Almost Consistent Systems of Linear Equations[*]

Konrad K. Dabrowski[†] Peter Jonsson[‡] Sebastian Ordyniak[§] George Osipov[¶]
Magnus Wahlström[‖]

Abstract

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 minimizes 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 Euclidean domains—a family of commutative rings that generalize finite and infinite fields including the rationals, the ring of integers and many other structures. We show that if every equation contains at most two variables, the problem is fixed-parameter tractable. This generalizes 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 Euclidean domains. On the technical side, we introduce the notion of important balanced subgraphs, generalizing important separators of Marx [Theor. Comput. Sci. 2006] to the setting of biased graphs. Furthermore, we use recent results on parameterized MinCSP [Kim et al., SODA 2021] to efficiently solve a generalization of Multicut with disjunctive cut requests.

1 Introduction

Algorithms for systems of linear equations have been studied since ancient times [16]. As Håstad [19] aptly remarks, for computer science "[t]his problem is in many regards as basic as satisfiability". Well-known methods like Gaussian elimination can recognize and solve consistent systems of equations. However, these methods are not well suited for dealing with inconsistent systems. In the optimization version of the problem called MaxLin one seeks an assignment maximizing the number of satisfied equations. In its dual, called MinLin, the objective is to minimize 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 [27] and the following discussion). This motivates exploring other approaches to resolving inconsistent systems.

Crowston et al. [9] 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 admits a $\mathcal{O}^*(2^k)$ [1] algorithm, while 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 $\mathcal{O}^*(f(k))$ time for any computable function f under the standard assumption FPT $\neq$ W[1]. In this paper we substantially extend the study of the parameterized complexity of MinLin by considering equations over commutative rings. Thus, we study *Euclidean domains*, which include the finite fields $\mathbb{F}_q$, infinite fields such as the rationals $\mathbb{Q}$, the ring of integers $\mathbb{Z}$, the Gaussian integers $\mathbb{Z}[i]$, the ring of polynomials $\mathbb{F}[x]$ over a field $\mathbb{F}$

[*]The full version of the paper can be accessed at `https://arxiv.org/abs/2208.02732`

[†]Newcastle University, UK, `konrad.dabrowski@newcastle.ac.uk`
[‡]Linköping University, Sweden, `peter.jonsson@liu.se`
[§]University of Leeds, UK, `sordyniak@gmail.com`
[¶]Linköping University, Sweden, `george.osipov@pm.me`
[‖]Royal Holloway, University of London, UK, `Magnus.Wahlstrom@rhul.ac.uk`

[1]$\mathcal{O}^*$ hides polynomial factors in the bit-size of the instance.

Problem	Solution	Method	Reduces to
Bipartization	[37] in 2004	Iterative compression	Min-2-Lin($\mathbb{F}_2$)
q-Multiway Cut	[33] in 2006	Important separators (IS)	Min-2-Lin($\mathbb{F}_q$)
Multiway Cut	[33] in 2006	Important separators	Min-2-Lin($\mathbb{Q}$)
Multicut	[4], [35]† in 2011	†Random sampling of IS	Min-2-Lin($\mathbb{Z}$)

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

and many more structures. Perhaps unsurprisingly, we show that with three or more variables per equation, the problem over Euclidean domains is W[1]-hard (in fact, the hardness proof only uses coefficients 0, 1 and -1, so the result holds for equations over any 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 every effective Euclidean domain, i.e. those that admit representations such that the basic operations are polynomial-time computable and multiplication is well behaved (see Property 4.1 for details). Note that asking about the parameterized complexity of Min-2-Lin over a domain only makes sense if the problem of checking consistency of a system is not NP-hard (otherwise, the problem is intractable even for $k = 0$). This is where the effectiveness of Euclidean domains becomes important. To the best of our knowledge, there are no published algorithms for solving systems of equations over Euclidean domains in the literature even for the special case with at most two variables per equation. Thus, we develop methods for checking consistency of such systems in Section 4. These methods form the underpinning of our fpt algorithm for Min-2-Lin over Euclidean domains.

Background. We start 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 we 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 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 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 $w(Z) \leq k$?

Crowston et al. [9] 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 of progress).

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. [37] using the newly introduced method of *iterative compression*, which has since become a common opening of fpt algorithms including those presented in this paper (see Chapter 4 in [10] 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 at most k edges 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 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 [33] presented the first fpt algorithm for Multiway Cut,

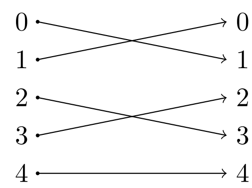


Figure 1: A permutation of $\{0, 1, 2, 3, 4\}$ not expressible as a linear equation.

which is based on *important separators*. This work was followed by a string of improvements [5, 42] including the approach based on linear programming [12, 17] 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 at most k edges from G to separate s_i from t_i for all i. This problem clearly generalizes Multiway Cut: a reduction may 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. [4] and Marx and Razgon [35]. The latter introduced the method of *random sampling of important separators*, also known as *shadow removal*.

Another problem related to Min-2-Lin is Unique Label Cover (ULC) [6, 22, 26]. 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. 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. 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. To connect this problem with Min-2-Lin, consider for example a field $\mathbb{F}$ and an equation $ax + by = c$ with $a, b, c \in \mathbb{F}$. 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 an immediate consequence, observe that Min-2-Lin($\mathbb{F}_q$) is in FPT for all finite fields $\mathbb{F}_q$, and can be solved in $\mathcal{O}^*(q^{2k})$ time using the best known algorithm for ULC [22, 23]. 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.

Results. We prove that Min-2-Lin($\mathbb{D}$), where $\mathbb{D}$ is an efficient Euclidean domain, is fixed-parameter tractable. For the special case when $\mathbb{D}$ is a field, we provide a faster $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ algorithm. Furthermore, if $\mathbb{D}$ is a finite, q-element field, we provide a $\mathcal{O}^*((2q)^k)$ algorithm improving upon the $\mathcal{O}^*(q^{2k})$ upper bound obtained by reduction to ULC. To complement the results, we show that Min-3-Lin($\mathbb{D}$) is W[1]-hard (ruling out the existence of fpt algorithms for Min-r-Lin($\mathbb{D}$) when $r \geq 3$) and we show that Min-2-Lin($\mathbb{K}$) is W[1]-hard for certain commutative rings $\mathbb{K}$ that are not Euclidean domains. For example, the hardness result holds if $\mathbb{K}$ is isomorphic to the direct product of nontrivial rings (such as the ring $\mathbb{Z}/6\mathbb{Z}$ of integers modulo 6).

Important balanced subgraphs. Our main technical contribution is the notion of *important balanced subgraphs*, which is a substantial generalisation of the *important separators* of Marx [33]. We believe that they can be applied to other problems as well, so we give a general explanation. 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 [40] 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

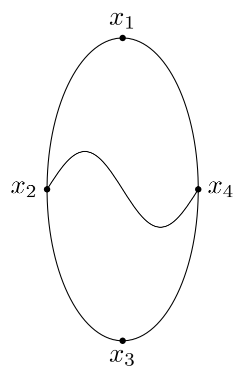


Figure 2: An example of a theta graph.

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 either 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 [43]. 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 maximizing the set of covered vertices and minimizing the weight. However, note that the 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, 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. We handle this by proving that there is a *dominating family* $\mathcal{H}$ of important balanced subgraphs such that $|\mathcal{H}| \leq 4^k$ and every balanced subgraph of G 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 note 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 2.3 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 $\mathcal{O}^*(4^k)$ time, both of which match the bounds for important separators [5, 33]. Moreover, the increased generality is crucial for our algorithms, since 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. Thus, 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.

Let us illustrate important balanced subgraphs using a few examples.

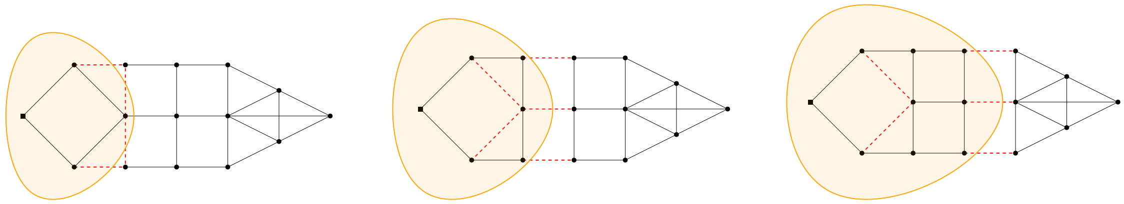
- Consider the biased graph $(G, \mathcal{B})$ defined above, where $\mathcal{B}$ contains the set of even cycles. Then a balanced subgraph of $(G, \mathcal{B})$ is precisely a bipartite graph, and our result outputs connected bipartite subgraphs containing the root vertex x.
- More generally, a *group-labelled graph* is a graph G in which every oriented edge of G is assigned a *group label* γ from a group Γ so that for any edge $\{u, v\} \in E(G)$, the labels observe $\gamma(uv) = \gamma(vu)^{-1}$. Let a cycle $C = (v_1, \dots, v_n)$ be *balanced* if the product $\gamma(v_1v_2)\gamma(v_2v_3)\dots\gamma(v_nv_1)$ 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. Thus the important balanced subgraph theorem can be used to, for example, output connected subgraphs where every cycle has a length which is a multiple of b for some $b \in \mathbb{N}$. This holds even if Γ is non-abelian.
- 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 [40] and Zaslavsky [43].

From a technical perspective, the result follows from a refined analysis of the LP formulation of Wahlström [40] for the problem of computing a minimum (vertex) transversal for the set of unbalanced cycles. Wahlström showed that this can be solved in $\mathcal{O}^*(4^k)$ 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" so that $V(H) \subseteq V(H')$ without increasing $c(H')$ (and such that the edges of value 1 in the LP are not contained in $E(H')$). The 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. [5] for the bound 4^k on the number of important separators.

Min-2-Lin Algorithms for Fields. In short, our fpt algorithms are based 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 $V(X)$. Finally, the problem reduces to computing a cut in the cleaned graph that fulfills certain requirements.

For a basic example, consider Min-2-Lin($\mathbb{Q}$) i.e. Min-2-Lin over the field of rationals. Every such instance can be viewed as a graph where an edge connecting two variables is labelled by an equation from S ranging over these two variables. 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. By standard linear algebra, a cycle may have zero, one, or infinitely many satisfying assignments—we call such cycles *inconsistent*, *non-identity* or *identity*, respectively. If an instance contains only identity 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.



(a) Subgraph H_1. (b) Subgraph H_2. (c) Subgraph H_3.

Figure 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 orange 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.

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. Since $|V(X)| \leq 2k+2$, we can enumerate partitions of $V(X)$ in fpt time, and compute a minimum cut Z using the algorithm for Multiway Cut. In the general case when $S-X$ is not flexible, we assume that X is minimal and hence every connected component of $S-X$ contains a non-identity cycle. This implies that $S-X$ admits a unique satisfying assignment φ_X (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). 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 T in $S-Z$ has a different value under φ_X and φ_Z. Since non-identity 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 non-identity cycles in $S-X$ has the theta property. This allows us to use the method of important balanced subgraphs to get rid of non-identity 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 non-identity 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 [35], but works in $\mathcal{O}^*(4^k)$ time instead of $\mathcal{O}^*(2^{k^3})$ (later improved to $\mathcal{O}^*(2^{k^2})$ in [7]) time required by random sampling of important separators. We also note that essentially the same algorithm work for Min-2-Lin($\mathbb{F}$), where $\mathbb{F}$ is a field.

Min-2-Lin Algorithms for Euclidean Domains. Let us now consider the general Min-2-Lin($\mathbb{D}$) problem where $\mathbb{D}$ is a Euclidean domain. Euclidean domains are less restricted than fields and they consequently capture a wider and more multifaceted range of problems. There are many examples of interesting Euclidean domains that are not fields; the two prime examples are probably the ring of integers $\mathbb{Z}$ and the rings of univariate polynomials $\mathbb{F}[x]$ where the coefficients are members of some field $\mathbb{F}$. Important differences between fields and Euclidean domains 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, in Euclidean domains paths may also be obstructions. 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 $2x-2z=1$ and it has no integer solutions. This complicates the handling of Euclidean domains in algorithms. Another issue is that 2-Lin($\mathbb{D}$) is less studied for Euclidean domains where, in contrast, Gaussian elimination has been known for centuries and solves r-Lin($\mathbb{F}$) for every r in polynomial time. Polynomial-time algorithms are known for r-Lin($\mathbb{Z}$) and r-Lin($\mathbb{F}[x]$) [24, 25] for every r, but to the best of our knowledge, there are no general algorithms for arbitrary Euclidean domains described in the literature, even for the simpler 2-Lin($\mathbb{D}$) problem. This forces us to develop methods for checking consistency of 2-Lin($\mathbb{D}$) instances to be used in our Min-2-Lin algorithms. In short, Min-2-Lin($\mathbb{D}$) is intrinsically a more complicated problem and this is reflected in the more complicated fpt 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. [28, 29] solve this MinCSP using the recently introduced technique of flow augmentation.

Further related work. Group Feedback Edge Set (GFES) is a problem that is usually defined in terms of labelled graphs, but can also be defined in terms of equation systems 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 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*, as defined above; 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 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, hence GFES can equivalently be defined as finding a set of at most k edges which hits every inconsistent (i.e., unbalanced) 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 [17] was the first to study GFVS in terms of parameterized complexity, and showed FPT algorithms parameterized by $k + |\Gamma|$. Cygan et al. [11] showed 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. [22] showed a faster algorithm, solving GFVS in time $\mathcal{O}^*(4^k)$ using an LP-branching approach, also in the oracle model.

For comparison with Min-2-Lin, note that equations $x \cdot \gamma = y$ are equivalent to equations $x \cdot \alpha = y \cdot \beta$ for constants $\alpha, \beta \in \Gamma$ (taking $\gamma = \beta\alpha^{-1}$). Hence GFES corresponds roughly to the special case of Min-2-Lin($\mathbb{F}$) where $\mathbb{F}$ is a field, the instance is homogeneous, and with the additional restriction that no variable is allowed to take the zero value from $\mathbb{F}$. (Indeed, handling the set of variables taking the value 0 is a significant part of our algorithm for Min-2-Lin($\mathbb{F}$) over a field $\mathbb{F}$.) The case of Min-2-Lin($\mathbb{D}$) over general Euclidean domains $\mathbb{D}$, or even over the integers, is more complex by comparison, since multiplication in Euclidean domains is not guaranteed to form a group.

Guillemot [17] was also among the first to use half-integral LP-relaxations as a tool for constructing fixed-parameter algorithms. The method was refined by Cygan et al. [12] who used it to solve Multiway Cut in time $\mathcal{O}^*(2^k)$ and Almost 2-SAT in time $\mathcal{O}^*(4^k)$. The latter was further improved by Lokshtanov et al. [32] to $\mathcal{O}^*(2.3146^k)$. Iwata et al. [22] generalized the approach using tools from constraint satisfaction problems to provide $\mathcal{O}^*(4^k)$-time algorithms for a range of problems, including GFVS as noted above. Iwata et al. [23] later improved this to *linear time* $\mathcal{O}^*(4^k)$-time algorithms for an important subclass of these problems, by providing a fast combinatorial solution to the LP-relaxation. Wahlström [40] generalized the GFVS-results of Iwata et al. [22] further to the setting of biased graphs, via the Biased Graph Cleaning problem.

Göke et al. [15] considered the parameterized complexity of problems of resolving linear systems of equalities and inequalities under a mix of parameters, showing W[1]-hardness and FPT results (see also Bérczi et al. [3]).

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 used in the Min-2-Lin algorithms. In Sections 4 and 5 we present the general algorithm for Euclidean domains and faster algorithms for fields, respectively. Section 6 is devoted to W[1]-hardness results. We finish off in Section 7, summarizing and discussing the results, open questions and possible directions for future work.

Preliminaries and notation. We assume familiarity with the basics of graph theory, linear and abstract algebra, and combinatorial optimization. The necessary material can be found in, for instance, the textbooks by Diestel [13], Artin [1], and Schrijver [38], respectively.

We use the following graph-theoretic terminology in what follows. 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 the 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.

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 [43]. 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 $\mathcal{B}$. We refer to a set of cycles $\mathcal{B}$ with this property as *linear*. 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 over Euclidean domains.

2.1 LP-relaxation for Rooted Biased Graph Cleaning

Previous work by Wahlström [40] 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) path P connecting C to v_0. Such a rooted unbalanced cycle is referred to as a *balloon* [40]. 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 obviously 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 [40] 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 $x^* \in \{0, 1/2, 1\}^{V(G)}$. Finally, it is *persistent* in the sense that there is an optimal solution $X \subseteq E(G)$ such that if $x^*(v) = 1$ then $v \in X$ (and 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 $\text{supp}(x) = \{v \in V(G) \mid x(v) > 0\}$. 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 $G - \text{supp}(x)$. 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 [40].

In fact, an even stronger, more technical property holds, which we will need in what follows. Let G be an undirected graph with vertex weights $w : V(G) \to \mathbb{N}$. For a set $U \subseteq V(G)$, we let $w(U) = \sum_{v \in U} w(v)$.

Lemma 2.1. (Wahlström [40, Lemma 6]) *Let $x = V_1 + \frac{1}{2}V_{1/2}$ be a half-integral extremal LP-optimum for a* RBGC *instance $I = ((G, \mathcal{B}), w, v_0, k)$ with vertex weights $w : V(G) \to \mathbb{N}$ 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 we can find a replacement solution that grows the closed neighbourhood $N_G[S]$ to $N_G[S \cup V_R(x)]$ without paying a larger cost for deleting vertices. More formally, 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 [40].

Proposition 2.1. (Wahlström [40, Theorem 1]) RBGC *admits an fpt-algorithm with run-time $\mathcal{O}^*(2^k)$ assuming that $\mathcal{B}$ is given by a polynomial-time membership oracle.*

Proof. [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 is a set of vertex weights on G, $v_0 \in V(G)$ is the root vertex and $k \in \mathbb{N}$ is the deletion budget. We note that the algorithm for the unweighted case [40, Lemma 8] also applies in the presence of integer weights. We use the slightly more careful version of the extended preprint [31]. 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 2.1; such an optimum can be computed efficiently in a greedy manner [31]. 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 [31], 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 $\mathcal{O}^*(2^{2(k-\lambda)})$ time, and since $\lambda \geq k/2$ initially, this is $\mathcal{O}^*(2^k)$. □

Next, we consider the *edge deletion* version of the problem, Rooted Biased Graph Cleaning Edge (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 2.2. RBGCE *admits an fpt-algorithm with run-time $\mathcal{O}^*(2^k)$ 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 2.1 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 versions of the problems RBGC and RBGCE are in FPT. The result follows from same procedure as in the original paper [40], building on Prop. 2.1 and 2.2.

Proposition 2.3. *The non-rooted variants of* RBGC *and* RBGCE *both admit an fpt-algorithm with run-time $\mathcal{O}^*(4^k)$ assuming that $\mathcal{B}$ is given by a polynomial-time membership oracle.*

2.3 Important Balanced Subgraphs *Important separators* are a central concept in fpt algorithms for graph separation problems that was originally defined by Marx [33]. 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 $|C'| > |C|$. Marx showed that for any graph G and sets X, Y, there are at most $f(k)$ distinct important separators C with $|C| \leq k$ [33], and this bound was later improved to $f(k) = 4^k$ (see [10]). 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 [33] and Multicut [35] (see Cygan et al. [10] 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, and for $F \subseteq E(G)$ we denote $w(F) = \sum_{e \in F} w(e)$. 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. Note by Lemma 2.2 that we may assume here that H' is also connected. 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 $\mathcal{G} := \mathcal{G}(G, \mathcal{B}, k, v_0)$ be the family of connected balanced subgraphs in $(G, \mathcal{B})$ that contain v_0 and have cost at most k. 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 2.1. (Dominating family of important balanced subgraph) *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 $\mathcal{G} := \mathcal{G}(G, \mathcal{B}, k, v_0)$ be the family of connected balanced subgraphs in $(G, \mathcal{B})$ that contain v_0 and have cost at most k. Then, in $\mathcal{O}^*(4^k)$ time we can compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ such that $|\mathcal{H}| \leq 4^k$. Furthermore, every member of $\mathcal{H}$ is an important balanced subgraph of $(G, \mathcal{B})$.*

Before we present our proof of Theorem 2.1, we illustrate that important biased subgraphs are indeed a generalisation of important separators.

Example. 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, 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 2.1; this occupies the rest of the subsection. We first note that we may assume that strictly dominating subgraphs are connected.

Lemma 2.2. *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 minimize $c_G(H')$ 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 2.1, we begin by adapting the LP-relaxation for RBGC and Lemma 2.1 to the edge-deletion version RBGCE. More precisely, the LP-relaxation for RBGC and Lemma 2.1 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 give a natural reduction from RBGCE to RBGC, and as a consequence construct 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 2.1. 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 2.3. *Let $I = ((G, \mathcal{B}), v_0, k)$ 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 2.1 to $(G', \mathcal{B}')$ giving every vertex $v \in V(G)$ weight $w(v) = 2w(E(G)) + 1$ and the subdividing vertices weight 1. 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 2.1 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 $V(H') = V(G_R) \cup V(H)$ 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 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 2.4. *Let $x^* = X_1 + \frac{1}{2}X_{1/2}$ be a half-integral extremal optimum computed in Lemma 2.3. 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 $e \in E(C)$. Note that 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 $X_{1/2} = \delta_G(V_R)$. ∎

We can now show the main result.

Proof. [Proof of Theorem 2.1] 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 2.2 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 minimize $c_G(H')$. Then by Lemma 2.2 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 $B^* = X_1 + \frac{1}{2}X_{1/2}$ and let $X = X_1 \cup X_{1/2}$ be its 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, initialize a new branching state $B' = (E_0', E_1')$ 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 a 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 2.1.1. *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 $k' = k - |E_1|$ 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 2.4, 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 2.1.2. *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 2.4. 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 2.4. ◇

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 2.1.2 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 2.3 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 2.3. 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 2.1.2 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 2.3. 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 2.4 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 $\mathcal{O}^*(4^k)$ time. To see this, we use an approach that is similar to the one used in [40]. 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 optima 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 2.4 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 Euclidean domains, 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 $\mathcal{O}(1)$ amortized time per partition [21].

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 3.1. (Cygan et al. [12]) Multiway Cut *is solvable in $\mathcal{O}^*(2^k)$ time. If a solution exists, then the algorithm computes it in this time.*

Lemma 3.1. Partition Cut *is solvable in $\mathcal{O}^*(2^k)$ 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 $\mathcal{P} = \{B_1, \dots, B_m\}$. 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 $S = \{s_1, \dots, s_m\}$. 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 3.1. □

3.2 Pair Partition Cut For Min-2-Lin over arbitrary Euclidean domains, 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 $X \subseteq E(G)$ *fulfills* $(\{s, u\}, \{t, v\})$ if $G - X$ does not contain an $\{s, u\}$-path or a $\{t, v\}$-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 fulfills 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 $R(v_1, \dots, v_r)$, 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 satisfied every constraint in C i.e. $(\varphi(v_1), \dots, \varphi(v_r)) \in R$ 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 $\phi : V(G) \to \{0, \dots, k\}$ 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 either 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:

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

Theorem 3.1. (Section 5.7 in [28]) *The problem* MinCSP(Γ'_d) *is fpt parameterized by* $\ell = d + c$ *where* c *is total solution cost.*

The running time of MinCSP(Γ'_k) is significant even though it is fpt with parameter ℓ. It is not given explicitly by Kim et al. [29], but works out to 2^{ℓ^b} where $b \geq 12$ [28, Lemma 6.14]. Since this is greater than any other running time contribution in this paper, we treat this as a function $T(\ell)$ and give our other running time bounds (where appropriate) in terms of $T(\ell)$.

While there is a directed reduction from Pair Partition Cut to MinCSP(Γ'_k), we regard the following two-step reduction clearer and more readable, and the intermediate problem being an interesting example of a fixed-parameter tractable MinCSP.

Theorem 3.2. 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 $k+1$, and $\{\{u, v\} \in E(G) \mid (u = v) \in X\}$ is a $\mathcal{P}$-cut in G that fulfills $\mathcal{F}$. To obtain a solution to $((V, C), w, k)$ from a solution to the Pair Partition Cut instance, one may follow the same steps in the opposite direction.

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. To ensure that $v^{(i)} = 1$ for at most one value of i, add constraints $(\neg v^{(i)} \vee \neg v^{(j)})$ of weight $k+1$ for all $1 \leq i < j \leq k$. 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 $i \geq 1$, and $\phi(v^{(i)}) = 0$ otherwise. By construction, ϕ and ϕ' leave constraints of the same total weight unsatisfied. 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 $i \in \{1, \dots, k\}$, 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 Euclidean Domains

We let $\mathbb{D} = (D; +, \cdot)$ denote a Euclidean domain throughout this section. Our goal is to present an fpt algorithm for MIN-2-LIN($\mathbb{D}$). We start by reviewing basic definitions and facts about Euclidean domains in Section 4.1. In Section 4.2 we develop a polynomial-time algorithm for 2-LIN($\mathbb{D}$) and prove several useful lemmas along the way, building an understanding of the problem. We note that polynomial-time algorithms for r-LIN($\mathbb{D}$) are known for arbitrary $r \in \mathbb{N}$ when $\mathbb{D}$ is finite [2, Section 6] (in fact, this is true for arbitrary finite rings), the ring of integers [25] or the ring of univariate polynomials over $\mathbb{Q}$ [24]. However, we are unaware of such results for general Euclidean domains, even when $r = 2$. 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. Finally, in Section 4.6 we prove correctness of the algorithm and analyze its time complexity.

4.1 Basics of Euclidean Domains A *Euclidean domain* is an abstract algebraic structure generalizing properties of the integers. Informally, it is a commutative ring with integer division. Formally, $\mathbb{D} = (D; +, \cdot)$ is a Euclidean domain if it is an *integral domain* equipped with a *Euclidean function*. An integral domain is a commutative ring of size at least two where the product of any pair of nonzero elements is itself nonzero—that is, $\mathbb{D}$ does not contain a zero divisor. A Euclidean function on $\mathbb{D}$ is a function $f : D \to \mathbb{N}_0$ such that $f(0) = 0$ and for any $a, b \in D$ where $b \neq 0$, there exist $q, r \in D$ such that $a = bq + r$ and $f(r) < f(b)$. One may view q as a *quotient* and r as a *remainder*, and write $a \equiv r \mod b$ to denote that r is a remainder of integer division of a by b. All fields and the ring of integers $\mathbb{Z}$ are Euclidean domains; this follows from choosing the Euclidean function to be $f(x) = 1$ for all $x \neq 0$ and $f(x) = |x|$, respectively. Further examples include 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. When working with Euclidean domains, we assume that they are *effective* i.e. $\mathbb{D}$ admits a reasonable representation of elements such that basic operations (addition, subtraction, multiplication, computing quotients and remainders) requires polynomial time in the bit-size of the operands. In addition, we require the following property. Given an element $d \in D$, let $\|d\|$ denote the number of bits required to represent d.

PROPERTY 4.1. *In an effective ring $\mathbb{D}$, there is a polynomial function p such that $\|d_1 \cdot \ldots \cdot d_m\| \leq p(\|d_1\| + \ldots + \|d_n\|)$ for arbitrary $d_1, \dots, d_n \in \mathbb{D}$.*

This is a natural requirement since otherwise we cannot compute (or even write down) satisfying assignments to simple consistent instances of 2-LIN($\mathbb{D}$) like $\{x_1 = d_1 x_2, x_2 = d_2 x_3, \dots, x_{n-1} = d_n x_n\} \cup \{x_n = 1\}$ in polynomial time, we cannot perform efficient Gaussian elimination etc. In many cases (including all examples of Euclidean domains given above), p is the identity polynomial i.e. representing the product of elements requires at most as many bits as representing them individually.

It is important to note that quotients and remainders are not unique in $\mathbb{D}$. For a simple example, consider $\mathbb{D} = \mathbb{Z}$ with Euclidean function $f(x) = |x|$, let $a = 9$ and $b = 4$, and note that $9 = 4 \cdot 2 + 1$ and $9 = 4 \cdot 3 + (-3)$. Since $|1| < |4|$ and $|-3| < |4|$, both $q = 2$, $r = 1$ and $q = 3$, $r = -3$ are valid quotient-remainder pairs. However, if we fix the remainder r, then there is at most one value for q such that $a = b \cdot q + r$. As a corollary of this observation, if b divides a (i.e. $r = 0$), then the result of dividing a by b is unique. In such cases we write a/b to denote the unique quotient. A *unit* is an element u of D that admits a multiplicative inverse i.e. there is an element v of D such that $uv = 1$. For all $a, b \in D$, a *greatest common divisor* $\gcd(a, b)$ is a maximal (with respect to f) element of D that divides both a and b. If $\gcd(a, b)$ is a unit, then a and b are *co-prime*. A *least common multiple* $\operatorname{lcm}(a, b)$ is a minimal (with respect to f) element of D that is divisible by a and b. Observe that while $\gcd(a, b)$ is not unique, all greatest common divisors of a and b are congruent up to multiplication by units: if g_1 and g_2 are greatest common divisors of a and b, then there is a unit element u in D such that $a = b \cdot u$. The same congruence holds for the least common multiples. As a result, when discussing divisibility we can safely abuse notation by writing $\gcd(a,b)$ and $\mathrm{lcm}(a,b)$ to denote an arbitrary greatest common divisor or least common multiple of a and b, respectively. An analogue of the extended Euclidean algorithm works in effective Euclidean domains.

Proposition 4.1. ([39, Theorem 4.10]) *An equation $ax + by = c$ with $a, b, c \in D$ has a solution in $\mathbb{D}$ if and only if $g = \gcd(a, b)$ divides c, and all satisfying assignments are of the form $(x_0 + (b/g) \cdot r, y_0 - (a/g) \cdot r)$ for some fixed $x_0, y_0 \in D$ and arbitrary $r \in D$. Finally, there is a polynomial time algorithm that checks this condition and computes g, x_0, and y_0.*

In light of Proposition 4.1, we can assume that the instances of Min-2-Lin($\mathbb{D}$) that we are dealing with do not contain inconsistent equations (since those can be removed in polynomial time during a preprocessing stage where the parameter is decreased according to the weight of the equation). Moreover, we may assume that in every equation $ax + by = c$ the coefficients a and b are co-prime (since we may divide all coefficients by a $\gcd(a, b)$). By further preprocessing, we may assume that equations of the form $0 \cdot x + 0 \cdot y = 0$ do not appear in the instances: since they are satisfied by any assignment, they can be removed in advance without affecting the parameter.

We use the following distributive property of gcd and lcm:

Proposition 4.2. *Let $\mathbb{D}$ be a Euclidean domain and let $a_1, \ldots, a_n, b \in D$. Then every $\mathrm{lcm}(\gcd(a_1, b), \ldots, \gcd(a_n, b))$ is congruent to every $\gcd(\mathrm{lcm}(a_1, \ldots, a_n), b)$ up to multiplication by a unit element.*

A proof of this statement for $\mathbb{D} = \mathbb{Z}$ and $n = 2$ is a common exercise in number theory and algebra textbooks (see e.g. Exercise 23ε in [8] or Exercise III.3 in [30]). A proof can be found in [41]. It generalizes in a straightforward way to all Euclidean domains and all $n \in \mathbb{N}^+$ by noting that the elements of a Euclidean domain admit unique factorization up to multiplication by units.

4.2 2-Lin over Euclidean Domains The main goal of this section is to present a polynomial-time algorithm for 2-Lin($\mathbb{D}$). Our approach exploits a particular graph (known as the *primal* or *Gaifman* graph) that describes the structure of 2-Lin($\mathbb{D}$) instances. Thus, we begin by presenting algorithms for various graphs such as paths, stars, and acyclic graphs, where we use a result that resembles the Chinese Remainder Theorem (Lemma 4.3). Then we extend these results to *flexible* instances, which can be viewed as a generalization of acyclic instances. Finally, we use the algorithm for flexible instances as the basis for a polynomial-time algorithm that checks consistency of general 2-Lin($\mathbb{D}$) instances (Lemma 4.5). A useful simplification in our proofs is provided by *homogenization*—a procedure that transforms the solution space while preserving the primal graph of the instance (Lemma 4.1). This technique will be used frequently in this and following sections.

Let S be an instance of 2-Lin($\mathbb{D}$). We associate a *primal graph* with S: vertices of this graph correspond to the variables in $V(S)$, and two vertices x and y are connected by an edge if S contains an equation over x and y. We can think of an instance of 2-Lin($\mathbb{D}$) as a graph with edges $\{x, y\}$ labelled by equations over x and y. 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.

One useful trick to simplify consistent instances of 2-Lin($\mathbb{D}$) is *homogenization.* 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. Define a mapping Φ that acts on every variable $x \in V(S)$ by setting $x \mapsto a_x x' + b_x$ for some $a_x, b_x \in D$. 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 *homogenizing* if $\Phi(S)$ is homogeneous.

Lemma 4.1. *Every consistent instance of* 2-Lin($\mathbb{D}$) *admits a homogenizing variable substitution.*

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 homogenizing. $\square$

Now consider a path P of length $\ell - 1$ connecting variables x and y in S i.e. a system of $\ell - 1$ equations over ℓ distinct variables $p_1, \ldots, p_\ell$, where $x = p_1$ and $y = p_\ell$, with one equation relating p_i and p_{i+1} for all $i \in \{1, \ldots, \ell-1\}$. If $\ell = 2$, then P contains a single equation. Otherwise, we may eliminate intermediate variables to obtain an equation over x and y by recursively picking the first 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 over x and y obtained after eliminating all intermediate variables. This equation is denoted by e_P. For example, let P be a path with two equations over $\mathbb{Z}$: $x - 2z = 2$ and $z - y = 1$. The equation implied by P is obtained by eliminating z so it is $x - 2y = 4$. The following observation implies that variable elimination is safe.

Observation 4.1. *Every 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 $\{x, y\}$-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 $\{x, y\}$-paths in S. A simple example of flexible instances are acyclic instances. In the following lemma, we present a simple criterion for checking the consistency of such instances. We start with a useful observation. For a flexible instance S and any $x \in V(S)$, we define the instance $\mathrm{star}(S, x) = \{e_{xy}(S) \mid y \in V(S) \setminus \{x\}\}$ of 2-Lin($\mathbb{D}$).

Lemma 4.2. *Let S be a connected, acyclic instance of* 2-Lin($\mathbb{D}$). *For any $x \in V(S)$, S and $\mathrm{star}(S, x)$ have the same set of satisfying assignments.*

Proof. By Observation 4.1, an assignment that satisfies S also satisfies the equations implied by the paths in S. Consequently, it satisfies $\mathrm{star}(S, x)$ for any x. Now, suppose φ is a satisfying assignment to $\mathrm{star}(S, x)$ and consider an equation $e \in S$ over variables y and z. It suffices to show that φ satisfies e. Clearly, this holds if $y = x$ since then $e \in \mathrm{star}(S, x)$. Otherwise, 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 $\mathrm{star}(S, x)$, φ satisfies them, and hence also satisfies e. $\square$

We can now present the consistency criterion.

Lemma 4.3. *An acyclic instance of* 2-Lin($\mathbb{D}$) *is consistent if and only if it does not contain an inconsistent path.*

Proof. One direction of the proof follows by Observation 4.1: existence of an inconsistent path implies inconsistency of the whole system. For the opposite direction we proceed by induction on the number of equations. If a system contains only one equation, then the claims follow by Proposition 4.1. Now consider a system S with $n + 1$ equations where every path is consistent. If S contains more than one component, then the lemma follows by induction in each component. If S is connected (i.e. S is a tree), then pick a leaf z of S and assume x is the neighbour of z in S. By induction, the subtree $S' := S[V(S) \setminus \{z\}]$ without z is consistent, so Lemma 4.1 applies and there is a homogenizing substitution Φ to S'. Note that when Φ is applied to S, all equations except (possibly) the one involving x and z are homogenized.

Assume the variables in $V(S) \setminus \{x, z\}$ are $y_1, \ldots, y_n$. Then the equation $e_{xy_i}(\Phi(S'))$ can be written as $a_i x = b_i y_i$ for some co-prime $a_i, b_i \in D$. Let $B = \mathrm{lcm}(b_1, \ldots, b_n)$. An assignment satisfying $\Phi(S')$ can be obtained by setting $x \mapsto B \cdot r$ and $y_i \mapsto a_i(B/b_i) \cdot r$ for any $r \in D$. The assignment clearly satisfies $\mathrm{star}(\Phi(S'), x)$ so Lemma 4.2 implies that it also satisfies $\Phi(S')$.

We now obtain an assignment for all equations in S. Let the equation over x and z in $\Phi(S)$ be

$$a \cdot x + b \cdot z = c. \tag{4.1}$$

Since it is consistent, we may assume that a and b are co-prime by Proposition 4.1. We claim that the following holds by consistency of all paths in $\Phi(S)$.

CLAIM 4.3.1. $\gcd(b, b_i)$ *divides* c *for all* $i \in \{1, \ldots, n\}$.

Proof of claim: Consider $e_{zy_i}(\Phi(S))$. Since the $\{z, y_i\}$-path in $\Phi(S)$ goes through x, the equation can be obtained by cancelling out x from $ax + bz = c$ and $a_i x = b_i y_i$. Thus, we get

$$ab_i \cdot y_i + a_i b \cdot z = a_i c. \tag{4.2}$$

Assume without loss of generality that a and a_i are co-prime (otherwise divide all coefficients by $\gcd(a, a_i)$). By assumption, Equation (4.2) is consistent and, by Proposition 4.1, $\gcd(ab_i, a_i b)$ divides $a_i c$. Note further that the pairs (a, b), (a_i, b_i) and (a, a_i) are co-prime. Hence, $\gcd(ab_i, a_i b) = \gcd(b, b_i)$. Since a_i, b_i are co-prime, $\gcd(b, b_i)$ does not divide a_i and it only divides c. $\diamond$

To find a solution to the system S, substitute in $B \cdot r$ instead of x into Equation 4.1 obtaining $aB \cdot r + b \cdot z = c$, where r is a fresh variable. We claim that this equation is consistent. By Proposition 4.1, it suffices to show that $\gcd(aB, b)$ divides c. First, note that since a, b are co-prime, an element is a $\gcd(aB, b)$ if and only if it is a $\gcd(B, b)$. Now, let $\gcd(b, b_i)$ be g_i. By the definition of B and Proposition 4.2, every $\gcd(B, b)$ and $\operatorname{lcm}(g_1, \ldots, g_n)$ are congruent up to multiplication by units. Claim 4.3.1 implies that g_i divides c for all $i \in \{1, \ldots, n\}$, so by the definition of lcm, $\operatorname{lcm}(g_1, \ldots, g_n)$ also divides c. Thus, by choosing an appropriate value for r, we obtain an assignment that satisfies Equation 4.1 and $\Phi(S')$ simultaneously. Hence, the assignment satisfies $\Phi(S)$, and the lemma follows by applying the inverse variable substitution Φ^{-1}. $\square$

As an aside, a corollary of this lemma is that MIN-2-LIN($\mathbb{D}$) on acyclic instances is in FPT, since the problem can be reduced to solving (the edge-deletion variant of) MULTICUT on trees [18], where endpoints of every inconsistent path form a cut request. We also have the following algorithmic corollary.

COROLLARY 4.1. *There is a polynomial-time algorithm that checks whether an acyclic instance of* 2-LIN($\mathbb{D}$) *is consistent, and if so, computes a satisfying assignment.*

Proof. Let S be an acyclic instance of 2-LIN($\mathbb{D}$) and let $d_1, \ldots, d_n$ be the coefficients appearing in S. The proof of Lemma 4.3 is constructive – it produces a concrete satisfying assignment to S. The running time for constructing this solution is polynomial in Cn, where n is the number of equations in the system and C is the maximum bit-size of an element from $\mathbb{D}$ computed during the course of the algorithm. By Property 4.1, we have $C \leq \|d_1 \cdot \ldots \cdot d_n\| \leq p(\|d_1\| + \ldots + \|d_n\|) \leq p(\|S\|)$, where $\|S\|$ is the bit-size of S. We conclude that the algorithm runs in polynomial time. $\square$

The results for acyclic instances extend to flexible instances by considering spanning forests.

LEMMA 4.4. *A flexible instance* S *of* 2-LIN($\mathbb{D}$) *is consistent if and only if it contains no inconsistent path. Moreover, if* S *is connected, then* S *and* $\operatorname{star}(S, x)$ *have the same set of satisfying assignments for every* $x \in V(S)$.

Proof. Let S be a flexible instance of 2-LIN($\mathbb{D}$) and let T be a spanning forest of S. We claim that S and T have the same set of satisfying assignments, and then the lemma holds by Lemma 4.2 and Lemma 4.3. First, note that any assignment satisfying S also satisfies T since it is a subinstance of S. On the other hand, let φ be a satisfying assignment for T, and consider an equation e in $S \setminus T$ with variables x, y. It suffices to show that φ satisfies e. Since S is flexible, e is equivalent to $e_{xy}(S)$. Furthermore, T is a spanning forest, so it contains a path connecting x and y, and the path implies an equation equivalent to $e_{xy}(S)$ and e. Hence, φ satisfies e and the lemma holds. $\square$

Analogously to Corollary 4.1, we have the following algorithmic result.

COROLLARY 4.2. *There is a polynomial-time algorithm that checks whether a flexible instance of* 2-LIN($\mathbb{D}$) *is consistent, and if so, computes a satisfying assignment.*

Lemma 4.4 suggests an fpt algorithm for solving MIN-2-LIN($\mathbb{D}$) for flexible instances: find a spanning forest and reduce to MULTICUT by adding endpoints of every inconsistent path in the forest as a cut request. This is also an important stepping stone towards checking consistency of any instance of 2-LIN($\mathbb{D}$). We provide a full algorithm below.

Lemma 4.5. *There is a polynomial-time algorithm that checks whether an instance of* 2-Lin($\mathbb{D}$) *is consistent, and if so, computes a satisfying assignment.*

Proof. Let S be an instance of 2-Lin($\mathbb{D}$). Without loss of generality, assume that it is connected, and compute a spanning tree T. We proceed by checking whether S is flexible or not. To do so, we consider the equations $e \in S - T$. Assume that e equals $a_1x + b_1y = c_1$ and let $e_{xy}(T)$ equal $a_2x + b_2y = c_2$. We want to check whether these two equations are equivalent. To do so, we 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 $A = b_1a_2 - a_1b_2$, $B = c_1a_2 - a_1c_2$, and consider four cases:

- 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.

If $Ay = B$ is inconsistent, then no assignment can satisfy both e and $e_{xy}(T)$, hence S is inconsistent. If $A = B = 0$, then 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. Thus, the only case we need to consider further is when S is flexible. This case is handled by Corollary 4.2 and this concludes the proof. ∎

4.3 Iterative Compression We reduce Min-2-Lin($\mathbb{D}$) to a simpler problem by combining homogenization with *iterative compression*. The latter method uses a compression routine that takes a problem instance together with a solution as an input, and either calculates a smaller solution or verifies that the provided one has minimum size. An optimal solution is then computed by iteratively building up the instance while improving the solution at each step. If the compression routine runs in fpt time, then the whole algorithm also runs in fpt time. A more comprehensive treatment of the method can be found in [10, Chapter 4]. We use it to provide a reduction 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 4.6. *If* DML($\mathbb{D}$) *is solvable in* $\mathcal{O}^*(f(k))$ *time, then* Min-2-Lin($\mathbb{D}$) *is solvable in* $\mathcal{O}^*(2^k f(k))$ *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 may apply a homogenizing variable substitution to it by Lemma 4.1. 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 $\mathcal{O}^*(2^k f(k))$. ∎

4.4 Graph Cleaning for Euclidean Domains Lemma 4.4 provides us with a good idea of how to solve Min-2-Lin($\mathbb{D}$) restricted to acyclic and flexible instances. To approach the general solution, we now need to consider cycles that make instances rigid. If a consistent cycle is flexible, we say that it is an *identity* cycle. If a consistent cycle is consistent, but not flexible, then it is a *non-identity* cycle. There is an alternative characterization of identity and non-identity cycles in terms of the number of satisfying assignments.

Lemma 4.7. *A consistent cycle in* 2-Lin($\mathbb{D}$) *is identity if and only if it admits more than one satisfying assignment, while a consistent cycle is non-identity 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 4.1, we may assume without loss of generality that C is homogeneous. Then, C is satisfied by the all-zero assignment. To prove the lemma it suffices to show that C admits a non-zero assignment if and only if it is identity.

On the one hand, suppose that C is an identity 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 $i \in \{1, \dots, m\}$, 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 $i \in \{1, \dots, m\}$. 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 4.1, 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 non-identity cycle. By definition, there are variables $x, y \in V(C)$ such that the $\{x, y\}$-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 results allows us to use the graph cleaning machinery to remove non-identity cycles.

Lemma 4.8. *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 identity cycles in S. Then $(G_S, \mathcal{B}_S)$ is a biased graph.*

Proof. By Lemma 4.1, it suffices to consider a homogeneous instance S. 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 $\{x, y\}$-paths in G_S, and assume $P \cup R$ is a non-identity 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 non-identity, 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 4.7, define nonzero assignments φ_P and φ_R satisfying all equations in P and R, respectively. Note that by Observation 4.1, they also satisfy $ax = by$ i.e.

$$a \cdot \varphi_P(x) = b \cdot \varphi_P(y), \tag{4.3}$$

$$a \cdot \varphi_R(x) = b \cdot \varphi_R(y). \tag{4.4}$$

Therefore, after multiplying (4.3) by $\varphi_R(x)$ and (4.4) by $\varphi_P(x)$, we obtain:

$$a \cdot \varphi_P(x) \cdot \varphi_R(x) = b \cdot \varphi_P(y) \cdot \varphi_R(x), \tag{4.5}$$

$$a \cdot \varphi_R(x) \cdot \varphi_P(x) = b \cdot \varphi_R(y) \cdot \varphi_P(x). \tag{4.6}$$

Since the right hand sides of (4.5) and (4.6) 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 identity by Lemma 4.7 and we arrive at a contradiction. ◻

By Lemma 4.6, 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 identity or non-identity. To apply graph cleaning, we construct a *rooted graph for I* as follows.

DEFINITION 4.1. *Let $I = (S, w_S, X, k)$ be an instance of* DML($\mathbb{D}$). *The* rooted graph *for I is a biased graph $(G_I, \mathcal{B}_I)$ 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 $V(X)$. Moreover, let $\mathcal{B}_I \subseteq 2^{E(G_I)}$ be the set of identity 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 two paths connecting them on the cycle imply the same equation.

LEMMA 4.9. *$(G_I, \mathcal{B}_I)$ 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 non-identity 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 4.8. ◻

The following is an immediate algorithmic consequence of Theorem 2.1 and Lemma 4.9.

OBSERVATION 4.2. *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 $\mathcal{O}^*(4^q)$ we can compute a dominating family $\mathcal{H}$ for $\mathcal{G}$ of size at most 4^q.*

Now we characterize yes-instances of MIN-2-LIN($\mathbb{D}$). To this end, let Z be an optimal solution to I disjoint from X, and φ be an assignment satisfying $S - Z$. The variables in $V(S)$ are partitioned into two sets by φ: $V_0 = \{v \in V(S) \mid \varphi(v) = 0\}$ and $V_\emptyset = \{v \in V(S) \mid \varphi(v) \neq 0\}$, i.e. those assigned zero and non-zero values, respectively.

LEMMA 4.10. *Let $K \subseteq V(S)$ be a connected component of $S - (X \cup Z)$.*

1. *Either $K \subseteq V_0$ or $K \subseteq V_\emptyset$.*
2. *If $K \subseteq V_\emptyset$, then $(S - (X \cup Z))[K]$ is flexible.*
3. *If $K \cap V(X) = \emptyset$, we may assume without loss of generality that $K \subseteq V_0$.*

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 now introduce the *zero-free subgraph $H_\emptyset$* of the rooted graph $(G_I, \mathcal{B}_I)$.

DEFINITION 4.2. *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, the* 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 4.10) and add the edge $\{s, x\}$ to $E(H_\emptyset)$.*

LEMMA 4.11. *$H_\emptyset$ is a connected balanced subgraph of $(G_I, \mathcal{B}_I)$, and $c_{G_I}(H_\emptyset) \leq 3k + 1$.*

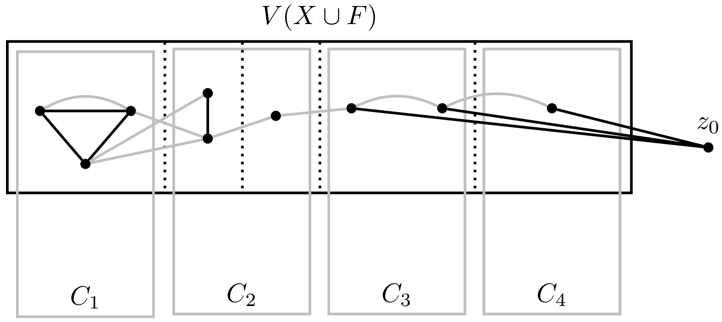


Figure 4: An illustration of the auxiliary instance $H_{\mathcal{P}} = H(S, X, F, \mathcal{P})$. Here, $C_1, \ldots, C_4$ are all components of $S' = S - (X \cup F)$ with C_3 and C_4 being the only rigid components. Black circular vertices represent the vertices of the primal graph of $H_{\mathcal{P}}$, 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 $C_1 \ldots, C_4$.

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 4.10, $(S - (X \cup Z))[K]$ 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 Algorithm for Min-2-Lin over Euclidean Domains In the end of this section we will present our fpt algorithm for Min-2-Lin($\mathbb{D}$). By Lemma 4.6, 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. 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 4.2.

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 4 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 non-identity 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 formalized below.

LEMMA 4.12. *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 non-identity 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 4.12.1 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 non-identity 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 summarized 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 4.1. 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 4.2.

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 Lemma 4.5 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 3.2 to solve $I_{\mathcal{P}}$. If $I_{\mathcal{P}}$ has a solution Y, then use Lemma 4.5 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 4.14) that show 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 4.14 is simplified with the aid of the following lemma.

Lemma 4.13. *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 fulfills $\mathcal{F}_{\mathcal{P}_Z}$.*

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 4.12.1, 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 4.12.2, $\varphi_H(x) = \varphi_Z(x)$. 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 4.14. *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 fulfills $\mathcal{F}_{\mathcal{P}_Z}$ is a solution to I.*

Proof. We know that $H_{\mathcal{P}_Z}$ is consistent by Lemma 4.12.1 and 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 fulfills $\mathcal{F}_{\mathcal{P}_Z}$, 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 $\text{star}(K, x)$, which due to Lemma 4.4 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 4.14.1. $(S-Y)[U]$ *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 non-identity cycle avoiding $X \cup F$. Furthermore, if there were a non-identity 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 non-identity cycle and it is indeed flexible.

We now show that $(S-Y)[U]$ is consistent. Because $(S-Y)[U]$ is flexible, we obtain from Lemma 4.4 that it suffices to show that $(S-Y)[U]$ contains no 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 $\{x,y\}$-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 $\mathcal{F}_{\mathcal{P}_Z}$. But this contradicts our assumption that P is in $(S-Y)[U]$ because Y fulfills $\mathcal{F}_{\mathcal{P}_Z}$ and therefore intersects P. $\diamond$

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 4.14.2. *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. $\diamond$

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 $|Y| \le k$. Lemma 4.13 implies that Z is a $\mathcal{P}_Z$-cut in S' that fulfills $\mathcal{F}_{\mathcal{P}_Z}$. Moreover, Z is a solution of I so $|Z| \le k$. It follows that if Y is a minimum such $\mathcal{P}_Z$-cut, then $|Y| \le k$ 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. $\mathcal{O}^*(Q(k))$ is the run-time for the algorithm for Pair Partition Cut given in Theorem 3.2. This makes it clear that the main bottleneck for our algorithm is the underlying algorithm for Pair Partition Cut.

Theorem 4.1. Min-2-Lin($\mathbb{D}$) *is in* FPT *and can be solved in time $\mathcal{O}^*(k^{\mathcal{O}(k)}Q(k))$.*

Proof. We start by analyzing 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 $|Y| \le k$ 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 4.2. By Lemma 4.11, $H_\emptyset$ is balanced, connected, and $c_{G_I}(H_\emptyset) \le 3k+1$. 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 4.12.1 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 4.14 on the instance I' and therefore every $\mathcal{P}_Z$-cut Y in S'' that fulfills $\mathcal{F}_{\mathcal{P}_Z}$ 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 analyzing 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 $\mathcal{O}^*(4^{3k+1})$ using Observation 4.2. 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 $\mathcal{O}((4k)^{4k})$. 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 Lemma 4.5. 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 3.2 in fpt-time with respect to $k' \leq k$, i.e. in time $\mathcal{O}^*(Q(k))$. Therefore, the total time required by the algorithm is at most

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

which shows that DML($\mathbb{D}$) is fpt with respect to k. By Lemma 4.6, there is another factor of 2^k in the running time of the algorithm for Min-2-Lin($\mathbb{D}$), which is dominated by $k^{\mathcal{O}(k)}$, so asymptotically we obtain the same running time for Min-2-Lin($\mathbb{D}$). ∎

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 $\mathcal{O}^*(k^{\mathcal{O}(k)})$ time algorithm for arbitrary fields and an $\mathcal{O}^*((2p)^k)$ time algorithm for finite p-element fields. The improvements use the nicer structural properties of fields, mainly the fact that every nonzero element has a multiplicative inverse. Section 5.1 demonstrates how Min-2-Lin($\mathbb{F}$) differs from the more general Min-2-Lin over Euclidean domains 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 in arbitrary Euclidean domains. 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 Lemmas 4.2 and 4.3.

Corollary 5.1. *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 5.1. *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 5.1 so Lemma 4.1 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 $2^{\mathcal{O}(k)}$ from the time complexity of iterative compression.

LEMMA 5.2. *If* DML($\mathbb{F}$) *is solvable in* $\mathcal{O}^*(f(k))$ *time, then* MIN-2-LIN($\mathbb{F}$) *is solvable in* $\mathcal{O}^*(f(k))$ *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 4.6, 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 5.2. 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 Euclidean domains, 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 5.2. 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 5.2) 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 4.1. 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 4.2.

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 3.1. 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 5.2 avoids the branching step in the general compression algorithm from Lemma 4.6).

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 4.14 but its proof is noticeably different.

LEMMA 5.3. *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 non-identity cycles. Then K is flexible, and by Lemma 5.1, 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 4.10 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 5.1. MIN-2-LIN($\mathbb{F}$) *can be solved in in* $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ *time.*

Proof. By Lemma 5.2, 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 4.2). By Lemma 4.11, 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 5.3 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 4.2, the algorithm computes a dominating family $\mathcal{H}$ of $\mathcal{G}$ of size at most 4^{3k+1} in time $\mathcal{O}^*(4^{\mathcal{O}(k)})$. 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 4.11 so $|V(X \cup F_H)| \leq 4k$ and the enumeration of partitions requires $\mathcal{O}^*((4k)^{\mathcal{O}(4k)})$ time. Computing the $\mathcal{P}$-cut requires at most $\mathcal{O}^*(2^{4k})$ time by Lemma 3.1 and the total running time is

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

∎

5.4 Even Faster Algorithm for Finite Fields Let $\mathbb{F}_p$ be a finite p-element field with $p \geq 3$. For Min-2-Lin($\mathbb{F}_2$) a $\mathcal{O}^*(1.977^k)$ time algorithm can be obtained using the approach of [36]. Every finite field obviously has an effective representation so we assume without loss of generality that $\mathbb{F}_p$ is effective. Wedderburn's Little Theorem (see, for instance, [20]) implies that if $\mathbb{D}$ is a finite Euclidean domain, then $\mathbb{D}$ is a field. Hence, the results in this section cover Min-2-Lin for every finite Euclidean 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 $\mathcal{O}^*(p^{2k})$ time by the currently best algorithm for ULC [22]. In this section we present a faster algorithm for Min-2-Lin($\mathbb{F}_p$) that runs in $\mathcal{O}^*((2p)^k)$ time. By equation subdivision and Lemma 5.2, 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 $\mathcal{O}^*(2^k)$ 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 identity 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. Hence, we immediately obtain the following.

Lemma 5.4. ([43]) *$(G_\alpha, \mathcal{B}_\alpha)$ is a biased graph.*

Clearly, there is a polynomial time algorithm that checks whether a cycle is identity 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 5.2. 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 $\mathcal{O}^*((2p)^k)$ time.*

Proof. By equation subdivision and Lemma 5.2, 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 5.2.1. *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 non-identity 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 non-identity, $\alpha(x_1) \cdot \beta \neq \alpha(x_2)$. 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 5.2.2. *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 non-identity cycle, $t \notin V(H)$. Since every cycle of H is identity, 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 5.1. 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 5.2.1 and 5.2.2 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 $\mathcal{O}^*(2^k)$ time by Proposition 2.2 so the total running time is $\mathcal{O}^*((2p)^k)$. □

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 $r \geq 2$), and it rules out the possibility that our fixed-parameter tractable algorithms can be improved to polynomial-time algorithms. 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. [9] 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] contains 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 [10, Section 13.6.3], however many details are different. The reduction is based on the following W[1]-hard problem [14, Lemma 1].

MULTICOLOURED CLIQUE	
INSTANCE:	A graph $G = (V, E)$ with vertices partitioned into k colour classes $V_1, \dots, V_k$.
PARAMETER:	k.
QUESTION:	Does G contain a clique with exactly one vertex from each V_i, $1 \le i \le k$?

THEOREM 6.1. *Let $\mathbb{G} = (D; +)$ denote a group with at least two elements. Then,* MIN-r-LIN($\mathbb{G}$) *is* W*[1]-hard for any $r \ge 3$ even if all equations have weight 1.*

Proof. The reduction is presented in two steps: given an arbitrary instance $(G, k, (V_1, \dots, V_k))$ 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 $(G, k, (V_1, \dots, V_k))$ 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 $1 \le i \le k$.

(4) $\sum_{e \in E_{ij}} y_e = 1$ for all $1 \le i < j \le k$.

(5) $\sum_{u \in V_j \setminus \{v\}} x_u + \sum_{e \in E_{ijv}} y_e = 1$ for all $v \in V(G)$.

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 $1 \le i \le k$ and exactly one variable in $\{y_e \mid e \in E_{ij}\}$ for each pair $1 \le i < j \le k$ may be set to 1 since the budget $k + \binom{k}{2}$ is tight.

Now consider the equations in (5). Intuitively, for any variable $v \in V_i$, the corresponding equation implies that either x_v is set to 0 or at least one y_e for an edge e incident to v is set to 1. Formally, let φ be an assignment that does not satisfy k' constraints in S. If $\varphi(x_v) = 1$, then $\varphi(x_u) = 0$ for all $u \in V_i \setminus \{v\}$. Hence, $\sum_{u \in V_j \setminus \{v\}} \varphi(x_u) = 0$ and $\varphi(y_{\{v,w\}}) = 1$ for some $w \in V_j$. Moreover, $\varphi(y_e) = 0$ for all edges $e \in E_{ij} \setminus \{v, w\}$. In the equation for w in (5) we have $\sum_{e \in E_{ijw}} y_e = 1$. Hence, $\varphi(u) = 0$ for all $u \in V_j \setminus \{w\}$, and $\varphi(w) = 1$. On the other hand, if $\varphi(x_v) = 0$, then there is exactly one $u \in V_i \setminus \{v\}$ such that $\varphi(x_u) = 1$ so $\varphi(y_e) = 0$ for all edges $e \in E(G)$ incident to v. 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 $1 \le i \le r$ and $1 \le j \le k + 2$. 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 $v_i^{(1)}, \dots, v_i^{(k+2)}$ 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 $1 \le \ell \le k$, this quantity is minimized 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 $i \in \{1, \dots, r\}$ and replace the equation with the following system:

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

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 a Euclidean domain cannot contain a zero divisor. Next, we give examples of commutative rings $\mathbb{K}$ with zero divisors such that MIN-2-LIN($\mathbb{K}$) is W[1]-hard. Our starting point is the following problem, which has previously been used as a source of W[1]-hardness for MINCSP problems [29, 34].

PAIRED MIN CUT	
INSTANCE:	A graph G, vertices $s, t \in V(G)$, and an integer k, where the st-max flow in G is $2k$; a set of disjoint edge pairs $C \subseteq \binom{E(G)}{2}$
PARAMETER:	k
QUESTION:	Is there an st-mincut $X \subseteq E$ which is the union of k pairs from C?

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 6.1. PAIRED MIN CUT *is* W*[1]-hard, even for split instances.*

Proof. It is well known that PAIRED MIN CUT is W[1]-hard in its standard form [29, 34]. 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

$$X = \{e \in E(G) \mid \{e_1, e'_1, e_2, e'_2\} \subseteq X'\}.$$

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. ∎

We now show a general W[1]-hardness result for MIN-2-LIN($\mathbb{K}$).

THEOREM 6.2. *Let $\mathbb{K} = (K; +, \cdot)$ be a commutative ring with additive neutral element 0. If there are elements $\alpha_1, \alpha_2 \in K$ such that $\alpha_1^2 \neq 0$, $\alpha_2^2 \neq 0$, and $\alpha_1 \cdot \alpha_2 = 0$, then* MIN-2-LIN($\mathbb{K}$) *is* W*[1]-hard.*

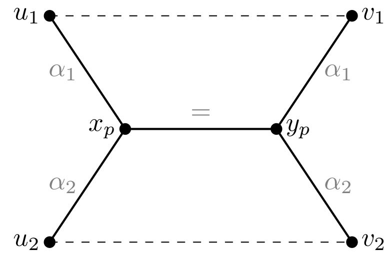


Figure 5: System of equations obtained from a pair of edges $p = \{e_1, e_2\}$ where $e_i = \{u_i, v_i\}$ in the fpt reduction from Paired Min Cut. Edges e_1 and e_2 are illustrated by dashed lines, while the equations are illustrated by solid lines with labels describing equations between connected variables.

Proof. We reduce from an arbitrary split instance of Paired Min Cut. Let $I = (G, s, t, k, C)$ be the input instance and let $G = G_1 \cup G_2 = G[U_1] \cup G[U_2]$ form the split. Divide the source s into two vertices s_1, s_2 where $N_G(\{s_i\}) = N_G(\{s\}) \cap U_i$ for $i \in \{1, 2\}$, but keep the sink t intact. We compute an instance of Min-2-Lin($\mathbb{K}$) as follows. Introduce one variable for every vertex in the resulting graph and initially turn every edge $\{u, v\}$ into an equation $u = v$ of weight $k + 1$. Force $s_1 = \alpha_1$, $s_2 = \alpha_2$ and $t = 0$ by equations of weight $k + 1$ each. Finally, for every pair $p \in C$, say $p = \{e_1, e_2\}$ where $e_i \subseteq U_i$ for $i \in \{1, 2\}$, do the following. Remove the equations corresponding to the edges in p. Create two new variables x_p, y_p. Let $e_1 = \{u_1, v_1\}$ and $e_2 = \{u_2, v_2\}$. Create equations

$$\begin{aligned} \alpha_1 \cdot u_1 &= \alpha_1 \cdot x_p, \\ \alpha_2 \cdot u_2 &= \alpha_2 \cdot x_p, \\ \alpha_1 \cdot y_p &= \alpha_1 \cdot v_1, \\ \alpha_2 \cdot y_p &= \alpha_2 \cdot v_2 \end{aligned}$$

of weight $k+1$ each, and an equation $x_p = y_p$ of weight 1. See Figure 5 for an illustration. Perform this for every pair in C. Let S be the resulting set of equations. We claim that S has a solution of cost at most k if and only if I is a yes-instance.

On the one hand, let $X \subseteq E(G)$ be the union of k pairs of edges, and let $X' = \{x_p = y_p \mid p \in C \text{ and } p \subseteq X\}$. We claim that $S - X'$ is satisfiable. For a vertex $v \in U_i$, set $v = \alpha_i$ if v is reachable from s in $G - X$, and $v = 0$ otherwise. Consider a pair of vertices x_p, y_p for $p \in C$, and suppose that the assignment above cannot be consistently extended to x_p and y_p. Then this implies that there is an edge $e_i = \{u, v\} \in p$ such that $u, v \in U_i$ and $\alpha_i \cdot u \neq \alpha_i \cdot v$. Since the value assigned to u and v is either 0 or α_i, we have $u \neq v$. This implies that e_i crosses the cut in $G - X$, contradicting our assumption that $e_i \notin X$. Hence, $S - X'$ is satisfiable.

On the other hand, suppose that there is a solution where equations of cost at most k are not satisfied, and let X' be the set of these equations. Then clearly every equation in X' is of the form $x_p = y_p$ for some pair $p \in C$. Let $X \subseteq C$ be the union of edges participating in these pairs. We claim that X is an st-cut. Assume to the contrary that $G - X$ contains a path P from s to t. Then that path corresponds to a chain of equations in S, from s_i to t ($i \in \{1, 2\}$), where every edge $\{u, v\}$ of the path corresponds to either an equation $u = v$ or a chain of equations $\alpha_i u = \alpha_i x$, $x = y$, $\alpha_i y = \alpha_i v$, where every equation in the chain is satisfied. Since $\alpha_i^2 \neq 0$, we have $\alpha_i \cdot s_i \neq \alpha_i \cdot 0$ so every variable on the path is assigned a non-zero value. This contradicts that $S - X'$ is satisfiable since $t = 0$. □

We illustrate Theorem 6.2 with an example. The *direct product* of two rings $\mathbb{K}_1 = (K_1; +_1, \cdot_1)$ and $\mathbb{K}_2 = (K_2; +_2, \cdot_2)$ is denoted $\mathbb{K}_1 \times \mathbb{K}_2 = (K; +, \cdot)$. Its domain R consists of the ordered pairs $\{(d_1, d_2) \mid d_1 \in K_1, d_2 \in K_2\}$ 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 $\mathbb{K} = \mathbb{K}_1 \times \mathbb{K}_2$ and $\mathbb{K}_1, \mathbb{K}_2$ are commutative rings that are not zero rings, then Min-2-Lin($\mathbb{K}$) is W[1]-hard. To see this, let $0_1 \in K_1, 0_2 \in K_2$ denote the additive identities and $1_1 \in K_1, 1_2 \in K_2$ denote the multiplicative identities. By setting $\alpha_1 = (0_1, 1_2)$ and $\alpha_2 = (1_1, 0_2)$, Theorem 6.2 is applicable and Min-2-Lin($\mathbb{K}$) is W[1]-hard. This argument can easily be extended to products of several commutative rings. 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. For example, $\mathbb{Z}/6\mathbb{Z} \cong \mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/3\mathbb{Z}$. Hence, Min-2-Lin($\mathbb{Z}/6\mathbb{Z}$) and more generally Min-2-Lin($\mathbb{Z}/m\mathbb{Z}$) where m is not a prime power is W[1]-hard.

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 an efficient Euclidean domain. 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 commutative rings $\mathbb{K}$ (that are not Euclidean domains) such that Min-2-Lin($\mathbb{K}$) is W[1]-hard.

The borderline between fixed-parameter tractable and W[1]-hard Min-2-Lin problems is not clear, and this is true even for finite commutative rings. Wedderburn's Little Theorem (see, for instance, [20] for a proof) states that if $\mathbb{K}$ is a finite ring, then either (1) $\mathbb{K}$ is a field (and Min-2-Lin($\mathbb{K}$) is in FPT) or (2) $\mathbb{K}$ contains zero divisors. We know that there are $\mathbb{K}$ with zero divisors such that Min-2-Lin($\mathbb{K}$) is W[1]-hard, but it is an open question whether the problem is always W[1]-hard when $\mathbb{K}$ 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 $n \geq 2$? 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. A noticeable example is the four-element commutative ring $\mathbb{F}_2[x]/(x^2+x)$ whose elements can be viewed as arrays

$$\begin{pmatrix} x & 0 \\ y & x \end{pmatrix}$$

with $x, y \in \mathbb{F}_2$.

We suspect that our fixed-parameter tractable algorithm for Min-2-Lin over Euclidean domains 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, $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ or better would be a significant challenge. There is also room for improvements in the Min-2-Lin algorithms for fields. Consider our $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$ 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 $\mathcal{O}^*((2p)^k)$ time. Is there an $\mathcal{O}^*(c^k)$ 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 $\mathcal{O}^*((dp)^k)$ time?

Another more general question is about 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 $\mathcal{O}^*(2^{\mathcal{O}(k^3)})$ to $\mathcal{O}^*(2^{\mathcal{O}(k \log k)})$, and simplifying the algorithm for Euclidean domains. 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. [23], 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

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. The first and third authors acknowledge support from the Engineering and Physical Sciences Research Council (EPSRC, project EP/V00252X/1).

References

[1] Michael Artin. *Algebra*. Pearson, 2011.

[2] Vikraman Arvind and T. C. Vijayaraghavan. Classifying problems on linear congruences and abelian permutation groups using logspace counting classes. *Computational Complexity*, 19(1):57–98, 2010.

[3] Kristóf Bérczi, Alexander Göke, Lydia Mirabel Mendoza-Cadena, and Matthias Mnich. Resolving infeasibility of linear systems: A parameterized approach. Technical Report TR-2022-02, Egerváry Research Group, 2022. www.cs.elte.hu/egres.

[4] Nicolas Bousquet, Jean Daligault, and Stéphan Thomassé. Multicut is FPT. *SIAM Journal on Computing*, 47(1):166–207, 2018.

[5] Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[6] Rajesh Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michał Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. *SIAM Journal on Computing*, 45(4):1171–1229, 2016.

[7] Rajesh Chitnis, Marek Cygan, Mohammataghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(4):1–28, 2015.

[8] Allan Clark. *Elements of Abstract Algebra*. Courier Corporation, 1984.

[9] Robert Crowston, Gregory Gutin, Mark Jones, and Anders Yeo. Parameterized complexity of satisfying almost all linear equations over $\mathbb{F}_2$. *Theory of Computing Systems*, 52(4):719–728, 2013.

[10] Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[11] Marek Cygan, Marcin Pilipczuk, and Michał Pilipczuk. On group feedback vertex set parameterized by the size of the cutset. *Algorithmica*, 74(2):630–642, 2016.

[12] Marek Cygan, Marcin Pilipczuk, Michał Pilipczuk, and Jakub Onufry Wojtaszczyk. On multiway cut parameterized above lower bounds. *ACM Transactions on Computation Theory*, 5(1):1–11, 2013.

[13] Reinhard Diestel. *Graph Theory*. Springer, 2016.

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

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

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

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

[18] Jiong Guo and Rolf Niedermeier. Fixed-parameter tractability and data reduction for multicut in trees. *Networks: An International Journal*, 46(3):124–135, 2005.

[19] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001.

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

[21] Semba Ichiro. An efficient algorithm for generating all partitions of the set $\{1, 2, \ldots, n\}$. *Information Processing Letters*, 7(1):41–42, 1984.

[22] Yoichi Iwata, Magnus Wahlström, and Yuichi Yoshida. Half-integrality, LP-branching, and FPT algorithms. *SIAM Journal on Computing*, 45(4):1377–1411, 2016.

[23] Yoichi Iwata, Yutaro Yamaguchi, and Yuichi Yoshida. 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)*, pages 462–473, 2018.

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

[25] Ravindran Kannan and Achim Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM Journal on Computing*, 8(4):499–507, 1979.

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

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

[28] Eun Jung Kim, Stefan Kratsch, Marcin Pilipczuk, and Magnus Wahlström. Solving hard cut problems via flow-augmentation. *CoRR*, abs/2007.09018, 2020.

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

[30] Edmund Landau. *Elementary Number Theory*. American Mathematical Society, 2021.

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

[32] Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Trans. Algorithms*, 11(2):15:1–15:31, 2014.

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

[34] Dániel Marx and Igor Razgon. Constant ratio fixed-parameter approximation of the edge multicut problem. *Information Processing Letters*, 109(20):1161–1166, 2009.

[35] Dániel Marx and Igor Razgon. Fixed-parameter tractability of multicut parameterized by the size of the cutset. *SIAM Journal on Computing*, 43(2):355–388, 2014.

[36] Marcin Pilipczuk, Michał Pilipczuk, and Marcin Wrochna. Edge bipartization faster than 2^k. *Algorithmica*, 81(3):917–966, 2019.

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

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

[39] Joachim von zur Gathen and Jürgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, 2013.

[40] Magnus Wahlström. LP-branching algorithms based on biased graphs. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA-2017)*, pages 1559–1570, 2017.

[41] Proof Wiki. GCD and LCM distribute over each other. `https://proofwiki.org/wiki/GCD_and_LCM_Distribute_Over_Each_Other`. (Accessed: 04.06.2022).

[42] Mingyu Xiao. Simple and improved parameterized algorithms for multiterminal cuts. *Theory of Computing Systems*, 46(4):723–736, 2010.

[43] Thomas Zaslavsky. Biased graphs. I. Bias, balance, and gains. *Journal of Combinatorial Theory, Ser. B*, 47(1):32–52, 1989.