathbb{N}^+$, a set of vertices (terminals) $T \subseteq V(G)$, a partition $\mathcal{P}$ of T, a set $\mathcal{F}$ of pair cut requests, and an integer k.
Parameter:	k.
Question:	Is there a $\mathcal{P}$-cut $X \subseteq E(G)$ of total weight at most k that fulfils every pair cut request in $\mathcal{F}$?

We prove that this problem is in FPT by casting it into the constraint satisfaction framework. A constraint satisfaction problem (CSP) is defined by a constraint language Γ, which is a set of relation over a domain D. A relation of arity r is a subset of D^r. An instance $I = (V, C)$ of CSP(Γ) is a set of variables V and a set of constraints C of the form $R(v_1, \ldots, v_r)$, where $R \in \Gamma$ is a relation of arity r. The instance I is consistent if it admits an assignment $\varphi : V(C) \to D$ that satisfies every constraint in C i.e. $(\varphi(v_1), \ldots, \varphi(v_r)) \in R$ holds for all constraints. In the parameterized version MinCSP(Γ) the input is an instance $I = (V, C)$ of CSP(Γ) together with a weight function $w_C : C \to \mathbb{N}^+$ and the parameter $k \in \mathbb{N}^+$, and the goal is to check whether there is a subset $X \subseteq C$ of equations with total weight at most k such that $(V, C \setminus X)$ is consistent.

For the intuition behind the reduction, consider an instance of Pair Partition Cut with a solution X. Assume without loss of generality that G is connected. Since X contains at most k edges, removing X splits G into at most $k+1$ connected components. Enumerate connected components of $G-X$ with integers from 0 to k so that terminals from subset B_i of $\mathcal{P}$ are in the ith connected component. This is possible since X is a $\mathcal{P}$-cut. We define a function $\phi : V(G) \to \{0, \ldots, k\}$ such that $\phi(x) = i$ whenever x belongs to ith component of $G-X$. Then for every pair cut request

$(\{s,u\},\{t,v\})$ with $s \in B_i$ and $t \in B_j$, we have $\phi(u) \neq i$ or $\phi(v) \neq j$. This reasoning suggests that all requirements of Pair Partition Cut can be encoded using the following constraint language Γ_k with domain $\{0,\ldots,k\}$ and relations:

- unary relations $(x = i)$ for all $0 \leq i \leq k$,
- binary equality relation $(x = y)$, and
- binary relation $(x \neq i) \vee (y \neq j)$ for all $1 \leq i, j \leq k$.

To solve CSP(Γ_k), we define another constraint language Γ'_k with domain $\{0,1\}$ and relations:

- $(x = 0)$, $(x = 1)$,
- $R_k(x_1, y_1, \ldots, x_k, y_k) \equiv \bigwedge_{1\leq i\leq k}(x_i = y_i) \wedge \bigwedge_{1\leq i<j\leq k}(\neg x_i \vee \neg x_j)$,
- $(\neg x \vee \neg y)$.

Kim et al. [39] prove a dichotomy characterising MinCSP(Γ) as fpt or W[1]-hard for every finite Boolean language Γ. We note that the problem MinCSP(Γ'_k) is fpt thanks to their result.

Lemma 12. *The problem MinCSP(Γ'_k) is in* FPT *when parameterized by $\ell = k + c$ where c is total solution cost.*

Proof. Let $R \subseteq \{0,1\}^r$ be a Boolean relation. The relation R is *bijunctive* if it can be defined as the set of solutions to a 2-CNF formula F over the variables $X = \{x_1,\ldots,x_r\}$, i.e. $(a_1,\ldots,a_r) \in R$ if and only if the assignment $x_i \mapsto a_i$, $1 \leq i \leq r$, satisfies F. Let F be a 2-CNF formula defining $R(x_1,\ldots,x_r)$ this way. The *Gaifman graph* of F is the graph on vertex set X with an edge $\{x_i, x_j\}$ for $x_i, x_j \in X$ if and only if F contains a 2-clause on the variables x_i and x_j. We say that R is $2K_2$*-free* if R can be defined via a 2-CNF formula whose Gaifman graph does not contain $2K_2$ as an induced subgraph. Kim et al. [39, Theorem 1.2] showed that if Γ is a Boolean language where every relation is bijunctive and $2K_2$-free, then MinCSP(Γ) is fpt parameterized by the solution cost c. Furthermore, in the extended preprint version it is shown that this remains true even if the maximum arity of a constraint is taken as a second parameter instead of a constant [38, Theorem 3.1]. Every relation $R \in \Gamma'_d$ is definable by a 2-CNF formula with a $2K_2$-free Gaifman graph, and the result follows. □

We present a two-step reduction from Pair Partition Cut to MinCSP(Γ_k) to MinCSP(Γ'_k).

Theorem 13. *Pair Partition Cut is in* FPT.

Proof. First, we spell out the reduction from Pair Partition Cut to MinCSP(Γ_k). Given an instance $(G, w_G, T, \mathcal{P}, \mathcal{F}, k)$ of Pair Partition Cut, we construct an instance $((V, C), w, k)$ of MinCSP(Γ_k). Let $V = V(G)$ denote the set of variables. We define the set of constraints C and the weight function w as follows. Enumerate subsets in $\mathcal{P}$ as $B_1,\ldots,B_m$ and for every subset B_i, add the constraints $(t = i)$ for all $t \in B_i$ of weight $k+1$. For every edge $\{u,v\} \in E(G)$, add the constraint $(u = v)$ of weight $w_G(\{u,v\})$. Finally, for every pair cut request $(\{u,s\},\{v,t\})$ in $\mathcal{F}$ with $s \in B_i$ and $t \in B_j$, add the constraint $(u \neq i) \vee (v \neq j)$ of weight $k+1$. Clearly, the reduction can be carried out in polynomial time. A solution X to $((V,C),w,k)$ may only contain equality equations because every other constraint is assigned weight $k+1$. It is easy to see that $\{\{u,v\} \in E(G) \mid (u = v) \in X\}$ is a $\mathcal{P}$-cut in G that fulfils $\mathcal{F}$. To obtain a solution to $((V,C),w,k)$ from a solution to the Pair Partition Cut instance, one may pick the equality constraints corresponding to the edges of the cut.

We continue by reducing MinCSP(Γ_k) to MinCSP(Γ'_k). Given an instance $I = ((V,C),w,k)$ of the former problem, we produce an equivalent instance $I' = ((V',C'),w',k)$ of the latter, while keeping the parameter unchanged. To this end, introduce variables $v^{(i)}$ for every $v \in V$ and $i \in \{1,\ldots,k\}$. Intuitively, setting $v^{(i)} = 1$ corresponds to assigning

value i to v, while setting $v^{(i)} = 0$ for all $i \in \{1, \ldots, k\}$ corresponds to assigning 0 to v. Every constraint c in C is replaced by constraints in C' of the same weight as follows:

(1) if c is $t = i$ for $i \in \{1, \ldots, k\}$, then add $t^{(i)} = 1$ to C',
(2) if c is $t = 0$, then add $t^{(i)} = 0$ for all $i \in \{1, \ldots, k\}$ to C', each of weight $w(c)$,
(3) if c is $(u = v)$, then add $R_k(u^{(1)}, v^{(1)}, \ldots, u^{(k)}, v^{(k)})$ to C', and
(4) if c is $(s \neq i) \vee (t \neq j)$, then add $(\neg s^{(i)} \vee \neg t^{(j)})$ to C'.

This concludes the reduction.

Suppose ϕ is an assignment to (V, C). Define ϕ' by letting $\phi'(v^{(i)}) = 1$ if $\phi(v) = i$ for some $i \in \{1, \ldots, k\}$, and $\phi(v^{(i)}) = 0$ otherwise. By construction, ϕ and ϕ' break constraints of the same total weight. Hence, if the set of constraints unsatisfied by ϕ is a solution to I, then the set of constraints unsatisfied by ϕ' is a solution to I'. The same argument works in the opposite direction: given an assignment ρ' to (V', C'), define assignment ρ to (V, C) by letting $\rho(v) = i$ if $\rho'(v^{(i)}) = 1$ for some $i \in \{1, \ldots, k\}$, and $\rho(v) = 0$ otherwise. Constraints of the type $(\neg v^{(i)} \vee \neg v^{(j)})$ ensure that ρ is well-defined. Moreover, the total weight of constraints unsatisfied by ρ and ρ' is the same. Thus, the reduction is correct and the theorem follows. □

4 ALGORITHM FOR Min-2-Lin

Our goal with this section is to present an fpt-algorithm for Min-2-Lin that is applicable to a broad class of rings. We start by reviewing basic definitions and facts about rings and integral domains in Section 4.1. We introduce the *Helly dimension* in Section 4.2 and we develop a polynomial-time algorithm for 2-Lin($\mathbb{D}$) when $\mathbb{D}$ is a commutative integral domain with finite Helly dimension (and $\mathbb{D}$ is represented in a way that satisfies some mild technical assumptions). We note that polynomial-time algorithms for r-Lin($\cdot$) are known for arbitrary $r \in \mathbb{N}$ and arbitrary finite rings [2, Section 6], the ring of integers [34] or the ring of univariate polynomials over $\mathbb{Q}$ [33]. However, we are unaware of more general results of this kind, even when $r = 2$. We continue by presenting the details of our algorithm for Min-2-Lin. This algorithm is applicable to the class H_2 of commutative domains with Helly dimension 2 (and satisfying the same technical assumptions as required by our 2-Lin($\cdot$) algorithm). The next three sections follow the common steps of compression, cleaning, and cutting: we simplify the problem by applying iterative compression in Section 4.3, then simplify it even further by applying the important balanced subgraph machinery in Section 4.4, and finally reduce the resulting problem to Pair Partition Cut in Section 4.5, giving an overview of the whole algorithm. We prove correctness of the algorithm and analyze its time complexity in Section 4.6. Finally, we take a closer look at the rings in H_2 in Section 4.7, where we show that reasonably represented *Prüfer* domains are members of H_2. This, in turn, shows that a wide range of interesting rings are to be found in H_2.

4.1 Rings and Integral Domains

A *ring* is an Abelian group (whose operation is called *addition*), with a second binary operation called *multiplication* that is associative, is distributive over the addition operation, and has an identity element. We will exclusively consider *commutative rings* where multiplication is a commutative operator. Let $\mathbb{D} = (D, +, \cdot)$ denote such a ring. We let 0 denote the additive identity element and 1 the multiplicative identity element. An element $d \in D$ is a *zero divisor* if $d \neq 0$ and there exists an element $0 \neq d' \in D$ such that $dd' = 0$. The ring $\mathbb{D}$ is an *integral domain* (or simply a *domain*) if it contains at least two elements and it contains no zero divisors.

Let us now fix a particular representation of the elements in $\mathbb{D}$. Given an element $d \in D$, let $\|d\|$ denote the number of bits required to represent d. We say that $\mathbb{D}$ is *effective* if there are polynomial-time algorithms for the following operations for any pair $a, b \in D$:

- computing the sum $a + b$,
- computing the product ab,
- computing additive inverses $-a$,
- checking if b divides a, i.e. checking whether there exists c such that $a = bc$, and if so, computing c,
- checking if a is a unit, i.e. admits a multiplicative inverse, and if so, computing the inverse a^{-1}, and

there is a polynomial function p such that the result of evaluating any arithmetic expression with $O(n)$ operations, with arbitrary constants $d_1, \ldots, d_n \in \mathbb{D}$, operations + and $\cdot$, and brackets has bit-size at most $p(\|d_1\| + \cdots + \|d_n\|)$. The final requirement is natural since otherwise we cannot compute (or even write down) satisfying assignments to simple consistent instances of 2-Lin($\mathbb{D}$) like

$$\{x_1 = d_1 x_2, x_2 = d_2 x_3, \ldots, x_{n-1} = d_n x_n\} \cup \{x_n = 1\}$$

in polynomial time, we cannot perform efficient Gaussian elimination etc. In many cases, p is the identity polynomial i.e. representing the sum or product of elements requires at most as many bits as representing them individually.

Consider a binary linear equation $ax + by = c$ with $a, b, c \in \mathbb{D}$. The lack of zero divisors in $\mathbb{D}$ is equivalent to the following: fixing the value of one variable (say, x) leaves at most one possible value for the other (say, y). To see this, assume without loss of generality that $b \neq 0$ and that the equation is satisfied by two assignments $(x, y) \mapsto (e, f)$ and $(x, y) \mapsto (e, f')$ where $f \neq f'$. This implies that $(ae + bf) - (ae + bf') = 0$ so $b(f - f') = 0$ and b is a zero divisor. We exploit this property to make basic observations towards solving 2-Lin($\mathbb{D}$). If the value of one variable in a connected instance is fixed, this propagates to all variables; this propagation will either lead to a satisfying assignment for the whole instance or to a contradiction (either implying two different values for a variable or falsifying an equation, e.g. an equation $2x = y$ over $\mathbb{Z}$ with propagation implying $y = 1$).

Let S be an instance of the problem, and recall that the primal graph of S is an undirected graph with a vertex for each variable and an edge for each equation. We may (without loss of generality) assume that the graph does not have self-loops by introducing a zero variable z_0 and an auxiliary variable z'_0, adding equations $z'_0 + z_0 = 0$ and $z'_0 - z_0 = 0$, and replacing single-variable equations $ax = b$ with $ax - z_0 = b$. Thus, we assume that the zero variable z_0 is available in every instance of 2-Lin($\mathbb{D}$), and in Min-2-Lin($\mathbb{D}$) equations $z'_0 + z_0 = 0$ and $z'_0 - z_0 = 0$ are given weight $k + 1$. We use graph-related terminology (such as connectedness, paths, cycles etc.) to describe the structure of S while having the primal graph in mind. First, consider an instance P of 2-Lin($\mathbb{D}$) whose primal graph is a path connecting variables x and y. If $|V(P)| = 2$, then P contains a single equation. Otherwise, we can eliminate intermediate variables to obtain an equation over x and y by recursively picking any two equations in P, say, $ap_1 + bp_2 = c$ and $a'p_2 + b'p_3 = c'$, and taking their linear combination $a'(ap_1 + bp_2) - b(a'p_2 + b'p_3) = a'c - bc'$, which simplifies to $(a'a)p_1 - (b'b)p_3 = a'c - bc'$. We say that P *implies* the final equation e_P over x and y obtained after eliminating all intermediate variables. For example, suppose P is a path instance of 2-Lin($\mathbb{Z}$) with two equations, $x - 2z = 2$ and $z - y = 1$, then the equation $x - 2y = 4$ implied by P is obtained by eliminating z. Clearly, elimination is safe, i.e. any assignment that satisfies P also satisfies e_P. Moreover, the computation of e_P boils down to evaluating arithmetic expressions with $O(|V(P)|)$ constants. Since we are working over effective rings, we arrive at the following conclusion.

Observation 14. *Let $\mathbb{D}$ be an effective integral domain. For every instance P of 2-Lin($\mathbb{D}$) whose primal graph is a path connecting variables x and y, there exists an equation e_P over x and y computable in polynomial time such that any assignment that satisfies P also satisfies e_P.*

We say that an instance S of 2-Lin($\mathbb{D}$) is *flexible* if for every pair of variables $x, y \in V(S)$, every xy-path in S implies *equivalent* equations on x and y, i.e. equations with the same set of satisfying assignments. Otherwise, we say that S is *rigid*. If S is flexible, we write $e_{xy}(S)$ to denote the equation implied by the xy-paths in S. For example, acyclic instances are flexible. An instance of 2-Lin($\mathbb{D}$) is a *star instance* if there is exactly one variable shared by all equations. For a flexible instance S and any $x \in V(S)$, we define the corresponding star instance $\text{star}(S, x) = \{e_{xy}(S) \mid y \in V(S) \setminus \{x\}\}$ of 2-Lin($\mathbb{D}$).

Lemma 15. *Let S be a connected, flexible instance of 2-Lin($\mathbb{D}$). For any $x \in V(S)$, instances S and $\text{star}(S, x)$ have the same set of satisfying assignments.*

Proof. By Observation 14, an assignment that satisfies S also satisfies the equations implied by the paths in S. Consequently, it satisfies $\text{star}(S, x)$ for any x. Now, suppose φ is a satisfying assignment to $\text{star}(S, x)$ and consider an equation $e \in S$ over variables y and z. It suffices to show that φ satisfies e. Clearly, this holds if $y = x$ or $z = x$ since then $e \in \text{star}(S, x)$. Otherwise, since S is connected, in the primal graph we have a path that contains x, y and z and uses the edge corresponding to the equation e. By the construction of equations implied by the paths, the equation e can be written as a linear combination of $e_{xy}(S)$ and $e_{xz}(S)$. Since these two equations are present in $\text{star}(S, x)$, φ satisfies them, and hence also satisfies e. □

Thus, checking consistency of a flexible instance is equivalent to checking consistency of a corresponding star instance. We return to the question of solving star instances in the next section, while focusing on rigid instances in the next lemma.

Lemma 16. *There is a polynomial-time algorithm that takes a connected instance S of 2-Lin($\mathbb{D}$) as input, checks whether it is rigid and consistent, and if so, computes a satisfying assignment.*

Proof. Let S be an instance of 2-Lin($\mathbb{D}$). Compute a spanning tree T of S. To check whether S is flexible or not, we consider equations in $S - T$. Assume that $e \in S - T$ is $a_1x + b_1y = c_1$ and let $e_{xy}(T)$ equal $a_2x + b_2y = c_2$. To check whether these two equations are equivalent, multiply the first one with a_2, the second one with a_1, compute their difference and obtain $(b_1a_2 - a_1b_2)y = c_1a_2 - a_1c_2$. For conciseness, let $A = b_1a_2 - a_1b_2$, $B = c_1a_2 - a_1c_2$. There are four cases to consider:

- If $A = 0$ and $B \neq 0$, then $Ay = B$ is inconsistent.
- If $A = 0$ and $B = 0$, then $Ay = B$ is satisfied by assigning any value to y.
- If $A \neq 0$ and A does not divide B, then $Ay = B$ is inconsistent.
- If $A \neq 0$ and A divides B, then $Ay = B$ is only satisfied by setting y to B/A since $\mathbb{D}$ is an integral domain.

If $Ay = B$ is inconsistent, then no assignment can satisfy both e and $e_{xy}(T)$, hence S is inconsistent. If $A = B = 0$, we claim that e and $e_{xy}(T)$ are equivalent. Indeed, if $\varphi : \{x, y\} \to \mathbb{D}$ satisfies e, then

$$\begin{aligned} a_1\varphi(x) + b_1\varphi(y) &= c_1 && \iff \\ a_1a_2\varphi(x) + b_1a_2\varphi(y) &= c_1a_2 && \iff \\ a_2a_1\varphi(x) + b_2a_1\varphi(y) &= c_2a_1 && \iff \\ a_2\varphi(x) + b_2\varphi(y) &= c_2. \end{aligned}$$

Note that we have used $b_1a_2 = a_1b_2$ and $c_1a_2 = a_1c_2$ for the second implication, and $a_1 \neq 0$ for the third implication, yielding $a_1a_2\varphi(x) + a_1b_2\varphi(y) = a_1c_2$. Since $a_1 \neq 0$, we can divide both sides by a_1 and obtain that φ satisfies $e_{xy}(T)$. Thus, if $A = B = 0$, then equations e and $e_{xy}(T)$ are equivalent, and we proceed to the next equation in $S - T$. Finally, if $Ay = B$ has only one satisfying assignment (namely $y \mapsto B/A$), we assign this value to y and propagate to the rest of the instance, and then check whether the obtained assignment satisfies every equation. If the propagation does not occur at any step, we conclude that the instance is flexible. □

4.2 Helly Dimension

We will now introduce our notion of Helly dimension for commutative rings. We begin by giving a brief background and we refer the reader to the survey by Bárány and Kalai [3] for a more thorough introduction. A *Helly* family of order k is a pair (X, S) where X is a set and S is a collection of subsets of X, such that, for every finite $S' \subseteq S$ with $\bigcap_{T \in S'} T = \emptyset$, one can find $T' \subseteq T$ such that $\bigcap_{T'' \in T'} T'' = \emptyset$ and $|T'| \leq k$. Helly's theorem on convex sets is an important example: it states that convex sets in Euclidean space of dimension n are a Helly family of order $n + 1$. *Helly dimension* is a value that is often associated with a metric space $\mathcal{M}$: this dimension is one less than the Helly number of the family of metric balls in $\mathcal{M}$; clearly, Helly's theorem implies that the Helly dimension of a Euclidean space equals its dimension as a real vector space. Naturally, Helly dimensions of various kinds have also been applied to many other mathematical objects.

We continue by introducing some terminology related to rings. An *ideal* in a commutative ring $\mathbb{D}$ is a set $I \subseteq D$ such that (1) $(I, +)$ is a subgroup of $(D, +)$ and (2) for every $d \in D$ and every $x \in I$, the product dx is in I. If $\mathcal{I}$ is a (possibly infinite) set of ideals in $\mathbb{D}$, then $\bigcap \mathcal{I}$ is an ideal in $\mathbb{D}$, too. Thus, given a set $D' \subseteq D$, we let (D') denote the smallest ideal that contains the elements in D'—such an ideal always exists. Similarly, the sum $\sum \mathcal{I}$ consists of all finite sums of elements taken from the members of $\mathcal{I}$. The sum of ideals is again an ideal. Note that for arbitrary $a, b \in D$, we say that b *divides* a if and only if $a \in (b)$.

Let G denote an Abelian group with a subgroup H. The *cosets* of H are $H + g = \{h + g \mid h \in H\}$ where $g \in G$. Since ideals in $\mathbb{D}$ are subgroups of the Abelian group $(D, +)$, it makes sense to speak about the cosets of an ideal. A *principal ideal* I in $\mathbb{D}$ is an ideal that is generated by a single element, i.e. $I = (d) = \{dr \mid r \in \mathbb{D}\}$ for some $d \in D$. If $\mathbb{D}$ is a field such as $\mathbb{Q}$, then $\mathbb{D}$ has only two ideals (0), (1) and both are principal. The principal ideals in $\mathbb{Z}$ are of the form $(n) = n\mathbb{Z}$. Define the *Helly dimension* of $\mathbb{D}$ as the minimum number $\kappa(\mathbb{D})$ in $\mathbb{N} \cup \{\infty\}$ with the following property: if a family of principal ideal cosets in $\mathbb{D}$ has empty intersection, then it contains a subfamily of $\leq \kappa(\mathbb{D})$ cosets that have empty intersection. Put differently, if F is a family of principal ideal cosets in $\mathbb{D}$, and the cosets of every subfamily of F of size at most $\kappa(\mathbb{D})$ intersect, then all cosets in F intersect as well. If $\mathbb{D}$ is finite, then computing $\kappa(\mathbb{D})$ is clearly decidable by exhaustive enumeration of all triples of principal ideal cosets. If $\mathbb{D}$ is infinite, then it is currently unknown how to compute $\kappa(\mathbb{D})$. We note that a similar notion of Helly dimension has been studied in the context of groups [20, 21] but

most common dimension measures for rings (such as the Krull dimension or the weak global dimension) do not seem to be closely related to the Helly dimension.

Let us consider the field $\mathbb{Q}$ of rationals. Since $\mathbb{Q}$ only contains the two ideals (0) and (1), it is easy to verify that the intersection of principal ideal cosets $C_1, C_2, \dots$ is empty if and only if there exist distinct indices i, j such that $C_i = (0)+a$, $C_j = (0) + b$, and $a \neq b$. It follows that $\kappa(\mathbb{Q}) = 2$. Consider the ring $\mathbb{Z}$ instead and assume that $\mathcal{C} = \{C_1, C_2, \dots\}$ is a family of principal ideal cosets. Each coset C_i equals $(n_i) + a_i$ with $n_i, a_i \in \mathbb{Z}$. Each C_i can be viewed as a congruence $x = a_i \mod n_i$, and the Chinese Remainder Theorem implies that $\bigcap \mathcal{C} \neq \emptyset$ if and only if the greatest common divisor of n_i and n_j divides $a_i - a_j$ for all distinct i and j. Thus, if $\bigcap \mathcal{C} = \emptyset$, then there exists $C_i, C_j \in \mathcal{C}$ such that $C_i \cap C_j = \emptyset$ so $\kappa(\mathbb{Z}) = 2$.

We continue by demonstrating that even quite simple rings may have strictly higher Helly dimension than 2. Let $\mathbb{Z}[X]$ denote the ring whose elements are univariate polynomials with integer coefficients. This ring is a commutative domain. Consider the following equations over $\mathbb{Z}[X]$.

$$v = 2 \cdot w_1$$
$$v = X \cdot w_2$$
$$v = (2 + X) \cdot u + 2$$

One may observe that this system of equations is indeed a star instance. The first two equations have the zero solution. Equations 1 and 3 have a solution $v = 2, w_1 = 1, u = 0$. Equations 2 and 3 have a solution $v = -X$, $w_2 = -1$, $u = -1$. We show that the system as a whole has no solution. If there is a solution, then v is a multiple of $2X$, i.e. it has a zero constant-coefficient and all coefficients even. Then the constant term of u is -1. But this gives a contribution of a single negative X-term. There is no value a such that $(2 + X)(-1 + aX)$ has an even X-coefficient. It follows that the Helly dimension of $\mathbb{Z}[X]$ is at least 3.

We let the set H_m, $m \in \mathbb{N} \cup \{\infty\}$, consist of the rings $\mathbb{D}$ that have the following four properties.

P1. $\mathbb{D}$ is a commutative integral domain,
P2. $\mathbb{D}$ has Helly dimension at most $m \in \mathbb{N}$,
P3. $\mathbb{D}$ is effective, and
P4. there is a polynomial-time algorithm checking whether the intersection of at most m principal ideal cosets in $\mathbb{D}$ is nonempty.

We show that these conditions allow us to solve 2-Lin($\mathbb{D}$) in polynomial time. We begin by exhibiting a connection between Helly dimension and star instances.

Lemma 17. *Let $\mathbb{D}$ be an integral domain. Then the Helly dimension of $\mathbb{D}$ is the minimum value $\kappa(\mathbb{D}) \in \mathbb{N} \cup \{\infty\}$ with the following property. For any star instance S of 2-Lin($\mathbb{D}$), if every subinstance of S of size $\leq \kappa(\mathbb{D})$ is consistent, then S is consistent.*

Proof. Let $s(\mathbb{D})$ be the value described in the statement of the lemma, i.e. the minimum value in $\mathbb{N} \cup \{\infty\}$ with the property: for any star instance S of 2-Lin($\mathbb{D}$), if every subinstance of S of size $\leq \kappa(\mathbb{D})$ is consistent, then S is consistent. We will show that $s(\mathbb{D}) = \kappa(\mathbb{D})$.

Towards contradiction, assume $s(\mathbb{D}) < \kappa(\mathbb{D})$. Then there exists a family of cosets $\mathcal{C} = \{(b_i) + c_i : i \in [m]\}$ such that every subset of $\mathcal{C}$ of size $\leq s(\mathbb{D})$ has an intersection, but $\bigcap \mathcal{C} = \emptyset$. Construct a star instance $S = \{x = b_i y_i + c_i : i \in [m]\}$. Observe that an equation $x = b_i y_i + c_i$ is consistent with a value $\varphi(x)$ for x if and only if $\varphi(x) \in (b_i) + c_i$. Thus, $S' \subseteq S$

is consistent if the corresponding subset of cosets has an intersection. We conclude that $S' \subseteq S$ is consistent whenever $|S'| \leq s(\mathbb{D})$. By definition of $s(\mathbb{D})$, this implies that S is consistent, and contradicts that $\bigcap C = \emptyset$.

Now assume $s(\mathbb{D}) > \kappa(\mathbb{D})$. Then there exists an inconsistent star instance $S = \{a_i x = b_i y_i + c_i : i \in [m]\}$ such that every subinstance of S of size $\leq \kappa(\mathbb{D})$ is consistent. Without loss of generality, assume that $a_i \neq 0$ for all $i \in [m]$. Define a star instance T as follows. Let $a' = a_1 \cdot \cdots \cdot a_m$, $b'_i = (a'/a_i) \cdot b_i$ and $c'_i = (a'/a_i) \cdot c_i$. The equations of T are $x' = a'x$ and $x' = b'_i y_i + c'_i$ for all $i \in [m]$.

Claim 17.1. *If T is consistent, then S is consistent.*

Proof of claim: Suppose assignment ϕ satisfies T. Consider equation $x' = b'_i y_i + c'_i$. Since $\phi(x') = a'\phi(x)$, we have $a'\phi(x) = b'_i \phi(y_i) + c'_i$ for all $i \in [m]$. Recall that $a_i \neq 0$ for all $i \in [m]$, hence $a'/a_i \neq 0$. Divide both sides of the equation by a'/a_i, and obtain $a_i\phi(x) = b_i\phi(y_i) + c_i$ for all $i \in [m]$. ⋄

Since all coefficients in front of x in T equal 1, T is consistent if and only if cosets $C = \{(a'), (b'_i) + c'_i : i \in [m]\}$ have empty intersection. Assuming S is inconsistent, the claim above implies that $\bigcap C = \emptyset$. By definition of $\kappa(\mathbb{D})$, there exists a subset C' of C of size $\leq \kappa(\mathbb{D})$ with empty intersection. Without loss of generality, assume that C' is a proper subset of $\{(a'), (b'_i) + c'_i : i \in [\kappa(\mathbb{D})]\}$. Let $S' = \{a_i x = b_i y_i + c_i : i \in [\kappa(\mathbb{D})]\}$ and note that $|S'| \leq \kappa(\mathbb{D})$ implies that S' is consistent by assumption. Let φ be an assignment satisfying S', and consider value $d = a'\varphi(x)$. Clearly, $d \in (a')$. Moreover, we claim that $d \in (b'_i) + c'_i$ for all $i \in [\kappa(\mathbb{D})]$. To this end, consider an equation $a_i x = b_i y_i + c_i$ satisfied by φ, and multiply both sides by a'/a_i. We obtain $a'\varphi(x) = b'_i\varphi(y_i) + c'_i$ and $a'\varphi(x) \in (b'_i) + c'_i$ for all $i \in [\kappa(\mathbb{D})]$. Thus, $d \in \bigcap C$, contradicting that S is inconsistent. □

Theorem 18. *Let $\mathbb{D}$ be a ring in H_m where $m \in \mathbb{N}$. Then, 2-Lin($\mathbb{D}$) admits a polynomial-time algorithm.*

Proof. Let S be an instance of 2-Lin($\mathbb{D}$). Assume without loss of generality that S is connected. By Lemma 16, if S is rigid, then we can decide whether it is consistent. Otherwise, we know that S is a flexible instance. Apply Lemma 15 to obtain the corresponding star instance $\mathrm{star}(S)$. By the proof of Lemma 17, we can rewrite S as an equivalent star instance T with all coefficients before the common variable x being 1. Checking whether this instance is consistent is equivalent to checking whether every subset of corresponding cosets of size $\leq \kappa(D)$ has an intersection. In total, this requires $n^{O(\kappa(\mathbb{D}))}$ calls to the polynomial-time coset-intersection algorithm. □

In Theorem 18 we showed that, under reasonable technical assumptions, 2-Lin($\mathbb{D}$) can be solved in polynomial time whenever $\mathbb{D}$ is a commutative integral domain with finite Helly dimension. For the optimisation problem Min-2-Lin($\mathbb{D}$), we will additionally require that the Helly dimension of $\mathbb{D}$ is at most 2. The intuition behind the bound on Helly dimension comes from Lemma 15 which shows the reduction from a flexible instance S to an equivalent star instance $\mathrm{star}(S)$. A pair of equations in $\mathrm{star}(S)$ corresponds to a path in S, so we obtain the following consequence.

Observation 19. *If $\kappa(\mathbb{D}) = 2$, then a flexible instance S of Min-2-Lin($\mathbb{D}$) is consistent if and only if every path in S is consistent.*

Using this observation, Min-2-Lin($\mathbb{D}$) on flexible instances can be solved by a simple reduction to Multicut [6, 47] that creates an instance with the primal graph of S as input, computes a spanning tree T of S and adds a cut request $\{x, y\}$ for every pair of variables $x, y \in V(S)$ such that $e_{x,y}(T)$ is inconsistent. The algorithm for general instances requires more work which is covered in the following sections.

4.3 Iterative Compression

We let $\mathbb{D}$ denote an arbitrary member of H_2 in this and the following sections. We reduce Min-2-Lin($\mathbb{D}$) to a simpler problem by combining a trick called *homogenisation* with *iterative compression*. An equation $ax + by = c$ is *homogeneous* if $c = 0$ and an instance of 2-Lin($\mathbb{D}$) is homogeneous if every equation in the instance is homogeneous. Note that any such system is consistent since it is satisfied by the all-zero assignment. We show that by applying an invertible affine transformation to the solution space, we can turn every consistent instance S of 2-Lin($\mathbb{D}$) into a homogeneous system with the same primal graph. Iterative compression is a common opening step of fpt-algorithms, see e.g. [14, Chapter 4] for a thorough treatment. The basic idea is to solve the deletion problem by iteratively building up the instance one equation at a time. While the instance size is at most k, deleting all equations is an acceptable solution. For larger instance sizes, we can assume access to a deletion set of size $k + 1$, and design a *compression routine* that takes the instance and the oversized solution as input, and either produces a smaller solution or concludes that no smaller solution exists. If the routine runs in fpt-time, then the whole algorithm also runs in fpt-time.

Let S be an instance of 2-Lin($\mathbb{D}$). Define a mapping Φ that acts on every variable $x \in V(S)$ by setting $x \mapsto a_x x' + b_x$ for some $a_x, b_x \in \mathbb{D}$. We refer to Φ as a *variable substitution for* S, and write $\Phi(S)$ to denote the instance of 2-Lin($\mathbb{D}$) obtained by substituting every variable x with $a_x x' + b_x$. A variable substitution is *homogenising* if $\Phi(S)$ is homogeneous.

Lemma 20. *Every consistent instance S of 2-Lin($\mathbb{D}$) admits a homogenising variable substitution Φ such that S and $\Phi(S)$ have the same primal graph, and there is a bijection between satisfying assignments to S and to $\Phi(S)$. Moreover, if we can compute a solution to an instance of 2-Lin($\mathbb{D}$) in polynomial time, then Φ can be computed in polynomial time as well.*

Proof. Let S be an instance of 2-Lin($\mathbb{D}$) satisfied by assignment φ. Define Φ as $x \mapsto x' + \varphi(x)$ for all $x \in V(S)$. Note that Φ is reversible by subtracting $\varphi(x)$. Consider an equation $ax + by = c$ in S. Note that $a\varphi(x) + b\varphi(y) = c$ since φ satisfies the equation. Its counterpart in $\Phi(S)$ is $a(x'+\varphi(x))+b(y'+\varphi(y)) = c$, which simplifies to $ax'+by' = 0$. The right hand side in the obtained equation is 0. Thus, the variable substitution Φ is homogenising, and computing a satisfying assignment to S is sufficient to obtain Φ. Moreover, Φ does not change the primal graph, and an assignment α satisfies S if and only if $\alpha + \varphi$ satisfies $\Phi(S)$, so satisfying assignments to S and $\Phi(S)$ are in one-to-one correspondence. □

We use iterative compression to reduce from Min-2-Lin($\mathbb{D}$) to the following problem:

Disjoint Min-2-Lin($\mathbb{D}$) (DML($\mathbb{D}$))	
Instance:	An instance S of 2-Lin($\mathbb{D}$) with positive integer equation weights $w_S : S \to \mathbb{N}^+$, an inclusion-wise minimal set $X \subseteq S$ such that $S - X$ is homogeneous, and an integer k such that $w_S(X) \leq k + 1$.
Parameter:	k.
Question:	Is there a set $Z \subseteq S - X$ of weight at most k such that $S - Z$ is consistent?

Lemma 21. *If* DML($\mathbb{D}$) *is solvable in $O^*(f(k))$ time, then Min-2-Lin($\mathbb{D}$) is solvable in $O^*(2^k f(k))$ time.*

Proof. Let $I = (S, w_S, k)$ be an instance of Min-2-Lin($\mathbb{D}$). In this context it is simpler to view equations as a multiset S' where every equation $e \in S$ is present with multiplicity $w_S(e)$. Then by iterative compression, we may assume that apart from the input I, we also have access to a multiset X such that $|X| = k + 1$ and $S' - X$ is consistent.

Suppose Z is an optimal solution to S'. To reduce to DML($\mathbb{D}$), we branch on the possible intersections $Y = X \cap Z$ of the incoming solution with the optimal solution. Since there are $2^{|X|} = 2^{k+1}$ options, the branching step requires fpt-time. For every guess Y, consider the multisets $S' - Y$ and $X - Y$, and convert them into sets S_Y and X_Y, respectively,

defining the weight function w_Y so that $w_Y(e)$ for all equations e is the multiplicity of e in S_Y. Note that by definition $S_Y - X_Y$ is consistent, so we can compute and apply a homogenising variable substitution to it by Lemma 20. Finally, set the parameter to $k_Y = k - |Y|$. We obtain an instance (S_Y, w_Y, k_Y, X_Y) of DML($\mathbb{D}$). If this instance has a solution, then combining that solution with Y yields a solution to the instance I of Min-2-Lin($\mathbb{D}$). On the other hand, if there is no solution for any option Y, then by exhaustion I is a no-instance. Since we branch in 2^{k+1} directions, and in each branch we solve an instance of DML($\mathbb{D}$) with parameter bounded from above by k, we obtain the total running time of $O^*(2^k f(k))$. □

4.4 Graph Cleaning

Observation 19 provides us with a good idea of how to solve Min-2-Lin($\mathbb{D}$) restricted to acyclic and flexible instances by reduction to Multicut. To approach the general solution, we now need to consider cycles that make instances rigid.

Lemma 22. *A consistent cycle in 2-Lin($\mathbb{D}$) is flexible if and only if it admits more than one satisfying assignment, while a consistent cycle is rigid if and only if it admits a unique satisfying assignment.*

Proof. Let C be a consistent instance of 2-Lin($\mathbb{D}$) that is a cycle. By Lemma 20, we may assume without loss of generality that C is homogeneous because there is a homogenising variable substitution that preserves the number of satisfying assignments. Assuming C is homogeneous, it is satisfied by the all-zero assignment. Thus, it suffices to show that C admits a non-zero assignment if and only if it is flexible.

On the one hand, suppose that C is a flexible cycle. Pick an arbitrary equation $e \in C$. Note that $P := C \setminus \{e\}$ is a path. Let $|P| = m$ and assume that the equations on P are $a_i x_i = b_i x_{i+1}$ for $i \in \{1, \ldots, m\}$, where $a_i, b_i \in D \setminus \{0\}$ and x_i is a variable. We define the assignment φ using a particular *product construction*: set $\varphi(x_1) = b_1 \cdot b_2 \cdot \ldots \cdot b_m$, and $\varphi(x_{i+1}) = (\varphi(x_i)/b_i) \cdot a_i$ for all $i \in \{1, \ldots, m\}$. In other words, the value $\varphi(x_{i+1})$ is obtained from $\varphi(x_i)$ by replacing the factor b_i in the product by a_i. Consequently, $\varphi(x_{m+1}) = a_1 \cdot a_2 \cdot \ldots \cdot a_m$. Since all coefficients a_i, b_i are nonzero, φ is a nonzero assignment and it is easy to verify that φ satisfies P. By Observation 14, it also satisfies the implied equation e_P. Since equations e and e_P are equivalent, assignment φ satisfies e and, therefore, it satisfies $P \cup \{e\} = C$.

On the other hand, suppose that C is a rigid cycle. By definition, there are variables $x, y \in V(C)$ such that the xy-paths P_1 and P_2 forming C imply two non-equivalent equations $a_1 x = b_1 y$ and $a_2 x = b_2 y$, respectively. Multiplying the first equation by a_2 and the second by a_1, we obtain the same coefficient in front of x. Since the equations are not equivalent, the coefficients in front of y must differ i.e. $b_1 a_2 \neq a_1 b_2$. Hence, any assignment satisfying C also satisfies $(b_1 a_2 - a_1 b_2) \cdot y = 0$, which can only be satisfied by setting y to 0. The zero value propagates to all remaining variables, so C is only satisfied by the all-zero assignment. □

The following result allows us to use the graph cleaning machinery to remove rigid cycles.

Lemma 23. *Let G_S be the primal graph of a consistent instance S of 2-Lin($\mathbb{D}$) and $\mathcal{B}_S$ be the set of flexible cycles in S. Then $(G_S, \mathcal{B}_S)$ is a biased graph.*

Proof. By Lemma 20, it suffices to consider a homogeneous instance S because there is a homogenising variable substitution that does not change the primal graph; moreover, it preserves flexible cycles by Lemma 22 and the fact that it preserves the number of satisfying assignments. We want to verify that theta property holds for the family of unbalanced cycles in $(G_S, \mathcal{B}_S)$. To this end, let P, Q, R be three internally vertex-disjoint xy-paths in G_S, and assume $P \cup R$ is a rigid cycle. We claim that equations e_P and e_R are inequivalent. Then equation e_Q cannot be equivalent to both e_P and e_R. This implies that either $P \cup Q$ or $Q \cup R$ is rigid, and the lemma follows.

To prove the claim, assume towards contradiction that equations e_P and e_R are equivalent, and e_P is $ax = by$. Using the product construction from the proof of Lemma 22, define nonzero assignments φ_P and φ_R satisfying all equations in P and R, respectively. Note that by Observation 14, they also satisfy $ax = by$ i.e.

$$a \cdot \varphi_P(x) = b \cdot \varphi_P(y), \tag{1}$$

$$a \cdot \varphi_R(x) = b \cdot \varphi_R(y). \tag{2}$$

Therefore, after multiplying (1) by $\varphi_R(x)$ and (2) by $\varphi_P(x)$, we obtain:

$$a \cdot \varphi_P(x) \cdot \varphi_R(x) = b \cdot \varphi_P(y) \cdot \varphi_R(x), \tag{3}$$

$$a \cdot \varphi_R(x) \cdot \varphi_P(x) = b \cdot \varphi_R(y) \cdot \varphi_P(x). \tag{4}$$

Since the right hand sides of (3) and (4) are equal, we may equate the left hand sides and thus obtain

$$\varphi_P(y) \cdot \varphi_R(x) = \varphi_R(y) \cdot \varphi_P(x).$$

This equation allows us to define a nonzero assignment φ_{PR} that satisfies $P \cup R$ by scaling φ_P and φ_R so that they agree on the values of x and y, namely let

$$\varphi_{PR}(z) = \begin{cases} \varphi_P(z) \cdot \varphi_R(x) & \text{if } z \in P, \\ \varphi_R(z) \cdot \varphi_P(x) & \text{if } z \in R. \end{cases}$$

Since $P \cup R$ admits the nonzero satisfying assignment φ_{PR}, it is flexible by Lemma 22 and we arrive at a contradiction. □

By Lemma 21, Min-2-Lin($\mathbb{D}$) reduces to DML($\mathbb{D}$) in fpt-time. Let $I = (S, w_S, X, k)$ be an instance of the latter problem. Note that $S - X$ is consistent, so all cycles in it are either flexible or rigid. To apply graph cleaning, we construct a *rooted graph for* I as follows.

Definition 24. Let $I = (S, w_S, X, k)$ be an instance of DML($\mathbb{D}$). The *rooted graph* for I is a pair $(G_I, \mathcal{B}_I)$ defined as follows. The vertex set of G_I is the set of the variables of $S - X$ extended with a fresh *root vertex* s. The edge set contains all edges in the primal graph of $S - X$ (with the corresponding weights given by w_S) together with an edge of weight 1 from s to every vertex in $V(X)$. The set $\mathcal{B}_I \subseteq 2^{E(G_I)}$ consists of all flexible cycles in $S - X$.

Observe that the family of cycles $\mathcal{B}_I$ above admits a polynomial-time oracle, e.g. by checking for every pair of vertices whether the two paths connecting them on the cycle imply the same equation.

Lemma 25. *For every instance I of* DML($\mathbb{D}$)*, the rooted graph $(G_I, \mathcal{B}_I)$ is a biased graph.*

Proof. Consider a cycle in G_I that is outside of $\mathcal{B}_I$. Such a cycle either contains the root vertex s or is rigid in $S - X$. Adding a chordal path to a cycle of the first kind creates two cycles at least one of which also contains s. For the cycles of the second kind, invoke Lemma 23. □

The following is an immediate algorithmic consequence of Theorem 5 and Lemma 25.

Observation 26. *Let I be an instance of* DML($\mathbb{D}$) *and $(G_I, \mathcal{B}_I)$ be the rooted graph for I. Let q be a positive integer and let $\mathcal{G} := \mathcal{G}(G_I, \mathcal{B}_I, q, s)$ be the family of connected balanced subgraphs in $(G_I, \mathcal{B}_I)$ rooted in s with cost at most q. Then, in time $O^*(4^q)$ we can compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ of size at most 4^q.*

Now we characterise yes-instances of Min-2-Lin($\mathbb{D}$). For an assignment φ to an instance S of 2-Lin(D), let $V_0^\varphi = \{v \in V(S) \mid \varphi(v) = 0\}$ and $V_\emptyset^\varphi = \{v \in V(S) \mid \varphi(v) \neq 0\}$.

Lemma 27. *Let (S, w_S, X, k) be an instance of* DML($\mathbb{D}$)*, and let Z be an optimal solution to S disjoint from X. Then there exists a satisfying assignment φ to $S - Z$ such that for every connected component $K \subseteq V(S)$ of $S - (X \cup Z)$, the following hold.*

1. *Either $K \subseteq V_0^\varphi$ or $K \subseteq V_\emptyset^\varphi$.*
2. *If $K \subseteq V_\emptyset^\varphi$, then $(S - (X \cup Z))[K]$ is flexible.*
3. *If $K \cap V(X) = \emptyset$, then $K \subseteq V_0^\varphi$.*

Proof. First, note that $S - X$ is homogeneous, and so is the subset of equations in $S - (X \cup Z)$ induced by K. Statement 1 follows by observing that if one variable is assigned zero in a two-variable homogeneous system, then every connected variable must be assigned zero as well. For statement 2, note that if K is rigid, it can only be satisfied by the all-zero assignment. Finally, for statement 3, if $K \cap V(X) = \emptyset$, then K also induces a homogeneous connected component in $S - Z$, which can be satisfies by the all-zero assignment independently of all other variables. □

We drop superscripts in V_0^φ and $V_\emptyset^\varphi$ when φ is clear from the context. Now we introduce *zero-free subgraph* $H_\emptyset$ of the rooted graph $(G_I, \mathcal{B}_I)$.

Definition 28. Let $I = (S, w_S, X, k)$ be an instance of DML($\mathbb{D}$), Z be an optimal solution of I, φ be a satisfying assignment of $S - Z$, and $(G_I, \mathcal{B}_I)$ be the rooted graph for I. Then, *zero-free subgraph* $H_\emptyset := H_\emptyset(I, Z, \varphi)$ of G_I (with distinguished vertex s) is defined as follows. Let $V(H_\emptyset) = V_\emptyset \cup \{s\}$. Add every edge from $(G_I - Z)[V_\emptyset]$ to $E(H_\emptyset)$. Finally, for each zero-free component K of $S - (X \cup Z)$, pick one vertex $x \in K \cap V(X)$ (which exists by Lemma 27) and add the edge $\{s, x\}$ to $E(H_\emptyset)$.

Lemma 29. *For every instance $I = (S, w_S, X, k)$ of* DML($\mathbb{D}$)*, every optimal solution Z to I and every assignment φ satisfying $S - Z$, the zero-free subgraph $H_\emptyset = H_\emptyset(I, Z, \varphi)$ is a connected balanced subgraph of $(G_I, \mathcal{B}_I)$, and $c_{G_I}(H_\emptyset) \leq 3k + 1$.*

Proof. Note that by construction, $H_\emptyset$ contains edges from $G_I - Z$ and edges connecting s to $V(X)$ which are also present in G_I, hence it is a subgraph of G_I. $H_\emptyset$ is clearly connected through the vertex s. To see that all cycles in $H_\emptyset$ are balanced, consider a zero-free component K in $S - (X \cup Z)$. By Lemma 27, $(S - (X \cup Z))[K]$ is flexible so $H_\emptyset[K]$ is a balanced subgraph of $(G_I, \mathcal{B}_I)$ whenever K is zero-free. Finally, the vertex s has exactly one neighbour in each component K with $V(X) \cap K = \emptyset$, so $H_\emptyset$ does not contain any new cycle going through s.

The cost of $H_\emptyset$ in G_I is $c_{G_I}(H_\emptyset) = |Z| + |V(X)| - k_\emptyset$, where $k_\emptyset$ is the number of zero-free components in $S - (X \cup Z)$. Since $1 \leq |Z| \leq k$, $|V(X)| = 2k + 2$, and $k_\emptyset \geq 1$, we have that $c_{G_I}(H_\emptyset) \leq 3k + 1$. □

4.5 The Algorithm

In the end of this section we will present our fpt-algorithm for Min-2-Lin($\mathbb{D}$) with $\mathbb{D} \in \mathsf{H}_2$. By Lemma 21, it suffices to prove that DML($\mathbb{D}$) is in FPT. To this end, let (S, w_S, X, k) be an instance of DML($\mathbb{D}$), Z be a minimum solution, and φ_Z be a satisfying assignment to $S - Z$. Further, assume $F \subseteq S - (X \cup Z)$ is a set of equations such that every rigid component of $S' := S - (X \cup F)$ is zero under φ_Z. Informally, F takes care of all rigid components of $S - X$ that are connected to variables in $V(X)$ assigned to a non-zero value by φ_Z. These components are only satisfied by the all-zero assignment because $S - X$ is homogeneous, so the solution must either make them flexible or must disconnect them from variables in $V(X)$ that have non-zero value in φ_Z; please refer to Figure 6 for an illustration. We call vertices in $V(X \cup F)$ *terminals*, and refer to F as a *cleaning set with respect to* φ_Z. We will later show how to obtain a cleaning set using Observation 26.

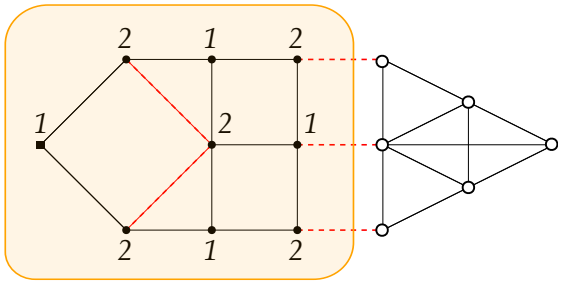


Fig. 6. An illustration of a cleaning set $F \subseteq S - (X \cup Z)$ for an instance (S, w_S, X, k) of DML($\mathbb{F}_3$), where Z is an optimal solution. The picture contains one component of the primal graph of $S - X$, and every edge uv corresponds to an equation $u = 2v$ (note that the equation defines a symmetric relation over $\mathbb{F}_3$, namely $\{(0,0),(1,2),(2,1)\}$). Note that all odd cycles are rigid and all even cycles are flexible. The dashed red edges constitute the cleaning set F. Let ϕ_Z be an assignment satisfying $S - Z$. The leftmost vertex is in $V(X)$ and it is assigned 1 in ϕ_Z. The yellow region contains all vertices assigned nonzero values in ϕ_Z, and these values are given by the corresponding labels. All vertices outside the yellow region (indicated by an empty circle) are assigned 0 in ϕ_Z. Observe that the yellow region contains a balanced subgraph with respect to the set of flexible cycles and its deletion edges are given by F. This is not a coincidence because we will use important balanced subgraphs to compute F; please also compare to Figure 3.

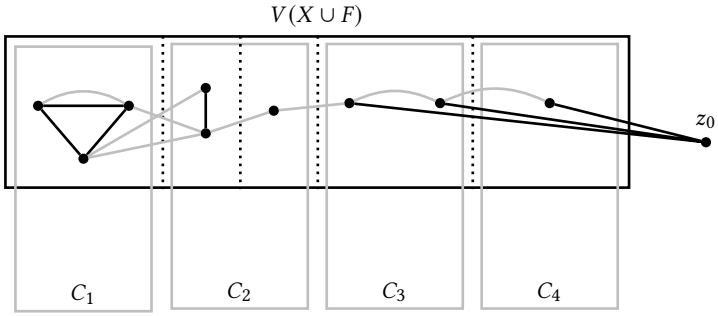


Fig. 7. An illustration of the auxiliary instance $H_{\mathcal{P}} = H(S, X, F, \mathcal{P})$. Here, $C_1, \ldots, C_4$ are all components of $S' = S - (X \cup F)$ with C_3 and C_4 being the only rigid components. Black circular vertices represent the vertices of the primal graph of $H_{\mathcal{P}}$, i.e. the terminals in $V(X \cup F)$. Moreover, edges in light grey represent (possible) edges in $X \cup F$ and black edges represent constraints added to $H_{\mathcal{P}}$, more specifically, black edges inside C_1 and C_2 represent the constraints $e_{xy}(S')$ and black edges incident with z represent the constraints $x - z_0 = 0$ and $x + z_0 = 0$. The dotted vertical lines within the rectangle for $V(X \cup F)$ give the partition $\mathcal{P}$ of $V(X \cup F)$, which is a refinement of the partition $\mathcal{P}'$ given by the components $C_1 \ldots, C_4$.

Let $\mathcal{P}'$ be the partition of terminals into connected components of S' i.e. $\mathcal{P}'(x) = \mathcal{P}'(y)$ if and only if x and y are in the same connected component of S'. For every partition $\mathcal{P}$ that refines $\mathcal{P}'$, we describe the construction of an auxiliary instance $H_{\mathcal{P}} = H(S, X, F, \mathcal{P})$ of 2-Lin($\mathbb{D}$) that is used in the algorithm (see Figure 7 for an illustration). $H_{\mathcal{P}}$ contains all variables in $V(X \cup F)$ plus an additional zero variable z_0. Moreover, $H_{\mathcal{P}}$ contains all equations in $X \cup F$ plus the following additional equations:

- For every terminal x that is in a rigid component of S', the equations $x - z_0 = 0$ and $x + z_0 = 0$.
- For every pair of terminals x, y such that $\mathcal{P}(x) = \mathcal{P}(y)$ and x, y appear in a flexible component of S', the equation $e_{xy}(S')$.

This completes the construction of $H_{\mathcal{P}}$. We distinguish between different kinds of terminals: terminals appearing in rigid components of $H_{\mathcal{P}}$ are called *determined*, while those appearing in flexible components are called *undetermined*. Note that all terminals appearing in the connected component of zero variable z_0 are determined since equations connecting z_0 and z_0' form a rigid cycle. We call them *zero-determined* terminals. Observe further that not all determined terminals have to be zero-determined as $H_{\mathcal{P}}$ may contain rigid components apart from the one including z_0.

If Z is a solution to (S, w_S, X, k) and $\mathcal{P}_Z$ is the partition of terminals into connected components of $S' - Z$, then, intuitively, $H_{\mathcal{P}_Z}$ serves as the "projection" of $S - Z$ onto the terminals i.e. it encapsulates all constraints in $S - Z$ between the pairs of terminals. This intuition is formalised below.

Lemma 30. *Let (S, w_S, X, k) be an instance of* DML($\mathbb{D}$)*, Z be a solution, and φ_Z be a satisfying assignment to $S - Z$. Let $F \subseteq S - (X \cup Z)$ be a cleaning set with respect to φ_Z, and $\mathcal{P}_Z$ be the partition of $V(X \cup F)$ into connected components of $S' - Z$, where $S' := S - (X \cup F)$. Then the following statements hold:*

1. *$H_{\mathcal{P}_Z}$ is consistent.*
2. *If a terminal $x \in V(X \cup F)$ is determined, then $\varphi(x) = \varphi_Z(x)$ for every satisfying assignment φ of $H_{\mathcal{P}_Z}$.*

Proof. *Statement 1.* We show that the assignment φ obtained from φ_Z after setting $\varphi(z_0) = 0$ for the zero variable z_0 satisfies $H_{\mathcal{P}_Z}$. To this end, let e be an equation of $H_{\mathcal{P}_Z}$. If $e \in X \cup F$, then $e \in S - Z$ and φ satisfies e. If e contains z_0, then e is equal to $x - z_0 = 0$ or $x + z_0 = 0$, where x is contained in a rigid component of S'. Because F is a cleaning set with respect to φ_Z, it holds that $\varphi(x) = \varphi_Z(x) = 0$ and therefore e is satisfied by φ. Otherwise, e is equal to $e_{xy}(S')$ for some terminals x and y with $\mathcal{P}_Z(x) = \mathcal{P}_Z(y)$ that appear together in some flexible component K of S'. Because $\mathcal{P}_Z(x) = \mathcal{P}_Z(y)$ and K is flexible, it holds that $e_{xy}(S' - Z)$ is equivalent to $e_{xy}(S')$ and e, and therefore φ satisfies e.

Statement 2. If x is a zero-determined terminal, then $\varphi(x) = 0$ for every satisfying assignment φ to $H_{\mathcal{P}_Z}$. Moreover, $\varphi_Z(x) = 0$ since x is in a rigid component of S' and F is a cleaning set. On the other hand, if x is not zero-determined, then by construction of $H_{\mathcal{P}_Z}$, x is contained in an equivalent rigid cycle in $S - Z$, so φ and φ_Z agree on all terminals in these cycles. Therefore, in both cases we have $\varphi(x) = \varphi_Z(x)$. □

Lemma 30.1 suggests that the algorithm for DML($\mathbb{D}$) can start by guessing the partition $\mathcal{P}$ of the terminals and checking whether $H_{\mathcal{P}}$ is consistent. If yes, then a $\mathcal{P}$-cut Y in S' of size k can be computed in fpt-time (or we can correctly report that no such cut exists). However, $S - Y$ is not necessarily consistent. The reason is that some paths of equations in $S - Y$ may be inconsistent. Thus, the cut needs to fulfil an additional set of requirements to ensure that it is a solution. The key insight for computing these requirements is that all paths avoiding X are homogeneous (hence they imply consistent equations satisfied by setting all variables to zero), so it is sufficient to take care of the paths containing a variable from $V(X)$. Then there are two kinds of inconsistent paths: those confined to a component connecting a terminal and a non-terminal and those connecting two non-terminals in different components using at least one equation from X. We show that these requirements can be handled using Pair Partition Cut. For this we will construct the set $\mathcal{F}_{\mathcal{P}} = \mathcal{F}(S, X, F, \mathcal{P})$ of pair cut requests one needs to fulfil as follows. Let φ_H be a satisfying assignment to $H_{\mathcal{P}}$. Then, the set $\mathcal{F}_{\mathcal{P}} = \mathcal{F}(S, X, F, \mathcal{P})$ of pair cut request contains the following pairs. For every determined terminal x that is in a flexible component K of S', consider every non-terminal v in K and compute $e_{xv}(S')$. Plug in $\varphi_H(x)$ for x into the equation $e_{xv}(S')$. If there is no value for v that satisfies the equation, then add $(\{x, v\}, \{x, v\})$ to $\mathcal{F}_{\mathcal{P}}$.

Now, for every flexible component K of $H_{\mathcal{P}}$, consider every pair of terminals $x, y \in K$ such that $\mathcal{P}(x) \neq \mathcal{P}(y)$. Note that x and y are undetermined. Let K_1' and K_2' be the (not necessarily distinct) components of S' that contain x and y, respectively. Note that $S'[K_1']$ and $S'[K_2']$ are flexible (otherwise, by construction of $H_{\mathcal{P}}$, variables x and y would

form rigid cycles with z_0). For every pair of non-terminals $u \in K'_1$ and $v \in K'_2$, compute $e_{ux}(S'[K'_1])$, $e_{xy}(H_{\mathcal{P}}[K])$, $e_{yv}(S'[K'_2])$, and let e_{uv} be the equation implied by composing them (i.e. treating them as parts of a path, and computing the implied equation). If e_{uv} has no solution, then add $(\{u, x\}, \{y, v\})$ to $\mathcal{F}_{\mathcal{P}}$. This concludes the definition of $\mathcal{F}_{\mathcal{P}}$.

The algorithm for DML($\mathbb{D}$) can now be summarised as follows. Let $I = (S, w_S, X, k)$ be an instance of DML($\mathbb{D}$).

(1) Construct the rooted graph $(G_I, \mathcal{B}_I)$ for I as described in Definition 24. Assume that s is the root of $(G_I, \mathcal{B}_I)$.
(2) Let $\mathcal{G} := \mathcal{G}(G_I, \mathcal{B}_I, k, s)$ be the family of connected balanced subgraphs in $(G_I, \mathcal{B}_I)$ rooted in s with cost at most $3k + 1$. Compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ using Observation 26.
(3) For every $H \in \mathcal{H}$, let F_H be the set of deleted edges excluding those incident to s. Guess the intersection F_Z with a solution, i.e. for every $F_Z \subseteq F_H$ with $w_S(F_Z) \leq k$, do the following. Let $I' = (S', w_S, X, k')$ be the instance obtained from I by removing all edges in F_Z from S and decreasing k by the weight of F_Z. Let $F = F_H \setminus F_Z$, $T = V(X \cup F)$, $S'' = S' - (X \cup F)$, and $\mathcal{P}'$ be the partition of T in S''. Then, for every partition $\mathcal{P}$ that refines $\mathcal{P}'$, proceed as follows:
   (a) Construct the auxiliary instance $H_{\mathcal{P}} = H(S', X, F, \mathcal{P})$ of 2-Lin($\mathbb{D}$) as described above.
   (b) Use Theorem 18 to decide whether $H_{\mathcal{P}}$ is consistent and if so to compute a satisfying assignment φ_H for $H_{\mathcal{P}}$. If $H_{\mathcal{P}}$ is inconsistent, then disregard the current partition $\mathcal{P}$ and continue with the next partition.
   (c) Use φ_H to construct the set $\mathcal{F}_{\mathcal{P}} = \mathcal{F}(S', X, F, \mathcal{P})$ of pair cut requests as described above. Let $I_{\mathcal{P}}$ be the instance $(S'', w_S, T, \mathcal{P}, \mathcal{F}_{\mathcal{P}}, k)$ of Pair Partition Cut.
   (d) Use Theorem 13 to solve $I_{\mathcal{P}}$. If $I_{\mathcal{P}}$ has a solution Y, then use Theorem 18 to check whether $S' - Y$ is consistent. If so, output $Y \cup F_Z$ as the solution for DML($\mathbb{D}$), otherwise disregard the current partition $\mathcal{P}$ and continue with the next partition.
(4) If no solution was output at Step 3d, then reject.

4.6 Correctness Proof and Complexity Analysis

We will now prove that the algorithm presented in Section 4.5 is correct and we will analyze its time complexity. The correctness proof is based on an auxiliary result (Lemma 32) that shows the connection between the cleaned DML($\mathbb{D}$) instance and the Pair Partition Cut instances that are computed in step 3 of the algorithm. The proof of Lemma 32 is simplified with the aid of the following lemma.

Lemma 31. *Let (S, w_S, X, k) be an instance of* DML($\mathbb{D}$) *with solution Z and let φ_Z be a satisfying assignment of $S - Z$. Let $F \subseteq S - (X \cup Z)$ be a cleaning set with respect to φ_Z, and $\mathcal{P}_Z$ be the partition of $V(X \cup F)$ into connected components of $S' - Z$, where $S' = S - (X \cup F)$. Then Z is a $\mathcal{P}_Z$-cut in S' that fulfils $\mathcal{F}_{\mathcal{P}_Z}$.*

Proof. Clearly, Z is a $\mathcal{P}_Z$-cut. Suppose now for a contradiction that Z does not fulfil $\mathcal{F}_{\mathcal{P}_Z}$. First consider the case that Z does not fulfil a cut request $(\{x, v\}, \{x, v\})$ in $\mathcal{F}_{\mathcal{P}_Z}$, where x is a determined terminal. Because of Lemma 30.1, we know that $H_{\mathcal{P}_Z}$ is consistent. Let φ_H be a satisfying assignment to $H_{\mathcal{P}_Z}$ and let K contain the connected component of S' that contains x and v. By Lemma 30.2, $\varphi_H(x) = \varphi_Z(x)$. Since Z does not separate x and v in S', at least one path implying the equation $e_{xv}(S'[K])$ persists in $S - Z$. However, due to the construction of $\mathcal{F}_{\mathcal{P}_Z}$ this implies that φ_Z does not satisfy $e_{xv}(S'[K])$ and this contradicts our assumption that $S - Z$ is consistent.

Now consider the only remaining case that Z does not fulfil a cut request $(\{u, x\}, \{y, v\})$ in $\mathcal{F}_{\mathcal{P}_Z}$, where x and y are undetermined terminals. Let K'_1 and K'_2 be the connected components of S' such that $\{u, x\} \subseteq K'_1$ and $\{y, v\} \subseteq K'_2$. Further, let K be the connected component of $H_{\mathcal{P}_Z}$ that contains x and y. Since Z does not disconnect u, x or y, v in S', a path implying $e_{ux}(S'[K'_1])$ and a path implying $e_{yv}(S'[K'_2])$ persist in $S - Z$. Moreover, by the construction of $H_{\mathcal{P}_Z}$, a

path implying $e_{xy}(H_{\mathcal{P}_Z}[K])$ exists in $S - Z$. Finally, the construction of $\mathcal{F}_{\mathcal{P}_Z}$ ensures that the composition of these equations does not have a solution in $\mathbb{D}$. We conclude that $S - Z$ is inconsistent and this leads to a contradiction. □

Lemma 32. *Let $I = (S, w_S, X, k)$ be an instance of* DML($\mathbb{D}$) *with solution Z and let φ_Z be a satisfying assignment of $S - Z$. Let $F \subseteq S - (X \cup Z)$ be a cleaning set with respect to φ_Z, and let $\mathcal{P}_Z$ be the partition of $V(X \cup F)$ into connected components of $S' - Z$, where $S' = S - (X \cup F)$. Then every minimum $\mathcal{P}_Z$-cut Y in S' that fulfils $\mathcal{F}_{\mathcal{P}_Z}$ is a solution to I.*

Proof. We know that $H_{\mathcal{P}_Z}$ is consistent by Lemma 30.1. We let φ_H denote a satisfying assignment. We construct an assignment φ_Y based on φ_H and prove that φ_Y satisfies $S'' := S' - Y$, considering one connected component of S'' at a time. Then we show that φ_Y also satisfies $X \cup F$, and conclude that it satisfies $S - Y$.

First note that every connected component of S'' is a subset of a component of S'. If a variable v appears in a rigid component of S', then let $\varphi_Y(v) = 0$. If v appears in a component that does not contain any terminal, then let $\varphi_Y(v) = 0$. Note that all equations in $S - X$ are homogeneous so φ_Y satisfies all equations inside the components of S'' considered so far.

Now consider a flexible component K of S'' that contains a determined terminal x. Set $\varphi_Y(x) = \varphi_H(x)$. Since Y fulfils $\mathcal{F}_{\mathcal{P}_Z}$, for every $v \in K$ the equation $e_{xv}(S'')$ has a solution where $x \mapsto \varphi_H(x)$. Therefore, we can extend φ_Y, by assigning every variable $v \in K$ with $v \neq x$ to the unique value satisfying $e_{xv}(S'')$ if x is set to $\varphi_Y(x)$. It follows that φ_Y satisfies star(K, x). Combined with the fact that Helly dimension of $\mathbb{D}$ is 2, Lemma 17 implies that φ_Y satisfies $S''[K]$.

All remaining variables appear in flexible components of S'' that only contain undetermined terminals. Let U be the set of all vertices appearing in these components.

Claim 32.1. *$(S - Y)[U]$ is flexible and consistent.*

Proof of claim: Towards showing that $(S-Y)[U]$ is flexible, first note that $S''[U]$ is flexible. Moreover, $H_{\mathcal{P}_Z}[U \cap V(X \cup F)]$ is also flexible, since all terminals in U are undetermined. Thus, $(S - Y)[U]$ does not contain any rigid cycle avoiding $X \cup F$. Furthermore, if there were a rigid cycle in $(S - Y)[U]$ intersecting $X \cup F$, then by construction there would also be such a cycle in $H_{\mathcal{P}_Z}[U \cap V(X \cup F)]$, which would be a contradiction. Hence, $(S - Y)[U]$ cannot contain a rigid cycle and it is indeed flexible.

We now show that $(S - Y)[U]$ is consistent. Because $(S - Y)[U]$ is flexible, Observation 19 implies that $(S - Y)[U]$ is consistent if it does not contain an inconsistent path. Suppose for a contradiction that $(S - Y)[U]$ contains an inconsistent path P between say u and v. We know that $S - X$ is consistent so we can assume that P intersects X. Let $x \in V(X \cup F)$ and $y \in V(X \cup F)$ be the closest terminals to u and v on P, respectively. Then, the equation $e_{ux}(P)$ is equivalent to $e_{ux}(S')$ and similarly the equation $e_{yv}(P)$ is equivalent to $e_{yv}(S')$. Moreover, an equation equivalent to the equation $e_{xy}(P)$ is implied by the xy-paths in $H_{\mathcal{P}_Z}$ due to the construction of $H_{\mathcal{P}_Z}$. Therefore, if P is inconsistent, then so is the equation obtained by combining $e_{ux}(S')$, $e_{xy}(P)$, and $e_{yv}(S')$, which implies that $(\{x, u\}, \{y, v\})$ is a pair cut request in $\mathcal{F}_{\mathcal{P}_Z}$. This contradicts our assumption that P is in $(S - Y)[U]$ because Y fulfils $\mathcal{F}_{\mathcal{P}_Z}$ and therefore intersects P. ◇

Using the claim above, we can now extend φ_Y to U using any satisfying assignment φ_U of $(S - Y)[U]$ by setting $\varphi_Y(u) = \varphi_U(u)$ for all $u \in U$. We show that φ_Y obtained in this manner satisfies not only $X \cup F$ but also $H_{\mathcal{P}_Z}$.

Claim 32.2. *The assignment φ_Y satisfies $H_{\mathcal{P}_Z}$.*

Proof of claim: Let K be a connected component of $H_{\mathcal{P}_Z}$. If $z_0 \in K$, then $\varphi_H(v) = 0$ for all $v \in K$. By the construction of $H_{\mathcal{P}_Z}$, $K \setminus \{z_0\}$ is a subset of a rigid component of S' so $\varphi_Y(v) = 0$ for all $v \in K$. If $z_0 \notin K$ and K is rigid, then K only

contains determined terminals and it follows that φ_Y agrees with φ_H on K by construction. Finally, if K is flexible, then consider arbitrary $x, y \in K$. By the construction of $H_{\mathcal{P}_Z}$, there is a path in $(S - Y)[U]$ that implies $e_{xy}(H_{\mathcal{P}_Z}[K])$. Hence, φ_Y satisfies $e_{xy}(H_{\mathcal{P}_Z}[K])$ for all $x, y \in K$. We have thus exhausted all cases and the claim holds. ⋄

We have shown that φ_Y satisfies both $S'' = S' - Y$ and $X \cup F \subseteq H_{\mathcal{P}_Z}$. Therefore, $S - Y$ is consistent and it only remains to show that $|Y| \leq k$. Lemma 31 implies that Z is a $\mathcal{P}_Z$-cut in S' that fulfils $\mathcal{F}_{\mathcal{P}_Z}$. Moreover, Z is a solution of I so $|Z| \leq k$. It follows that if Y is a minimum such $\mathcal{P}_Z$-cut, then $|Y| \leq k$ and Y is a solution of I. □

We are now ready to prove correctness and to provide the time complexity analysis of the algorithm. For the analysis of the run-time, we will use $Q(k)$ to denote the run-time dependency on the parameter k for the algorithm for Pair Partition Cut, i.e. $O^*(Q(k))$ is the run-time for the algorithm for Pair Partition Cut given in Theorem 13. The running time of Pair Partition Cut is significant: The running time given by Kim et al. [38] for MinCSP(Γ) where Γ is bijunctive and $2K_2$-free is not stated explicitly, but if applied to instances with solution cost k and maximum constraint arity b, then it is $O^*(2^{p(k,b)})$ for a polynomial $p(k, b)$ of non-trivial degree (see also [38, Theorem 3.11] which implies $p(k, b) = \Omega(k^4 b^8)$, and in our usage we have $b = \Theta(k)$). This makes it clear that the main bottleneck for our algorithm is the underlying algorithm for Pair Partition Cut.

Theorem 33. *Min-2-Lin($\mathbb{D}$) is in* FPT *and can be solved in time* $O^*(k^{O(k)}Q(k))$.

Proof. We start by analysing the algorithm for DML($\mathbb{D}$) presented in Section 4.5. Let $I = (S, w_s, X, k)$ be an arbitrary instance of DML($\mathbb{D}$). We show that the algorithm accepts if and only if I is a yes-instance. The forward direction is simple because if the algorithm returns a solution Y, then $|Y| \leq k$ and $S - Y$ is consistent because of Step 3d of the algorithm.

Towards showing the reverse direction, suppose that I is a yes-instance having a solution Z. Let φ_Z be a satisfying assignment of $S-Z$ and define V_0 and $V_\emptyset$ accordingly. Let $H_\emptyset$ denote the zero-free subgraph of G_I as given in Definition 28. By Lemma 29, $H_\emptyset$ is balanced, connected, and $c_{G_I}(H_\emptyset) \leq 3k + 1$. Because the family $\mathcal{H}$ that is computed in Step 2 of the algorithm is a dominating family for $\mathcal{G}$, there is an (important) balanced subgraph $H \in \mathcal{H}$ that dominates $H_\emptyset$. Moreover, because $H \in \mathcal{H}$, H is considered by the algorithm in Step 3.

Let F_H be the corresponding set of deleted edges in $G_I - \{s\}$, let $F_Z = F_H \cap Z$, and let $F = F_H \setminus F_Z$. Let $I' = (S', w_S, X, k')$ be the instance obtained from I by removing all edges in F_Z from S and decreasing k by the weight of F_Z. Note that I has a solution if and only if I' has a solution. Moreover, note that F is considered by the algorithm because the algorithm considers all subsets of F_Z of F_H of weight at most k in Step 3. Let $T = V(X \cup F)$, $S'' = S' - (X \cup F)$ and let $\mathcal{P}'$ be the partition of T in S''. Let $\mathcal{P}_Z$ be the partition of T in $S'' - Z$. Then, because the algorithm considers all refinements of $\mathcal{P}'$, it also considers the partition $\mathcal{P}_Z$. Finally, note that F is a cleaning set with respect to φ_Z in S'. This is because $V_\emptyset \subseteq V(H_\emptyset) \subseteq V(H)$ and all components in $S''[V(G_I) \setminus \{s\}]$ are flexible. Hence, all variables in the rigid components of S'' are assigned zero values by φ_Z. Therefore, $Z \setminus F_Z$, φ_Z, $\mathcal{P}_Z$, and F satisfy all conditions of Lemma 30.1 for the instance I', which implies that $H_{\mathcal{P}_Z}$ is consistent. Moreover, $Z \setminus F_Z$, φ_Z, $\mathcal{P}_Z$, and F also satisfy all conditions of Lemma 32 on the instance I' and therefore every $\mathcal{P}_Z$-cut Y in S'' that fulfils $\mathcal{F}_{\mathcal{P}_Z}$ is a solution for I'. Therefore, the set $Y \cup F_Z$ returned by the algorithm in Step 3d is a solution for I.

We continue by analysing the run-time of the algorithm. The algorithm starts by computing a dominating family $\mathcal{H}$ of $\mathcal{G} := \mathcal{G}(G_I, \mathcal{B}_I, k, s)$ of size at most 4^{3k+1} in time $O^*(4^{3k+1})$ using Observation 26. Let $H \in \mathcal{H}$ and let F_H be the set of deleted edges for H excluding those incident with s. Then, for every F_H, the algorithm considers at most $2^{|F_H|} \leq 2^{3k+1}$ (because $c_{G_I}(H) \leq 3k + 1$) subsets F_Z and computes the updated instance $I' = (S', w_S, X, k')$ in Step 3 in

polynomial-time. Let $F = F_H \setminus F_Z$, $S'' = S' - (X \cup F)$, $T = V(X \cup F)$, and let $\mathcal{P}'$ be the partition of $T = V(X \cup F)$ in S''. The algorithm then enumerates all refinements $\mathcal{P}$ of $\mathcal{P}'$. Because the number of such refinements $\mathcal{P}$ is at most $|T|^{|T|} \leq (4k)^{4k}$, this can be achieved in time $O((4k)^{4k})$. For each $\mathcal{P}$, the algorithm then constructs $H_{\mathcal{P}} = H(S', X, F, \mathcal{P})$ in polynomial-time and decides whether $H_{\mathcal{P}}$ is consistent in polynomial time using Theorem 18. If $H_{\mathcal{P}}$ is not consistent, the algorithm stops, otherwise it constructs the set of pair-cut requests $\mathcal{F}_{\mathcal{P}} = \mathcal{F}(S', X, F, \mathcal{P})$ and the instance $I_{\mathcal{P}}$ of Pair Partition Cut in polynomial-time. Finally, the algorithm solves $I_{\mathcal{P}} = (S'', w_S, T, \mathcal{P}, \mathcal{F}, k)$ using Theorem 13 in fpt-time with respect to $k' \leq k$, i.e. in time $O^*(Q(k))$. Therefore, the total time required by the algorithm is at most

$$O^*(4^{3k+1}2^{3k+1}(4k)^{4k}Q(k)) = O^*(k^{O(k)}Q(k))$$

which shows that DML($\mathbb{D}$) is fpt with respect to k. By Lemma 21, there is another factor of 2^k in the running time of the algorithm for Min-2-Lin($\mathbb{D}$), which is dominated by $k^{O(k)}$, so asymptotically we obtain the same running time for Min-2-Lin($\mathbb{D}$). □

4.7 Applicability of the Algorithm: Prüfer Domains

We will now consider a large class of commutative domains, known as *Prüfer* domains, that is known to contain many interesting and well-studied rings. We show that every Prüfer domain has Helly dimension 2 so Theorem 33 implies that Min-2-Lin($\mathbb{D}$) is in FPT whenever $\mathbb{D}$ is a Prüfer domain that satisfies properties P3 and P4 from Theorem 18. Prüfer domains can be defined in many equivalent ways. For instance, the survey by Bazzoni and Glaz lists 22 definitions [4, Theorem 1.1]. The most suitable one for us is the following definition based on the ideals. The set of all ideals of a given ring forms a lattice with sum as the join operation and intersection as the meet operation. Recall that a lattice L is *distributive* if

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$$

for all $x, y, z \in L$. The lattice of ideals is, in general, not a distributive lattice. Rings where this lattice is indeed distributive are known as *arithmetical rings*; see [53] for a compact overview. A *Prüfer domain* is a commutative arithmetical ring that is an integral domain. The relations between classes of rings that we discuss below are well-known and a nice summary can be found in the survey by Bouvier [7]. Prüfer domains generalize many well-studied classes of integral domains such as *Bézout domains* and *Dedekind domains*. A Bézout domain is an integral domain in which the sum of two principal ideals (i.e. ideals generated by a single element) is again a principal ideal. It follows immediately that Bézout domains include the *principal ideal domains* (where every ideal is principal) and thus *Euclidean domains*. All fields, the ring of integers $\mathbb{Z}$, Gaussian integers $\mathbb{Z}[i]$, Eisenstein integers $\mathbb{Z}[\omega]$ (where ω is a primitive non-real cubic root of unity), the ring of polynomials $\mathbb{F}[x]$ over a field $\mathbb{F}$, and many more integral domains are Euclidean domains. Furthermore, every *valuation ring* (where for every $r, q \in R$, at least one of r/q and q/r is in R) is a Bézout domain. If a Prüfer domain satisfies the ascending chain condition on ideals (i.e. it is *Noetherian*), then it is a *Dedekind domain*. This implies, for instance, that the ring O_K of algebraic integers in a number field K is a Prüfer domain. We show that Prüfer domains have Helly dimension 2.

Lemma 34. *Prüfer domains have Helly dimension 2.*

Proof. Let $C_1 = (a_1) + b_1$, $C_2 = (a_2) + b_2$, and $C_3 = (a_3) + b_3$ be a triple of principal ideal cosets that are pairwise non-disjoint. We will show that there is a common element in all three. First, observe that we can assume without loss of generality that $b_1 = b_2 = 0$: indeed, let $d \in C_1 \cap C_2$ (which exists because $C_1 \cap C_2 \neq \emptyset$); then $C_1 - d = (a_1)$, $C_2 - d = (a_2)$ and if $p \in (C_1 - d) \cap (C_2 - d) \cap (C_3 - d)$, then $p + d \in C_1 \cap C_2 \cap C_3$.

Since C_1 and C_3 intersect, we have $(a_1) \cap ((a_3) + b_3) \neq \emptyset$, so $b_3 \in (a_1) + (a_3)$. By a similar argument, since C_2 and C_3 intersect, we have $b_3 \in (a_2) + (a_3)$. Recall that $\mathbb{D}$ is a Prüfer domain, so by distributivity we have

$$b_3 \in ((a_1) + (a_3)) \cap ((a_2) + (a_3)) = ((a_1) \cap (a_2)) + (a_3),$$

i.e. there exists $q \in (a_3)$ such that $q + b_3 \in (a_1) \cap (a_2)$. Clearly, $q + b_3 \in C_3$, so $q + b_3$ is in the intersection of all three cosets. □

5 FASTER ALGORITHM FOR FIELDS

Let $\mathbb{F}$ be an effective field. In this section we present improved fpt-algorithms for Min-2-Lin($\mathbb{F}$)—an $O^*(k^{O(k)})$-time algorithm for arbitrary fields and an $O^*((2p)^k)$-time algorithm for finite p-element fields. Section 5.1 demonstrates how Min-2-Lin($\mathbb{F}$) differs from the more general Min-2-Lin over rings in H_2 and several useful observations are derived from this. In Section 5.2, we present the algorithm for fields and continue in Section 5.3 by proving its correctness and analysing its running time. Finally, we present a faster algorithm for Min-2-Lin over finite fields in Section 5.4.

5.1 2-Lin over Fields

Since the quotient of any two nonzero elements is an element of the field $\mathbb{F}$, instances of 2-Lin($\mathbb{F}$) enjoy rather pleasant properties that do not necessarily hold for rings in H_2. First, note that any single equation $ax+by = c$ over $\mathbb{F}$ is consistent unless $a = b = 0$ and $c \neq 0$. By preprocessing, we may assume that such equations do not occur in our instances. Hence, we may assume that all paths in our instances are consistent. This implies the following via Lemma 15 and Observation 19.

Corollary 35. *Every flexible instance S of 2-Lin($\mathbb{F}$) is consistent. Moreover, for any variable $z \in V(S)$ and any element d in $\mathbb{F}$, there is an assignment that satisfies S and sets z to d.*

Flexible instances have another useful property. We call a variable substitution Φ *equalising* if every equation in $\Phi(S)$ is equality, i.e. it has the form $x = y$.

Lemma 36. *Every flexible instance of 2-Lin($\mathbb{F}$) admits an equalising variable substitution.*

Proof. Let S be a connected flexible instance of 2-Lin($\mathbb{F}$). The instance S is consistent by Corollary 35 so Lemma 20 allows us to assume that S is homogeneous. We may additionally assume (by division of field elements) that every equation is of the form $x = ay$ for some $a \in \mathbb{F}$. Pick an arbitrary variable $z \in V(S)$ and construct a spanning tree $T \subseteq S$ rooted in z. Define a variable substitution Φ by $x \mapsto a_x x'$, where $x = a_x z$ is the equation $e_{xz}(S)$. Note that this map is reversible since division is available in $\mathbb{F}$. Clearly, $\Phi(S)$ is homogeneous. Moreover, equation $e_{x',z'}(\Phi(S))$ is $a_x x' = a_x z'$ which simplifies $x' = z'$. We conclude that every equation in $\Phi(S)$ is equality. □

Yet another consequence of division in $\mathbb{F}$ is the following lemma that allows us to remove a factor of $2^{O(k)}$ from the time complexity of iterative compression. We remark that the improvement is irrelevant for infinite fields like $\mathbb{Q}$, for which our algorithms are superexponential in k, but will become pronounced in Section 5.4 where we are dealing with finite fields and care about the base of the exponent.

Lemma 37. *If* DML($\mathbb{F}$) *is solvable in $O^*(f(k))$ time, then Min-2-Lin($\mathbb{F}$) is solvable in $O^*(f(k))$ time.*

Proof. Given an instance (S, w_S, k) of Min-2-Lin($\mathbb{D}$), apply *(equation) subdivision* to it: for every equation e of the form $ax + by = c$ in the instance, introduce a new variable z_e and replace the original equation by a *subdivided pair* of equations $P_e = \{x = z_e, az_e + by = c\}$. Both equations in the pair are assigned the same weight as the original one.

Clearly, any minimal solution only needs to contain one equation from each subdivided pair. Hence, the resulting instance has a solution of weight k if and only if the original instance has one. Moreover, when applying iterative compression to (S, w_S, k) and having a suboptimal but minimal solution X at hand, we may safely assume that the optimal solution Z to the instance is disjoint from X (e.g. if X and Z need to separate the same pair of original variables, they may choose different equations from the subdivided pair). Hence, there is no need to branch on the intersection of X and Z and the instance can be solved directly by passing it to the DML($\mathbb{F}$) algorithm. □

5.2 Algorithm for Min-2-Lin over Fields

Let $I = (S, w_s, X, k)$ be an instance of DML($\mathbb{F}$). By Lemma 21, it suffices to construct an fpt-algorithm for the latter problem. The opening of the algorithm is equation subdivision which allows for speeding up iterative compression by Lemma 37. In fact, we apply subdivision twice to replace every equation with three new ones, i.e. two fresh variables z_1 and z_2 are introduced and $ax + by = c$ is replaced by $\{x = z_1, z_1 = z_2, az_2 + by = c\}$. This allows us to avoid several branching steps—more details are given after the algorithm description. Then we construct the rooted graph $(G_I, \mathcal{B}_I)$ and compute a dominating family of important balanced subgraphs to obtain a cleaning set F. In contrast to the algorithm for rings in H_2, the following steps are simplified by the additional structure of fields. In the iterative compression step, it is ensured that the solution is disjoint from $X \cup F$ simply by using subdivision as in Lemma 37. The cutting step is simplified even more dramatically: it turns out that guessing the correct partition of the terminals $\mathcal{P}$ and computing a minimum $\mathcal{P}$-cut is sufficient since there are no inconsistent paths in the instances of 2-Lin($\mathbb{F}$). We claim that the following algorithm solves the instance $I = (S, w_S, k, X)$ of DML($\mathbb{F}$) in $O^*(2^{O(k \log k)})$ time.

(1) Apply equation subdivision (like in Lemma 37) twice to (S, w_S, k) so that every equation is divided into three equations.
(2) Construct the rooted graph $(G_I, \mathcal{B}_I)$ for I as described in Definition 24. Assume that s is the root.
(3) Let $\mathcal{G} := \mathcal{G}(G_I, \mathcal{B}_I, k, s)$ be the family of connected balanced subgraphs in $(G_I, \mathcal{B}_I)$ rooted in s with cost at most $3k + 1$. Compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ using Observation 26.
(4) For every $H \in \mathcal{H}$, let F_H be the set of deleted edges excluding those incident to s. For each partition $\mathcal{P}$ of $V(X \cup F_H)$, check if there is a $\mathcal{P}$-cut Y in $S - (X \cup F_H)$ of size at most k using Lemma 11. If Y exists and $S - Y$ is consistent, then output Y. Otherwise continue with the next partition.
(5) If no solution was output in the previous step, then reject I.

The double subdivision in step 1 allows us to assume that an optimal solution Z, the set X, and the current cleaning set F_H are pairwise disjoint. We can thus avoid branching on their intersections (analogously to how the iterative compression algorithm for fields presented in Lemma 37 avoids the branching step in the general compression algorithm from Lemma 21).

5.3 Correctness Proof and Complexity Analysis

We start with a lemma that will help us prove correctness of the algorithm. This lemma can be viewed as an analogue of Lemma 32 but its proof is noticeably different.

Lemma 38. *Let $I = (S, w_S, X, k)$ be an instance of* DML($\mathbb{F}$) *with solution Z and let φ_Z be a satisfying assignment of $S - Z$. Let $F \subseteq S - (X \cup Z)$ be a cleaning set with respect to φ_Z, and let $\mathcal{P}$ be the partition of $V(X \cup F)$ into connected components of $S' - Z$, where $S' = S - (X \cup F)$. Then every minimum $\mathcal{P}$-cut in S' is a minimum solution for (S, w_S, X, k).*

Proof. Let K be a component of S' which does not contain any rigid cycles. Then K is flexible, and by Lemma 36, we can perform a substitution $\Phi(S)$ on S such that K becomes equalised (i.e. all equations of $\Phi(S)[K]$ except those in $X \cup F$ are equalities). Perform this substitution for all flexible components K of S', and let φ_F be the updated satisfying assignment to $S - Z$. Observe that $\varphi^{-1}(0) = \varphi_F^{-1}(0)$ since $S - X$ is homogeneous. As before, we refer to the vertices of $V(X \cup F)$ as terminals. Consider a set $B \in \mathcal{P}$. Lemma 27 implies that if one variable in B is assigned the zero value, then all variables in B are assigned the zero value by φ_F. On the other hand, if no variable in B is assigned zero value, then, by variable substitution, all paths connecting variables in B imply equalities between all variables in B. Hence, φ_F is constant on every $B \in \mathcal{P}$ and every connected component of $S' - Z$.

Now, let Y be a minimum $\mathcal{P}$-cut. We show that Y is a solution by constructing an assignment φ_Y that satisfies $S - Y$. Let $\varphi_Y(v) = \varphi_F(v)$ for any terminal $v \in V(X \cup F)$, propagate values so that every connected component of $S' - Y$ takes the same value on every vertex, and set $\varphi_Y(v) = 0$ for any vertex v in a connected component of $S' - Y$ without terminals. We note that φ_Y is well-defined. Indeed, if u and v are terminals such that $\varphi_F(u) \neq \varphi_F(v)$, then u and v are in different parts of $\mathcal{P}$. Since Y is a $\mathcal{P}$-cut, no component of $S' - Y$ contains both u and v.

Consider an arbitrary equation $e \in S - Y$. If $e \in X \cup F$, then φ_Y matches φ_F on e. Since $e \notin Z$ by assumption, this implies that φ_Y satisfies e. Next, assume that e is in a flexible connected component K of S'. We know that $e \notin X \cup F$ and e is equality so by construction both variables of e take the same value in φ_Y. Finally, assume that e appears in a rigid component K of S'. By assumption, φ_F assigns zero to K. Assume first that there exists a path P in $S' - Y$ connecting e to a terminal v. Then necessarily P is contained in K and $\varphi_Y(v) = \varphi(v) = 0$. If no such path exists, then the variables in e take the value zero by default. In both cases, the variables in e are assigned zero and e is satisfied by φ_Y. This exhausts the cases and shows that Y is a solution.

Since Y is a minimum-weight $\mathcal{P}$-cut and Z is a $\mathcal{P}$-cut by definition, we conclude that Y is an optimal solution. □

Now we are ready to present the correctness proof and the analysis of the running time of the algorithm.

Theorem 39. *Min-2-Lin($\mathbb{F}$) can be solved in $O^*(2^{O(k \log k)})$ time.*

Proof. By Lemma 37, it suffices to analyze the algorithm for DML($\mathbb{D}$) that was presented at the end of Section 5.2. Let $I = (S, w_s, X, k)$ be an arbitrary instance of this problem. We show that the algorithm accepts if and only if I is a yes-instance. For the forward direction, note that if the algorithm finds a solution Y, then $|Y| \leq k$ and $S - Y$ is consistent because of Step 4 of the algorithm.

Towards showing the reverse direction, suppose that I is a yes-instance and Z is an optimal solution. Let φ be an assignment satisfying $S - Z$, and define V_0 and $V_\emptyset$ as in Section 4.4 i.e. $V_0 = \{v \in V(S) \mid \varphi(v) = 0\}$ and $V_\emptyset = \{v \in V(S) \mid \varphi(v) \neq 0\}$. Let $H_\emptyset$ denote the zero-free subgraph of G_I (see Definition 28). By Lemma 29, the subgraph $H_\emptyset$ is balanced and connected, and $c_G(H_\emptyset) \leq 3k + 1$. Hence, there is an important balanced subgraph $H \in \mathcal{H}$ considered by the algorithm in line 4 that dominates $H_\emptyset$. Let F_H be the corresponding set of deleted edges in $G_I - \{s\}$ and let $\mathcal{P}_Z$ be the partition of the terminals $T = V(X \cup F_H)$ into connected components of $S - (X \cup Z)$. The algorithm exhaustively considers all possible partitions $\mathcal{P}$ of T and tries to compute a minimum $\mathcal{P}$-cut in $S' := S - (X \cup F_H)$. We wish to apply Lemma 38 to prove that such a cut exists so we verify that the preconditions of the lemma are met. By subdividing equations into three parts in the first step of the algorithm, we can assume without loss of generality that X, F_H and

Z are pairwise disjoint. Further, we note that $V_\emptyset \subseteq V(H_\emptyset) \subseteq V(H)$ and all components in $S'[V(H) \setminus \{s\}]$ are flexible. Hence, all variables in the rigid components of S' are assigned zero values by φ, the set F_H is indeed a cleaning set with respect to φ_Z, and the lemma applies. We conclude that the algorithm accepts the instance I.

We continue by analysing the time complexity of the algorithm. Using Observation 26, the algorithm computes a dominating family $\mathcal{H}$ of $\mathcal{G}$ of size at most 4^{3k+1} in time $O^*(4^{O(k)})$. Let $H \in \mathcal{H}$ and let F_H be corresponding set of deleted edges excluding those incident to vertex s. Note that $c_{G_I}(H) \leq 3k+1$. For each H, every partition $\mathcal{P}$ of $V(X \cup F_H)$ is computed in line 4. Recall that $|X| = k+1$ and $|F_H| \leq 3k+1$ by Lemma 29 so $|V(X \cup F_H)| \leq 4k$ and the enumeration of partitions requires $O^*((4k)^{O(4k)})$ time. Computing the $\mathcal{P}$-cut requires at most $O^*(2^{4k})$ time by Lemma 11 and the total running time is

$$O^*(4^{O(k)}) + O^*(4^{O(k)}) \cdot O^*((4k)^{O(4k)}) \cdot O^*(2^{4k}) \in O^*(2^{O(k \cdot \log k)}).$$

□

5.4 Even Faster Algorithm for Finite Fields

Let $\mathbb{F}_p$ be a finite p-element field with $p \geq 3$. Every finite field obviously has an effective representation so we assume without loss of generality that $\mathbb{F}_p$ is effective. We recall *Wedderburn's Little Theorem* (see, for instance, [28]).

Theorem 40. *Every finite domain is a field.*

Hence, the results in this section cover Min-2-Lin for every finite domain. As mentioned in the introduction, Min-2-Lin($\mathbb{F}_p$) is a special case of ULC with a finite alphabet, so it can be solved in $O^*(p^{2k})$ time by the currently best algorithm for ULC [30]. In this section we present a faster algorithm for Min-2-Lin($\mathbb{F}_p$) that runs in $O^*((2p)^k)$ time. Note, however, that a slightly faster $O^*(1.977^k)$-time algorithm for Min-2-Lin($\mathbb{F}_2$) can be obtained using the approach of [49].

By equation subdivision and Lemma 37, the problem can be reduced to polynomially many instances of DML($\mathbb{F}_p$). Let (S, w_s, X, k) be an instance of the latter problem. The key to improved running time of our algorithm is the fact that X has at most p^k satisfying assignments, and an optimal assignment to S must extend one of these assignments. Suppose $\alpha : V(X) \to \mathbb{F}_p$ is an assignment that satisfies X. Then the problem can be solved by checking whether $S - X$ admits an assignment that extends α and leaves unsatisfied equations of total weight at most k. A reduction to RBGCE allows us to answer this question in $O^*(2^k)$ time. The reader should note that this approach avoids using the method of important balanced subgraphs.

We continue with some definitions. Given an instance S of 2-Lin($\mathbb{D}$), a subset of equations X such that $S - X$ is consistent, and an assignment α satisfying X, we define S_α as follows: start with all equations of $S - X$, introduce two new variables s and t, and add equations $x = s \cdot \alpha(x)$ of weight $k+1$ for all $x \in V(X)$ where $\alpha(x) \neq 0$, and $x = t$ of weight $k+1$ for all $x \in V(X)$ where $\alpha(x) = 0$. Finally, add two more variables t', t'' and equations $t' = \gamma t$, $t'' = t'$, $t = t''$ each of weight $k+1$, where γ is any element in $\mathbb{F}_p \setminus \{0, 1\}$. We refer to S_α as the *restriction of S to α*. Note that S_α is homogeneous by construction. Furthermore, setting s to 1 and t to 0 implies that the variables in $V(X)$ are assigned the values in accordance with α. Let G_α be the primal graph of S_α and define $\mathcal{B}_\alpha$ to be the family of flexible cycles in S_α. Since all equations in S_α are homogeneous, we can view it as a group-labelled graph with the group being $\mathbb{F}_p^*$ i.e. the multiplicative group of the field. We recall a result of Zaslavsky [56] and immediately obtain the following corollary.

Lemma 41 (Proposition 5.1 in [56]). *Let G be a graph with edges labelled by elements of a group Γ. Let $\mathcal{B}$ be the set of flexible cycles in G, i.e. cycles such that combining edge labels along the cycle using the group operation yields the identity element of Γ. Then $(G, \mathcal{B})$ is a biased graph.*

Corollary 42. *The pair $(G_\alpha, \mathcal{B}_\alpha)$ defined above is a biased graph.*

Clearly, there is a polynomial time algorithm that checks whether a cycle is flexible since we can multiply all labels along the cycle and check whether the result equals identity. Now we are ready to prove the theorem.

Theorem 43. *Min-2-Lin($\mathbb{F}_p$), where $\mathbb{F}_p$ is a finite p-element field with $p \geq 3$, is in* FPT *and solvable in $O^*((2p)^k)$ time.*

Proof. By equation subdivision and Lemma 37, we can focus on DML($\mathbb{F}_p$). Let (S, w_S, k, X) be an instance of this problem. Pick one variable from each equation in X and place them into a set U. Note that $|U| \leq |X| \leq k+1$. Enumerate assignments $\alpha : U \to \mathbb{F}_p$. For each α, propagate the values from the variables in U to $V(X) \setminus U$ according to the equations of X. If no conflict arises, i.e. if α satisfies X, then construct the restriction S_α of S to α. Recall that G_α is the primal graph of S_α. In the following we identify the edges of G_α and the equations of S_α.

Claim 43.1. *Suppose there exists $Z \subseteq S_\alpha$ such that $\sum_{e \in Z} w_S(e) \leq k$, and an assignment φ that satisfies $S_\alpha - Z$ and sets $\varphi(s) = 1$ and $\varphi(t) = 0$. Then $(G_\alpha, \mathcal{B}_\alpha, s, k)$ is a yes-instance of* RBGCE.

Proof of claim: We claim that Z is a solution for $(G_\alpha, \mathcal{B}_\alpha, s, k)$. Suppose for a contradiction that this is not the case, i.e. there is a rigid cycle $C \notin \mathcal{B}_\alpha$ that is reachable from s in $G_\alpha - Z$. There are two cases. If $s \notin V(C)$, then since S_α is homogeneous, C is only satisfied by the all-zero assignment. But $\varphi(s) = 1$ and the nonzero value propagates to C, which contradicts $S_\alpha - Z$ being satisfied by φ. Otherwise, let P be the path resulting from deleting s from C. Let x_1 and x_2 be the endpoints of P. Let β be the result of multiplying the edge labels of P from x_1 to x_2. Since C is rigid, $\alpha(x_1) \cdot \beta \neq \alpha(x_2)$. Then, P is a path in $S_\alpha - Z$ incompatible with setting $\varphi(x_1) = \alpha(x_1)$ and $\varphi(x_2) = \alpha(x_2)$, which is again a contradiction. ◇

Claim 43.2. *Suppose there exists a subset Z of edges of G_α such that the connected component of the vertex s in $G_\alpha - Z$ is balanced. Then $S_\alpha - Z$ admits a satisfying assignment φ such that $\varphi(s) = 1$ and $\varphi(t) = 0$.*

Proof of claim: Let H be the connected component of s in $G_\alpha - Z$. Since edges connecting t, t', t'' form a high-weight rigid cycle, $t \notin V(H)$. Since every cycle of H is flexible, H viewed as a subset of S_α is flexible. Then in particular there is a satisfying assignment φ to H setting $\varphi(s) = 1$ by Corollary 35. For every variable v not in H we can safely set $\varphi(v) = 0$ since S_α is homogeneous. Then φ satisfies $S_\alpha - Z$ and sets $\varphi(s) = 1$ and $\varphi(t) = 0$. ◇

Together, Claims 43.1 and 43.2 imply that the assignment α can be extended to S so that it leaves equations of total weight at most k unsatisfied if and only if $(G_\alpha, \mathcal{B}_\alpha, s, k)$ is a yes-instance of RBGCE. Note that the algorithm considers all satisfying assignments to X so by exhaustion (S, w_S, k, X) is a yes-instance if and only if the algorithm finds a suitable assignment α. There are p^{k+1} candidates for α. Computing the instance $(G_\alpha, \mathcal{B}_\alpha, s, k)$ requires polynomial time and RBGCE can be solved in $O^*(2^k)$-time by Proposition 3 so the total running time is $O^*((2p)^k)$. □

6 HARDNESS RESULTS

Let $\mathbb{D} = (D; +, \cdot)$ be a commutative ring. The reduction from Multicut presented in the introduction shows that Min-r-Lin($\mathbb{D}$) is NP-hard (for $r \geq 2$). In Section 6.1, we show W[1]-hardness for $r \geq 3$ whenever $(D; +)$ is an Abelian group with at least two elements. This result consequently covers all (commutative and non-commutative) rings except

the trivial zero ring. We continue in Section 6.2 by studying Min-2-Lin($\mathbb{D}$) for commutative rings $\mathbb{D}$ that contain a zero divisor (i.e. an element $\alpha \neq 0$ such that there exists an element $\beta \neq 0$ and $\alpha \cdot \beta = 0$). We show that Min-2-Lin($\mathbb{D}$) is W[1]-hard for many such structures. We note that hardness results for certain special cases have appeared earlier in the literature—for instance, Crowston et al. [13] prove W[1]-hardness for Min-3-Lin($\mathbb{F}_2$).

For proving the hardness results, we use *parameterized reductions* (or fpt-reductions). Consider two parameterized problems $L_1, L_2 \subseteq \Sigma^* \times \mathbb{N}$. A mapping $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ is a parameterized reduction from L_1 to L_2 if

(1) $(x, k) \in L_1$ if and only if $P((x, k)) \in L_2$,
(2) the mapping can be computed in $f(k) \cdot n^{O(1)}$-time for some computable function f, and
(3) there is a computable function $g : \mathbb{N} \to \mathbb{N}$ such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$, if $(x', k') = P((x, k))$, then $k' \leq g(k)$.

The class W[1] consists of all problems that are fpt-reducible to Independent Set parameterized by the solution size, i.e. the number of vertices in the independent set. Showing W[1]-hardness (by an fpt-reduction) for a problem rules out the existence of an fpt algorithm under the standard assumption that FPT $\neq$ W[1].

6.1 Three Variables per Equation

Let $\mathbb{G} = (D; +)$ denote an arbitrary Abelian group. An expression $x_1 + \cdots + x_r = c$ is an *equation over* $\mathbb{G}$ if $c \in D$ and $x_1, \ldots, x_r$ are either variables or inverted variables with domain D. We say that it is an *r-variable equation* if it contains at most r distinct variables. We consider the following group-based variant of the Min-r-Lin($\mathbb{D}$) problem.

Min-r-Lin($\mathbb{G}$)	
Instance:	A system S of equations over $\mathbb{G}$, a weight function $w : S \to \mathbb{N}^+$, and an integer k.
Parameter:	k.
Question:	Is there a set $Z \subseteq S$ such that $S - Z$ is consistent and $w(Z) \leq k$?

The crux of the proof is essentially the same as the W[1]-hardness proof for Odd Set presented in Theorem 13.31 of [14, Section 13.6.3], however many details are different. The reduction is based on the following W[1]-hard problem [23, Lemma 1].

Multicoloured Clique	
Instance:	A graph $G = (V, E)$ with vertices partitioned into k colour classes $V_1, \ldots, V_k$.
Parameter:	k.
Question:	Does G contain a clique with exactly one vertex from each V_i, $1 \leq i \leq k$?

Theorem 44. *Let $\mathbb{G} = (D; +)$ denote a group with at least two elements. Then, Min-r-Lin($\mathbb{G}$) is W[1]-hard for any $r \geq 3$ even if all equations have weight 1.*

Proof. The reduction is presented in two steps: given an arbitrary instance $(G, k, (V_1, \ldots, V_k))$ of Multicoloured Clique, we first compute an instance (S, w, k') of Min-s-Lin($\mathbb{G}$) where $s = |V(G)| + |E(G)|$, and then we transform this instance into an instance of Min-3-Lin($\mathbb{G}$) with unit weights. We let 0 denote the identity element in $\mathbb{G}$ and let 1 be any non-identity element.

Step 1. Consider the arbitrarily chosen instance $(G, k, (V_1, \ldots, V_k))$ of Multicoloured Clique. We will now reduce it to an instance of Min-s-Lin($\mathbb{G}$). We let E_{ij} denote the set of edges in $E(G)$ with one endpoint in V_i and another in V_j, and we let E_{ijv} be the subset of E_{ij} containing all edges incident to a vertex v. We define an instance (S, w, k') of

Min-s-Lin($\mathbb{G}$) as follows. Introduce variables x_v for all $v \in V(G)$ and y_e for all $e \in E(G)$. Set the parameter $k' = k + \binom{k}{2}$. Let S contain the following equations:

(1) $x_v = 0$ for all $v \in V(G)$.
(2) $y_e = 0$ for all $e \in E(G)$.
(3) $\sum_{v \in V_i} x_v = 1$ for all $1 \leq i \leq k$.
(4) $\sum_{e \in E_{ij}} y_e = 1$ for all $1 \leq i < j \leq k$.
(5) $\sum_{e \in E_{ijv}} y_e = x_v$ for all $v \in V(G)$ and $1 \leq i < j \leq k$.

The equations in (3)–(5) are assigned weight $k' + 1$, while all others are given unit weight. Thus, only equations in (1) and (2) may appear in a solution to (S, w, k'). Observe that the equations in (1)–(4) imply that exactly one variable in $\{x_v \mid v \in V_i\}$ for each $1 \leq i \leq k$ and one variable in $\{y_e \mid e \in E_{ij}\}$ for each pair $1 \leq i < j \leq k$ may be set to 1 since the budget $k + \binom{k}{2}$ is tight.

Now consider the equations in (5). If x_v is set to 1, then, for every $1 \leq i < j \leq k$, there is an edge $e \in E_{ij}$ incident to v such that y_e is set to 1. Otherwise, if x_v is set to 0, then y_e is set to 0 for every edge e incident to v. Thus, if φ is an assignment to S that breaks constraints of total weight k', then the set of vertices $\{v \in V(G) : \varphi(x_v) = 1\}$ forms a clique in G. In the opposite direction, if $X \subseteq V(G)$ forms a clique in G, then an assignment that sets x_v to 1 if $v \in X$, y_e to 1 if both endpoints of e are in X, and all other variables to 0 breaks constraints of total weight k'. We conclude that the reduction is correct and it can clearly be carried out in polynomial time.

Step 2. We continue by transforming the instance (S, w, k') into an instance of Min-3-Lin($\mathbb{G}$) with unit weights. Consider an equation $\sum_{i=1}^{r} v_i = 1$ in S. We first show how to make it undeletable without assigning it the weight $k' + 1$. To this end, introduce variables $v_i^{(j)}$ for all $1 \leq i \leq r$ and $1 \leq j \leq k' + 2$. Create a system of equations S' by adding equations $\sum_{i=1}^{r} v_i^{(j)} = 1$ for all j and equations $v_i^{(j)} - v_i^{(j')} = 0$ for all i and all $j < j'$. We claim that any assignment that does not set $v_i^{(1)}, \ldots, v_i^{(k'+2)}$ to the same value does not satisfy at least $k' + 1$ constraints. Suppose an assignment sets ℓ copies of the variable to one value and $k' + 2 - \ell$ remaining copies to another. Then at least $(k' + 2 - \ell)\ell$ equations are not satisfied by the assignment. For $1 \leq \ell \leq k'$, this quantity is minimised by $\ell = 1$ and it equals $k' + 1$. Thus, any assignment that does not satisfy at most k' constraints also satisfies S'.

Finally, we show how to reduce the number of variables in each equation to at most three. Again, consider an equation of the form $\sum_{i=1}^{r} v_i = 1$. Introduce auxiliary variables a_i for $i \in \{1, \ldots, r\}$ and replace the equation with the following system:

$$\begin{cases} v_1 + (-a_1) = 0, \\ a_i + v_{i+1} + (-a_{i+1}) = 0 \quad \text{for } i \in \{1, \ldots, r-1\} \\ a_r + v_r = 1 \end{cases}$$

where each equations is given weight 1. Observe that the sum of all equations above telescopes and the auxiliary variables cancel out, leaving exactly the equation $\sum_{i=1}^{r} v_i = 1$. Hence, an assignment that satisfies all equations in the system also satisfies the original equation. Moreover, any assignment φ that does not satisfy the original equation can be extended to the auxiliary variables to satisfy all but one equation by setting $\varphi(v_1) = \varphi(a_1)$ and $\varphi(v_{i+1}) = \varphi(a_i) + \varphi(v_i)$ for all $i \in \{1, \ldots, r-1\}$. Hence, replacing every long equation in this way reduces the initial instance to an instance of Min-3-Lin($\mathbb{G}$) with unit weights. □

6.2 Rings with Zero Divisors

Recall that an integral domain does not contain zero divisors. Next, we give examples of commutative rings $\mathbb{D}$ with zero divisors such that Min-2-Lin($\mathbb{D}$) is W[1]-hard. Our starting point is the following problem.

Paired Min Cut	
Instance:	A graph G, vertices $s, t \in V(G)$, an integer k, and a set of disjoint edge pairs $C \subseteq \binom{E(G)}{2}$
Parameter:	k.
Question:	Is there an st-mincut $X \subseteq E(G)$ which is the union of k pairs from C?

Paired Min Cut is commonly used as a source problem for proving W[1]-hardness (see e.g. [17, 37, 46, 48]). The W[1]-hardness of this problem follows from work by Marx and Razgon [46, Theorem 7] but they formulate the problem in logical terms. A graph-theoretic formulation, slightly different from the above, can be found in [38, Lemma 5.7]. In particular, Kim et al. [38] assume the graph to be a DAG, however, W[1]-hardness of Paired Min Cut as defined above follows easily from this.

We will consider a restricted variant of Paired Min Cut in the following. We say that an instance of Paired Min Cut is *split* if the following statements hold.

(1) There are two induced subgraphs $G_1 = G[U_1]$ and $G_2 = G[U_2]$ of G such that $U_1 \cup U_2 = V(G)$, $U_1 \cap U_2 = \{s, t\}$ and $G - \{s, t\}$ is the disjoint union of $G_1 - \{s, t\}$ and $G_2 - \{s, t\}$
(2) For every pair $\{e_1, e_2\} \in C$, one edge lies in G_1 and the other lies in G_2

Lemma 45. *Paired Min Cut is W[1]-hard, even for split instances.*

Proof. We established above that Paired Min Cut is W[1]-hard in its standard form. We show that we can also impose the split property. Thus, let $I = (G, s, t, k, C)$ be an arbitrary instance of Paired Min Cut. We construct an instance $I' = (G', s, t, k', C')$ of Paired Min Cut where I' is split and $k' = 4k$.

Create two graphs G_1 and G_2 on disjoint vertex sets, each a copy of G, and let G' be their union. For every edge $e = \{u, v\}$ in G', introduce a new vertex x_e and the two edges $e' = \{u, x_e\}$ and $e'' = \{x_e, v\}$. For an edge or vertex z of G and $i \in \{1, 2\}$, let z_i denote the copy of z in G_i. For every pair $p = \{e, f\}$ in C, place the four pairs

$$\{e_1, e_2\}, \{e_1', f_2\}, \{f_1, e_2'\}, \{f_1', f_2'\}$$

in C' (thereby keeping the pairs in C' disjoint). Finally, identify s_1 with s_2 as s and t_1 with t_2 as t. This finishes the description of our output $I' = (G', s, t, k', C')$. Note that G' is split, and that the st-max flow in G' is $8k = 2k'$.

We show that I is a yes-instance if and only if I' is a yes-instance. First, let $X \subseteq E(G)$ be a solution to I. Let $X' = \{e_1, e_1', e_2, e_2' \mid e \in X\}$. Then X' is the union of precisely four pairs for every pair in X, and it is clear that X' is an st-cut.

On the other hand, assume that I' has a solution $X' = X_1' \cup X_2'$ (where $X_i' \subseteq E(G_i)$, $i \in \{1, 2\}$). We claim that X_1' and X_2' represent the same edge set X in G. By assumption, X' partitions into edge pairs, and since the st-max flow in G is $2k'$, X' must be an st-min cut. In particular, by the structure of the pairs, for every $e \in E(G)$, X' contains e if and only if it contains e', and therefore also the other endpoint of the pair $p' \in C'$ containing the respective edge. Hence for every edge e represented in X' there must be a pair $\{e, f\} \in C$ such that all four pairs $\{e_1, f_1\} \times \{e_2, f_2\}$ are represented in X'. Hence

$$X = \{e \in E(G) \mid \{e_1, e_1', e_2, e_2'\} \subseteq X'\}.$$

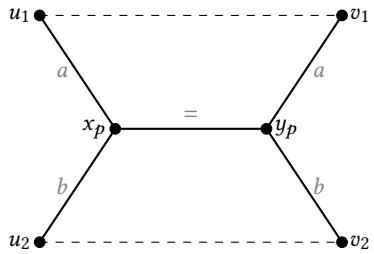


Fig. 8. System of equations obtained from a pair of edges $p = \{e_1, e_2\}$ where $e_i = \{u_i, v_i\}$ in the fpt reduction from Paired Min Cut. Edges e_1 and e_2 are illustrated by dashed lines, while the equations are illustrated by solid lines with labels describing equations between connected variables.

defines a set of $2k$ edges in G, which partitions into pairs from C. Furthermore, X is an st-cut, since X'_1 is an st-cut in G_1 and G_1 was created as a copy of G. □

Let $\mathbb{D}$ denote a commutative ring. For $a \in \mathbb{D}$, let the *annihilator* of a be the set $\mathrm{Ann}(a) = \mathrm{Ann}(a) = \{x \in \mathbb{D} \mid ax = 0\}$.

Theorem 46. *If a commutative ring $\mathbb{D}$ contains two elements a, b such that $\mathrm{Ann}(a)$ and $\mathrm{Ann}(b)$ are incomparable under set inclusion, then Min-2-Lin($\mathbb{D}$) is W[1]-hard.*

Proof. Let $f \in \mathrm{Ann}(a) \setminus \mathrm{Ann}(b)$ and $g \in \mathrm{Ann}(b) \setminus \mathrm{Ann}(a)$. We present the reduction from Paired Min Cut on split instances. Let (G, s, t, k, C) be an arbitrary split instance, and assume subsets $U_1, U_2 \subseteq V(G)$ form the split. Construct instance (S, k) of Min-2-Lin($\mathbb{D}$) as follows. Start by adding one variable to S for every vertex in $V(G) \setminus \{s, t\}$ with the same name. Create variables s_1, s_2 for $s \in V(G)$ and t_1, t_2 for $t \in V(G)$. Add equations $s_1 = ag$, $s_2 = bf$, $t_1 = t_2 = 0$ of weight $k + 1$ for the sources, and equations $u = v$ of weight $k + 1$ for all edges $uv \in E(G)$ not present in any pair of C. For every pair $p = (u_1v_1, u_2v_2) \in C$, we introduce two auxiliary vertices x_p and y_p, and create a gadget with equation $x_p = y_p$ and equations

$$u_1 = ax_p \qquad v_1 = ay_p,$$
$$u_2 = bx_p \qquad v_2 = by_p$$

of weight $k + 1$. See Figure 8 for an illustration. This completes the construction.

For one direction, assume $X \subseteq C$ is a solution to (G, s, t, k, C). Define the set of equations $X' = \{x_p = y_p \mid p \in X\}$ of the same size. We argue that $S - X$ is consistent under assignment α defined as follows. For the primary variables, let

$$\alpha(u) = \begin{cases} ag & \text{if } v \in U_1 \text{ and } s \text{ reaches } v \text{ in } G - \bigcup X, \\ bf & \text{if } v \in U_2 \text{ and } s \text{ reaches } v \text{ in } G - \bigcup X, \\ 0 & \text{if } v \in U_1 \cup U_2 \text{ and } t \text{ reaches } v \text{ in } G - \bigcup X. \end{cases}$$

Since $\bigcup X$ is an st-cut, this assignment is well-defined. Now extend it to auxiliary variables as $\alpha(x_p) = f(\alpha(u_1), \alpha(u_2))$ and $\alpha(y_p) = f(\alpha(v_1), \alpha(v_2))$, where the function $f : \mathbb{D}^2 \to \mathbb{D}$ is defined as

w_1	w_2	$f(w_1, w_2)$
0	0	0
0	bf	f
ag	0	g
ag	bf	$f + g$.

It is straightforward to verify that α satisfies equations $u_1 = ax_p$, $v_1 = ay_p$, $u_2 = bx_p$, and $v_2 = by_p$. Now suppose $p = (u_1v_1, u_2v_2) \notin C$. Then edges u_1v_1 and u_2v_2 are present in $G - \bigcup X$, hence $\alpha(u_1) = \alpha(v_1)$ and $\alpha(u_2) = \alpha(v_2)$. By definition, we obtain $\alpha(x_p) = \alpha(y_p)$, therefore α satisfies $S - X'$.

For the other direction, suppose Z is a solution to (S, k). Note that every equation except those in $S_C = \{x_p = y_p \mid p \in C\}$ has weight $k+1$. Hence, we may assume without loss of generality that $Z \subseteq S_C$ and define $Z' = \{p \in C \mid x_p = y_p \in Z\}$. We claim that Z' is a solution to (G, s, t, k, C). By definition, Z' is a union of $|Z| \leq k$ pairs in C so it remains to show that Z' is an st-cut in G. Observe that if $x_p = y_p$ is in $S - Z$ for some $p = (u_1v_1, u_2v_2)$, then the equations in $S - Z$ imply that $u_1 = ax_p = v_1$ and $u_2 = by_p = v_2$. Hence, if there is an st-path in $G - Z'$, then the equations in $S - Z$ imply that $s_1 = t_1$ or $s_2 = t_2$, which contradicts the assignments $s_1 = ag$, $s_2 = bf$ and $t_1 = t_2 = 0$. □

This result implies that if $\mathbb{D}$ is a commutative ring, then the annihilators of $\mathbb{D}$ must be totally ordered under set inclusion if Min-2-Lin(R) is not W[1]-hard. Such rings are called *lineal* [44]. We illustrate Theorem 46 with an example. The *direct product* of two rings $\mathbb{D}_1 = (D_1; +_1, \cdot_1)$ and $\mathbb{D}_2 = (D_2; +_2, \cdot_2)$ is denoted $\mathbb{D}_1 \times \mathbb{D}_2 = (D; +, \cdot)$. Its domain D consists of the ordered pairs $\{(d_1, d_2) \mid d_1 \in D_1, d_2 \in D_2\}$ and the operations are defined coordinate-wise: $(d_1, d_2) + (d_1', d_2') = (d_1 +_1 d_1', d_2 +_2 d_2')$ and $(d_1, d_2) \cdot (d_1', d_2') = (d_1 \cdot_1 d_1', d_2 \cdot_2 d_2')$. We claim that whenever $\mathbb{D} = \mathbb{D}_1 \times \mathbb{D}_2$ and $\mathbb{D}_1, \mathbb{D}_2$ are commutative rings that are not zero rings, then Min-2-Lin($\mathbb{D}$) is W[1]-hard. To see this, we first note that the following: if a, b in $\mathbb{D}$ satisfies $ab = 0$, $a^2 \neq 0$, and $b^2 \neq 0$, then Ann(a) and Ann(b) are incomparable since $b \in \text{Ann}(a)$, $a \notin \text{Ann}(a)$, $b \notin \text{Ann}(b)$, and $a \in \text{Ann}(b)$. Let $0_1 \in D_1, 0_2 \in D_2$ denote the additive identities and $1_1 \in D_1, 1_2 \in D_2$ denote the multiplicative identities. By setting $a = (0_1, 1_2)$ and $b = (1_1, 0_2)$, we see that $ab = 0$, $a^2 \neq 0$, and $b^2 \neq 0$ so Theorem 46 is applicable and Min-2-Lin($\mathbb{D}$) is W[1]-hard. This argument can easily be extended to products of several commutative rings. As an example, note that the ring $\mathbb{Z}/m\mathbb{Z}$ (i.e. the ring based on standard arithmetic modulo m) is isomorphic to a direct product of non-trivial commutative rings whenever m is not a prime power. This is a direct consequence of the Chinese Remainder Theorem. Hence, Min-2-Lin($\mathbb{Z}/6\mathbb{Z}$) = Min-2-Lin($\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$) and more generally Min-2-Lin($\mathbb{Z}/m\mathbb{Z}$) where m is not a prime power are W[1]-hard problems.

7 CONCLUSIONS AND DISCUSSION

We have proved that Min-2-Lin($\mathbb{D}$) is fixed-parameter tractable (with parameter k being the number of unsatisfied equations) when $\mathbb{D}$ is a commutative domain having Helly dimension at most 2 and the representation of $\mathbb{D}$ satisfies some mild computational restrictions. We have presented substantially faster algorithms when $\mathbb{D}$ is a field. We additionally proved that Min-r-Lin($\mathbb{D}$) is W[1]-hard when $r \geq 3$ and this result holds for all rings. Furthermore, we demonstrated that there exist (even finite) commutative rings $\mathbb{D}$ such that Min-2-Lin($\mathbb{D}$) is W[1]-hard. We discuss a selection of possible research directions below.

FPT/W[1]-hardness borderline. This boundary is not clear for Min-2-Lin problems, and this is true even for finite commutative rings. Theorem 40 states that if $\mathbb{D}$ is a finite ring, then either (1) $\mathbb{D}$ is a field (and Min-2-Lin($\mathbb{D}$) is in FPT) or (2) $\mathbb{D}$ contains zero divisors. We know that there are $\mathbb{D}$ with zero divisors (such as $\mathbb{Z}/6\mathbb{Z}$) where Min-2-Lin($\mathbb{D}$) is

W[1]-hard, but it is an open question whether the problem is always W[1]-hard when $\mathbb{D}$ contains zero divisors, even in the finite case. A concrete question is the following: what is the parameterized complexity of Min-2-Lin($\mathbb{Z}/4\mathbb{Z}$) or more generally Min-2-Lin($\mathbb{Z}/p^n\mathbb{Z}$) where p is a prime and $n \geq 2$? These rings are *chain rings*, i.e. rings where the ideals are totally ordered under set inclusion. Since every annihilator is an ideal, it follows that these rings are lineal and W[1]-hardness cannot be inferred from Theorem 46. Resolving these cases would give us a complete understanding of Min-2-Lin($\mathbb{Z}/m\mathbb{Z}$) for every m. However, there are still many open cases left, even for small commutative rings.

If we turn our attention to general commutative rings, then a natural next step would be to analyse the rings in H_3. To the best of our knowledge, there is no concrete example of a ring $\mathbb{D} \in \mathsf{H}_m$ with $m > 2$ where it is known if Min-2-Lin($\mathbb{D}$) is in FPT or if it is W[1]-hard. In particular, this holds true for the ring $\mathbb{Z}[X]$ that we encountered in Section 4.2. Rings of this kind have more complex minimal obstructions to satisfiability than paths and cycles, so our method for graph cleaning is no longer applicable and a fundamentally different approach may be necessary. We also note that the family of *k-independence* properties introduced in [12] is based on a concept that is similar to Helly dimension. Several general NP-hardness results for CSPs based on constraint languages that do not exhibit certain 2-independence properties are proved in [8]. Even though the setting is quite different, the idea that Min-2-Lin($\mathbb{D}$) for certain $\mathbb{D} \in \mathsf{H}_3$ may be computationally hard seems conceivable. This points in the direction that it may be a too large step to attack rings with higher Helly dimension directly. With this in mind, it is probably a better idea to first analyse more well-behaved classes of commutative domains such as unique factorisation domains.

Faster algorithms. We suspect that our fixed-parameter tractable algorithm for Min-2-Lin can be improved with respect to running time. The slowest part in it is solving the Pair Partition Cut problem. We solve this problem via a reduction to a finite-domain MinCSP problem that is solved by flow augmentation, but there may be alternative ways of doing this. However, as the problem is a strict generalization of (Edge) Multicut, a running time of, say, $O^*(2^{O(k \log k)})$ or better would be a significant challenge. There is also room for improvements in the Min-2-Lin algorithms for fields. Consider our $O^*(2^{O(k \log k)})$-time algorithm for arbitrary fields. After iterative compression and cleaning, the problem reduces to the following:

2-Lin($\mathbb{F}$) Compatibility	
Instance:	Two instances S_1, S_2 of 2-Lin($\mathbb{F}$) and an integer k such that $V(S_1) \subseteq V(S_2)$, $\lvert S_1\rvert \leq 3k$, and S_2 only contains equalities.
Parameter:	k.
Question:	Is there a set $Z \subseteq S_2$ such that $\lvert Z\rvert \leq k$ and $(S_1 \cup S_2) - Z$ is consistent?

This problem is the bottleneck for our Min-2-Lin($\mathbb{F}$) algorithm, since it is the only part that requires more than single-exponential time. Can it be solved in single-exponential time in k? For the finite field $\mathbb{F}_p$ with p elements we show that Min-2-Lin($\mathbb{F}_p$) can be solved in $O^*((2p)^k)$ time. Is there an $O^*(c^k)$ algorithm for Min-2-Lin($\mathbb{F}_p$), where c is a universal constant that does not depend on p? Or is there at least a constant $d < 2$ such that Min-2-Lin($\mathbb{F}_p$) is solvable in $O^*((dp)^k)$ time?

Important balanced subgraphs. A more general question concerns the utility of the method of important balanced subgraphs. Important separators are a key component of many classical fpt-algorithms for graph separation problems, and important balanced subgraphs appear to be a significant, and unexpected, generalization of them. It would be interesting to see more applications of the method. We have used it to avoid random sampling of important separators, speeding up our Min-2-Lin algorithm for fields from $O^*(2^{O(k^3)})$ to $O^*(2^{O(k \log k)})$, and simplifying the algorithm

for rings in H_2. What other problems can be solved using this method? Can we use it to obtain simpler algorithms or improve upper bounds for other parameterized deletion problems? Other questions include generalizations or improvements on the method of important balanced subgraphs itself. We have provided the result only for edge deletion problems; is there an equivalent statement for vertex deletion? Furthermore, the polynomial factor in the running time of the algorithm producing a dominating family is significant, since it comes from solving an LP given only oracle access to the constraints. However, the optima computed by the LP are extremal half-integral solutions with an inherent structure that can probably be exploited. A combinatorial method for computing such optima could substantially improve the polynomial factor. Such a result was developed in the algorithm for 0/1/all CSPs by Iwata et al. [31], where a previous method based on half-integral LP-relaxations was replaced by a linear-time combinatorial solver. Can a similar method be developed for the Rooted Biased Graph Cleaning problem, perhaps for special cases such as biased graphs coming from group-labelled graphs or the biased graphs used in the Min-2-Lin($\mathbb{D}$) algorithm?

ACKNOWLEDGEMENTS

A preliminary version of this article appeared in the proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA-2023) [18]. We thank the anonymous reviewers for suggestions and comments that helped simplify some of our proofs. The second and the fourth authors were supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. In addition, the second author was partially supported by the Swedish Research Council (VR) under grants 2017-04112 and 2021-04371, and the fourth author was supported by VR under grant 2024-00274. The first and third authors acknowledge support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

REFERENCES

[1] Michael Artin. 2011. *Algebra.* Pearson.
[2] Vikraman Arvind and T. C. Vijayaraghavan. 2010. Classifying Problems on Linear Congruences and Abelian Permutation Groups Using Logspace Counting Classes. *Computational Complexity* 19, 1 (2010), 57–98.
[3] Imre Bárány and Gil Kalai. 2022. Helly-type problems. *Bulletin of the AMS* 59, 4 (2022), 471–502.
[4] Silvana Bazzoni and Sarah Glaz. 2006. Prüfer Rings. In *Multiplicative Ideal Theory: a Tribute to the Work of Robert Gilmer*. Springer, New York, 55–72.
[5] Kristóf Bérczi, Alexander Göke, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. 2022. Resolving Infeasibility of Linear Systems: A Parameterized Approach. *CoRR* abs/2209.02017 (2022), 46 pages. https://doi.org/10.48550/ARXIV.2209.02017 arXiv:2209.02017
[6] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. 2018. Multicut is FPT. *SIAM J. Comput.* 47, 1 (2018), 166–207.
[7] Alain Bouvier. 1980. *Survey on Locally Factorial Krull Domains.* Technical Report. Département de Mathématiques Lyon.
[8] Mathias Broxvall, Peter Jonsson, and Jochen Renz. 2002. Disjunctions, independence, refinements. *Artificial Intelligence* 140, 1/2 (2002), 153–173.
[9] Jianer Chen, Yang Liu, and Songjian Lu. 2009. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica* 55, 1 (2009), 1–13.
[10] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, and Dániel Marx. 2015. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms* 11, 4 (2015), 1–28.
[11] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. 2016. Designing FPT Algorithms for Cut Problems Using Randomized Contractions. *SIAM J. Comput.* 45, 4 (2016), 1171–1229.
[12] David A. Cohen, Peter Jeavons, Peter Jonsson, and Manolis Koubarakis. 2000. Building tractable disjunctive constraints. *Journal of the ACM* 47, 5 (2000), 826–853.
[13] Robert Crowston, Gregory Gutin, Mark Jones, and Anders Yeo. 2013. Parameterized Complexity of Satisfying Almost All Linear Equations over $\mathbb{F}_2$. *Theory of Computing Systems* 52, 4 (2013), 719–728.
[14] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. 2015. *Parameterized Algorithms.* Springer.
[15] Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. 2016. On group feedback vertex set parameterized by the size of the cutset. *Algorithmica* 74, 2 (2016), 630–642.
[16] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. 2013. On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory* 5, 1 (2013), 1–11.

[17] Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, Marcin Pilipczuk, and Roohani Sharma. 2023. Parameterized Complexity Classification for Interval Constraints. In *Proc. 18th International Symposium on Parameterized and Exact Computation (IPEC-2023)*. 11:1–11:19.

[18] Konrad K. Dabrowski, Peter Jonsson, Sebastian Ordyniak, George Osipov, and Magnus Wahlström. 2023. Almost consistent systems of linear equations. In *Proc. 34th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2023)*. 3179–3217.

[19] Reinhard Diestel. 2016. *Graph Theory*. Springer.

[20] Mátyás Domokos. 2007. Typical separating invariants. *Transformation Groups* 12, 11 (2007), 49–63.

[21] Mátyás Domokos and Endre Szabó. 2011. Helly dimension of algebraic groups. *Journal of the London Mathematical Society* 84, 1 (2011), 19–34.

[22] Jesse Elliott. 2019. *Rings, Modules, and Closure Operations*. Springer.

[23] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. 2009. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science* 410, 1 (2009), 53–61.

[24] Alexander Göke, Lydia Mirabel Mendoza Cadena, and Matthias Mnich. 2019. Resolving infeasibility of linear systems: A parameterized approach. In *Proc. 14th International Symposium on Parameterized and Exact Computation (IPEC-2019)*. 17:1–17:15.

[25] Joseph F. Grcar. 2011. How ordinary elimination became Gaussian elimination. *Historia Mathematica* 38, 2 (2011), 163–218.

[26] Sylvain Guillemot. 2011. FPT algorithms for path-transversal and cycle-transversal problems. *Discrete Optimization* 8, 1 (2011), 61–71.

[27] Johan Håstad. 2001. Some optimal inapproximability results. *J. ACM* 48, 4 (2001), 798–859.

[28] Israel N. Herstein. 1961. Wedderburn's Theorem and a theorem of Jacobson. *The American Mathematical Monthly* 68, 3 (1961), 249–251.

[29] Semba Ichiro. 1984. An Efficient Algorithm for Generating all Partitions of the Set $\{1, 2, \ldots, n\}$. *Inform. Process. Lett.* 7, 1 (1984), 41–42.

[30] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. 2016. Half-integrality, LP-branching, and FPT algorithms. *SIAM J. Comput.* 45, 4 (2016), 1377–1411.

[31] Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 2018. 0/1/all CSPs, half-integral A-path packing, and linear-time FPT algorithms. In *Proc. IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS-2018)*. 462–473.

[32] Bart M.P. Jansen, Jari J.H. de Kroon, and Michał Włodarczyk. 2025. Single-exponential FPT algorithms for enumerating secluded $\mathcal{F}$-free subgraphs and deleting to scattered graph classes. *J. Comput. System Sci.* 148 (2025), 103597.

[33] Ravindran Kannan. 1985. Solving systems of linear equations over polynomials. *Theoretical Computer Science* 39 (1985), 69–88.

[34] Ravindran Kannan and Achim Bachem. 1979. Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix. *SIAM J. Comput.* 8, 4 (1979), 499–507.

[35] Subhash Khot. 2002. On the power of unique 2-prover 1-round games. In *Proc. 24th Annual ACM Symposium on Theory of Computing (STOC-2002)*. 767–775.

[36] Subhash Khot and Dana Moshkovitz. 2016. Candidate hard unique game. In *Proc. 48th Annual ACM Symposium on Theory of Computing (STOC-2016)*. 63–76.

[37] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2021. Solving hard cut problems via flow-augmentation. In *Proc. 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2021)*. 149–168.

[38] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2022. Flow-augmentation III: Complexity dichotomy for Boolean CSPs parameterized by the number of unsatisfied constraints. *CoRR* abs/2207.07422 (2022), 60 pages.

[39] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. 2023. Flow-augmentation III: Complexity dichotomy for Boolean CSPs parameterized by the number of unsatisfied constraints. In *Proc. 2023 ACM-SIAM Symposium on Discrete Algorithms (SODA-2023)*. 3218–3228.

[40] Eun Jung Kim, Tomás Masarík, Marcin Pilipczuk, Roohani Sharma, and Magnus Wahlström. 2024. On Weighted Graph Separation Problems and Flow Augmentation. *SIAM Journal on Discrete Mathematics* 38, 1 (2024), 170–189.

[41] Euiwoong Lee and Magnus Wahlström. 2020. LP-branching algorithms based on biased graphs. *CoRR* abs/1610.06060v2 (2020), 22 pages. Updated and extended version of (Wahlström, SODA-2017)..

[42] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. 2014. Faster Parameterized Algorithms Using Linear Programming. *ACM Transactions on Algorithms* 11, 2 (2014), 15:1–15:31.

[43] Daniel Lokshtanov and M. S. Ramanujan. 2012. Parameterized Tractability of Multiway Cut with Parity Constraints. In *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP-2012)*. 750–761.

[44] Greg Marks and Ryszard Mazurek. 2016. Rings with linearly ordered right annihilators. *Israel Journal of Mathematics* 216 (2016), 415–440.

[45] Dániel Marx. 2006. Parameterized graph separation problems. *Theoretical Computer Science* 351, 3 (2006), 394–406.

[46] Dániel Marx and Igor Razgon. 2009. Constant ratio fixed-parameter approximation of the edge multicut problem. *Inform. Process. Lett.* 109, 20 (2009), 1161–1166.

[47] Dániel Marx and Igor Razgon. 2014. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM J. Comput.* 43, 2 (2014), 355–388.

[48] George Osipov and Magnus Wahlström. 2023. Parameterized Complexity of Equality MinCSP. In *Proc. 31st Annual European Symposium on Algorithms (ESA-2023)*. 86:1–86:17.

[49] Marcin Pilipczuk, Michał Pilipczuk, and Marcin Wrochna. 2019. Edge Bipartization Faster than 2^k. *Algorithmica* 81, 3 (2019), 917–966.

[50] Heinz Prüfer. 1932. Untersuchungen über Teilbarkeitseigenschaften in Körpern. *Journal für die Reine und Angewandte Mathematik* 168 (1932), 1–36.

[51] Bruce Reed, Kaleigh Smith, and Adrian Vetta. 2004. Finding odd cycle transversals. *Operations Research Letters* 32, 4 (2004), 299–301.

[52] Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

[53] Askar A. Tuganbaev. 2021. Arithmetical Rings. *Journal of Mathematical Sciences* 258, 2 (2021), 129–198.
[54] Magnus Wahlström. 2017. LP-branching algorithms based on biased graphs. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2017)*. 1559–1570.
[55] Mingyu Xiao. 2010. Simple and improved parameterized algorithms for multiterminal cuts. *Theory of Computing Systems* 46, 4 (2010), 723–736.
[56] Thomas Zaslavsky. 1989. Biased graphs. I. Bias, balance, and gains. *Journal of Combinatorial Theory, Ser. B* 47, 1 (1989), 32–52.