The Complexity of $k$-Means Clustering when Little is Known

Robert Ganian $^{*1}$ Thekla Hamm $^{*1}$ Viktoria Korchemna $^{*1}$ Karolina Okrasa $^{*2}$ Kirill Simonov $^{*1}$

Abstract

In the area of data analysis and arguably even in machine learning as a whole, few approaches have been as impactful as the classical $k$-means clustering. Here, we study the complexity of $k$-means clustering in settings where most of the data is not known or simply irrelevant. To obtain a more fine-grained understanding of the tractability of this clustering problem, we apply the parameterized complexity paradigm and obtain three new algorithms for $k$-means clustering of incomplete data: one for the clustering of bounded-domain (i.e., integer) data, and two incomparable algorithms that target real-valued data. Our approach is based on exploiting structural properties of a graphical encoding of the missing entries, and we show that tractability can be achieved using significantly less restrictive parameterizations than in the complementary case of few missing entries.

1. Introduction

$k$-means clustering is a fundamental task in machine learning and data analysis, one that has been the focus of extensive empirical as well as theoretical research. In general, the aim in $k$-means clustering is to partition the rows in an input matrix $A$ into $k$ clusters and compute one center per cluster so as to minimize the within-cluster sum of squares—or, when viewed as a decision problem, achieve a within-cluster sum of squares of at most a specified target value $\ell$. Depending on the setting, $A$ could be real-valued (Lloyd, 1982; Moshkovitz et al., 2020; Fomin et al., 2021) or contain integers from a bounded domain; for instance, Fomin et al. (Fomin et al., 2018) studied the case where $A$ was binary, and Ganian et al. (Ganian et al., 2018) the bounded-domain case where $\ell = 0$.

It is well known that $k$-means clustering is NP-complete, even when restricted to $k = 2$ clusters (Drineas et al., 2004; Aloise et al., 2009). On the theoretical side, a prominent approach used to circumvent such lower bounds is the use of the parameterized complexity paradigm, which provides a more fine-grained look into the complexity of problems. Since its inception, parameterized complexity has been successfully applied to a multitude of problems relevant to machine learning and artificial intelligence such as, e.g., Bayesian network learning (Ordyniak & Szeider, 2013; Grittemeier et al., 2021; Ganian & Korchemna, 2021), integer linear programming (Ganian & Ordyniak, 2018; 2019; Dvorák et al., 2021), principal component analysis (Simonov et al., 2019; Dahiya et al., 2021), and of course clustering (Cohen-Addad et al., 2018; Fomin et al., 2018).

Apart from classical clustering problems, one direction that has been gaining traction in recent years is the study of clustering in settings where some of the data is not known or simply irrelevant—a task that essentially combines data completion with clustering (Wang et al., 2019). Indeed, several authors have studied the parameterized complexity of various clustering problems for data with missing entries (Ganian et al., 2018; Koana et al., 2020; 2021; Eiben et al., 2021b), and Eiben et al. (Eiben et al., 2021a) obtained a fixed-parameter approximation algorithm specifically for (k-)Means Clustering with Missing Entries (MCME).

While their approaches and techniques differ, all of these works target the case where nearly all entries are known, i.e., where the unknown entries only occur in a precisely defined “sparse” way. On the other hand, very little is known about the complexity of MCME when the number of unknown entries is large; such situations are of practical relevance in, e.g., recommender systems and predictive analytics. For example, in the classical Netflix Prize challenge where the task was to predict user ratings for movies based on previous ratings, only about 1% of the user-movie pairs were originally supplied with a rating. The central mission of this

---

$^{*}$Equal contribution

$^{1}$Algorithms and Complexity Group, TU Wien, Austria

$^{2}$Institute of Informatics, University of Warsaw, Poland.

Correspondence to: Robert Ganian <rganian@gmail.com>, Thekla Hamm <thamm@ac.tuwien.ac.at>, Viktoria Korchemna <vkorchemna@ac.tuwien.ac.at>, Kirill Simonov <ksimonov@ac.tuwien.ac.at>, Karolina Okrasa <k.okrasa@mini.pw.edu.pl>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

---

$^{2}$See Preliminaries for a brief introduction.

$^{3}$The dataset is available at https://www.kaggle.com/netflix-inc/netflix-prize-data.
article is to push the boundaries of tractability for MCME to instances where known entries are sparse.

**Contribution.** A natural approach for handling instances of MCME with many missing entries would be to invert the parameterizations that have been developed for tackling instances where almost all entries are known. For instance, Eiben et al. (Eiben et al., 2021b) and Ganian et al. (Ganian et al., 2018) obtained several fixed-parameter clustering algorithms by using a parameter called the covering number, which is the minimum number of rows and columns needed to cover all the unknown entries. Equivalently, if we use an auxiliary binary matrix $W$ (the mask) to specify which entries of $A$ are relevant/known, the covering number is simply the minimum number of rows and columns needed to cover all the 0’s in $W$. It would be tempting to instead parameterize by the number of rows and columns needed to cover all the 1’s in $W$; such an “inverse covering number” would lead to fixed-parameter algorithms targeting instances where all known entries occur in a few columns and rows.

However, we show that in our setting it is in fact possible to do much better than that. Our approach is based on using the incidence graph representation of $W$, which is the graph containing one vertex for each column and for each row of $W$ and where an edge connects row $a$ with column $b$ if and only if $W[a, b] = 1$. Observe that the inverse covering number considered in the previous paragraph would simply be the size of a minimum vertex cover in $W$, while the covering number is the size of a minimum vertex cover in the complement of $W$. It is worth noting that in graph-theoretic settings, the size of a minimum vertex cover is considered a highly restrictive parameterization, one that is used primarily when more desirable parameterizations fail. On the other hand, the by far most prominent parameter used for graphs is treewidth (Robertson & Seymour, 1984), which has found ubiquitous applications throughout computer science and which is smaller (i.e., less restrictive as a parameter) than the size of a minimum vertex cover. As our first result, we use the treewidth of the incidence graph of the mask $W$ (the incidence treewidth) to prove:

**Result 1:** Bounded-domain MCME is fixed-parameter tractable when parameterized by the incidence treewidth of the mask.

Result 1 is noteworthy not only due to the use of treewidth instead of the inverse covering number, but also because incidence treewidth is the only parameter required to achieve tractability. In particular, unlike previous algorithms for clustering incomplete (bounded-domain or real-valued) data by Eiben et al. (Eiben et al., 2021a;b), Ganian et al. (Ganian et al., 2018), and Koana et al. (Koana et al., 2020; 2021), Result 1 can also be applied to instances where the number $k$ of clusters is large. As we will see later, we will be able to retain this benefit in all our algorithms.

Unfortunately, an extension of Result 1 towards real-valued instances seems difficult at this point. Indeed, a fixed-parameter algorithm for real-valued MCME parameterized by the incidence treewidth of the mask would imply, as a special case, fixed-parameter tractability parameterized by the number $d$ of columns. However, the existence of a fixed-parameter algorithm even for $k$-MEANS (corresponding to the restriction of MCME where all entries are known) when parameterized by $d$ is a long-standing open problem dating back to Inaba et al.’s celebrated XP algorithm parameterized by $k + d$ (Inaba et al., 1994).

Instead, we show that one can obtain fixed-parameter algorithms for real-valued MCME by using the treewidth of a different graph representation of $W$. In particular, we use the primal graph—a well-established counterpart to the incidence graph representation in settings such as Boolean satisfiability (Samer & Szeider, 2009), constraint satisfaction (Samer & Szeider, 2010) or integer programming (Ganian & Ordyniak, 2019). In our context, the primal graph of $W$ contains a vertex for each row of $W$, with edges connecting pairs of rows that share at least one coordinate which is known/relevant according to the mask, and the primal treewidth is simply the treewidth of this graph.

**Result 2:** Real-valued MCME is fixed-parameter tractable when parameterized by the primal treewidth of the mask.

Note that, due to the nature of the primal graph representation, the primal treewidth is a more restrictive parameter than the incidence treewidth (in the sense of being bounded on fewer classes of instances). In fact, the primal treewidth of a mask may in certain cases be unboundedly large even when its incidence graph is just a tree. It would hence be useful to also have an algorithm for real-valued MCME that can exploit properties of the incidence graph, even for properties that are more restrictive than incidence treewidth.

However, the same obstacle towards extending Result 1 to the real-valued setting also applies to all traditional structural parameters that are, intuitively, based on small vertex separators in the graph—including pathwidth (Robertson & Seymour, 1983), treedepth (Nesetril & de Mendez, 2012), the feedback vertex number e.g., (Mertzios et al., 2020) and even the vertex cover number e.g., (Gaspers & Najeebullah, 2019; Fellows et al., 2018), which is the most restrictive vertex separator based parameter. In spite of this, as our third and final result we identify a parameter of the incidence graph that yields fixed-parameter tractability of real-valued MCME: the recently introduced local feedback edge number (lfen) (Ganian & Korchemna, 2021). On a high level, lfen can be seen as an edge-based restriction of treewidth, and as a less restrictive parameter than the edge deletion distance to acyclicity (i.e., the feedback edge number).
The Complexity of k-Means Clustering when Little is Known

Result 3: Real-valued MCME is fixed-parameter tractable when parameterized by the local feedback edge number of the incidence graph of the mask.

We note that the parameters used in Results 2 and 3 are orthogonal and the gap between them may be arbitrarily large. Because of this, both results are incomparable and give rise to different tractable classes for the problem.

Statements where proofs or details are omitted due to space constraints are marked with *. A full version of the paper containing all proofs and details is provided in the appendix.

2. Preliminaries

For an integer $i$, we let $[i] = \{1, 2, \ldots, i\}$ and $[i]_0 = [i] \cup \{0\}$. On the other hand, if $p$ is an equivalence relation on domain $X$, we denote by $[p]$ the set of equivalence classes of $p$ and by $|[p]|$ the cardinality of $[p]$, i.e., the number of equivalence classes of $p$. For every $x \in X$, $[x]_p \in [p]$ is the equivalence class containing $x$.

Notation and Problem Formulation. Let $A$ be a matrix with $n$ rows and $d$ columns. We use $V_A$ and $C_A$ to denote the set of row and column indices of $A$, i.e., $V_A = [n]$ and $C_A = [d]$. Let $M_A = \max_{v \in V_A} \max_{c \in C_A} |A[v, c]|$. For two matrices $A$, $B$, we denote by $A - B$ the entry-wise subtraction of the two matrices, i.e., $(A - B)[i, j] = A[i, j] - B[i, j]$, and similarly by $A \circ B$ the entry-wise product of the two matrices, i.e., $(A \circ B)[i, j] = A[i, j] \cdot B[i, j]$. For a matrix $A \in \mathbb{R}^{n \times d}$ we denote its squared Frobenius norm by $\|A\|_F^2 = \sum_{i=1}^n \sum_{j=1}^d A[i, j]^2$. We can now formalize our problems of interest (abbreviated as MCME):

**MCME with Missing Entries**

Input: Matrix $A \in D^{n \times d}$ over a domain $D \subseteq \mathbb{R}$, binary matrix $W$ (the mask), integers $k$ and $\ell$.

Task: Determine if there exists a matrix $B$ over $D$ containing at most $k$ distinct rows such that $\|W \circ (A - B)\|_F^2 \leq \ell$.

We distinguish between BOUNDED-DOMAIN MCME and REAL-VALUED MCME depending on whether $D = \{z\}_0$ for some fixed integer $z$, or $D$ is the set of reals. In particular, for $z = 1$ this coincides with BINARY k-MEANS CLUSTERING (Fomin et al., 2018). In this case, minimizing the Frobenius norm also minimizes the sum of Hamming distances for each row to its cluster center. Using the Frobenius norm also for $z > 1$ is not only consistent with the continuous case, but also allows us to capture further applications, e.g., when the data matrix to cluster is composed of user ratings each taking a small numeric value. While we state MCME as a decision problem for complexity-theoretic reasons, every algorithm presented here is constructive and can output a solution matrix $B$ if one exists.

In this formulation, the distinct rows of $B$ are the centers of the sought-after clusters while $\ell$ is the target upper-bound on the sum of squares. For $V' \subseteq V_A$, we call a mapping $\varphi : V' \rightarrow [k]$ a cluster-assignment w.r.t. $B$ if for every $v_1, v_2 \in V'$ such that $\varphi(v_1) = \varphi(v_2)$, it holds that $B[v_1] = B[v_2]$. Notice that the existence of a cluster assignment w.r.t. a matrix $B'$ implies that $B'$ has at most $k$ distinct rows. The clusters associated with $\varphi$ are then sets of the form $\{v \in V : \varphi(v) = i\}$ for some $i \in [k]$.

In our algorithms, we often construct the solution matrix dynamically by “merging” smaller matrices; we formalize this procedure below. Let $\{B_i \mid i \in [m]\}$ be a set of matrices, where each $B_i$ has row labels $V_i$, column labels $C_i$ and is associated with a cluster-assignment $\varphi_i : V_i \rightarrow [k]$ and the following two consistency conditions hold:

- $\varphi_i(v) = \varphi_j(v)$ for every $v \in V_i \cap V_j$, $i, j \in [m]$,
- if $\varphi_i(v_i) = \varphi_j(v_j)$ for some $v_i \in V_i$ and $v_j \in V_j$, then $B_i[v_i, c] = B_j[v_j, c]$ for every $c \in C_i \cap C_j$.

Let $\varphi = \bigcup_{i \in [m]} \varphi_i$. We define the composition of $(B_i, \varphi_i)$, $i \in [m]$, to be the matrix $B^* \circ \varphi$ with row labels $\bigcup_{i \in [m]} V_i$, column labels $C = \bigcup_{i \in [m]} C_i$ and entries as follows. For every $v \in V$ and $c \in C$, pick any $i \in [m]$ such that $c \in C_i$ and there exists $v_i \in V_i$ with $\varphi_i(v_i) = \varphi(v)$. We set $B^*[v, c] = B_i[v_i, c_i]$. If there is no such $i \in [m]$, we simply set $B^*[v, c]$ to some arbitrary but uniform default value—in our case, we will always use 0. Observe that if both consistency conditions hold, then both $B^*$ and $\varphi$ are well-defined and that $\varphi$ is a cluster-assignment w.r.t. $B^*$.

**Parameterized Complexity.** In parameterized algorithms (Downey & Fellows, 2013; Cygan et al., 2015), the running-time of an algorithm is studied with respect to a parameter $k \in \mathbb{N}_0$ and input size $n$. The basic idea is to find a parameter that describes the structure of the instance such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (fixed-parameter tractable), which contains all problems that can be solved in time $f(k) \cdot n^{O(1)}$, where $f$ is a computable function. Algorithms with this running-time are called fixed-parameter algorithms.

**Sparse Matrices.** It follows from definition of MCME that entries of the input matrix $A$ and the target matrix $B$ are only relevant when the respective entry of the mask $W$ is set to “1”. We thus assume that only those entries are stored explicitly; in particular, the size of the input is therefore linear in the number of non-zero entries of $W$. All the structural parameters in our considerations automatically imply that the mask $W$ is sparse, in the sense that it has at most $O_k(n + d)$ non-zero entries, where the constant under $O_k(\cdot)$ depends on the respective parameter $k$. All algorithms in this paper except one are linear-time in the number of
non-zero entries of the mask, and the above allows us to state their running times in the form \((n + d) \cdot f(k)\), where \(f\) is a certain function of the respective parameter \(k\).

**Tree-width.** A nice tree-decomposition \(T\) of a graph \(G = (V, E)\) is a pair \((T, \chi)\), where \(T\) is a tree (whose vertices we call nodes) rooted at a node \(r\) and \(\chi\) is a function that assigns each node \(t\) a set \(\chi(t) \subseteq V\) such that:

- For every edge \(uv \in E\) there is a node \(t\) where \(u, v \in \chi(t)\).
- For every vertex \(v \in V\), the set of nodes \(t\) satisfying \(v \in \chi(t)\) forms a subtree of \(T\).
- \(|\chi(t)| = 1\) for every leaf \(t\) of \(T\) and \(|\chi(r)| = 0\).
- There are only three kinds of non-leaf nodes in \(T\):
  - **Introduce node:** a node \(t\) with exactly one child \(t'\) such that \(\chi(t) = \chi(t') \cup \{v\}\) for some vertex \(v \notin \chi(t')\).
  - **Forget node:** a node \(t\) with exactly one child \(t'\) such that \(\chi(t) = \chi(t') \setminus \{v\}\) for some vertex \(v \in \chi(t')\).
  - **Join node:** a node \(t\) with two children \(t_1, t_2\) such that \(\chi(t) = \chi(t_1) \cup \chi(t_2)\).

The width of a nice tree-decomposition \((T, \chi)\) is \(\max_{t \in V(T)}(|\chi(t)| - 1)\), and the tree-width of the graph \(G\), denoted \(\text{tw}(G)\), is the minimum width of a nice tree-decomposition of \(G\). We let \(T_t\) denote the subtree of \(T\) rooted at a node \(t\), and we use \(\chi_t\) to denote the set \(\bigcup_{t' \in V(T_t)} \chi(t')\). The sets \(\chi(t)\) are commonly called bags.

Fixed-parameter algorithms are known for computing nice tree-decompositions of optimal width and linearly many nodes (Bodlaender, 1996; Kloks, 1994), albeit more efficient fixed-parameter approximation algorithms are often used to achieve better running times (Bodlaender et al., 2016).

**Local Feedback Edge Number.** The local feedback edge number was recently introduced by Ganian and Korchemna (2021) as an edge-cut based restriction of treewidth that acts as a “local” measure of the size of a feedback edge set; they used the parameter to obtain fixed-parameter algorithms for learning Bayesian networks and polytrees.

For a connected graph \(G = (V, E)\) and a spanning tree \(T\) of \(G\), the local feedback edge set at \(x \in V\) is defined as \(E_{\text{loc}}^T(x) = \{yz \in E \setminus E(T) \mid \text{the unique path between } y \text{ and } z \text{ in } T \text{ contains } x\}\). The local feedback edge number of \((G, T)\) (denoted \(\text{lfen}(G, T)\)) is then equal to \(\max_{x \in V} |E_{\text{loc}}^T(x)|\), and the local feedback edge number of \(G\) is simply the smallest local feedback edge number among all possible spanning trees of \(G\), i.e., \(\text{lfen}(G) = \min_T \text{lfen}(G, T)\) if it is a spanning tree of \(G\). An example of a graph \(G\) with \(\text{lfen}(G) = 2\) is provided in Figure 1.

**Figure 1.** Example of a graph \(G\) with the spanning tree \(T\) (marked in thick red) such that \(\text{lfen}(G) = \text{lfen}(G, T) = 2\). The feedback edge number of \(G\), i.e., its edge deletion distance to acyclicity, is exactly the number of black edges and can be made arbitrarily large in this fashion while preserving \(\text{lfen}(G) = 2\).

3. **Solving Bounded-Domain MCME by Exploiting the Mask**

Since **bounded-domain** MCME is NP-complete (Drineas et al., 2004; Aloise et al., 2009), it is natural to attempt and circumvent this lower bound by exploiting structural measures of the inputs to obtain fixed-parameter tractability. Eiben et al. (Eiben et al., 2021a) recently obtained a fixed-parameter approximation algorithm for **bounded-domain** requiring the simultaneous parameterization by three measures: the number of clusters, the desired approximation ratio, and a technical measure that captures the sparsity of the missing data (i.e., the “0” entries in \(W\)).

In this section, we will present a fixed-parameter algorithm for **bounded-domain** MCME which is aimed at the complementary case where the relevant data is sparse. To formalize this, we consider the incidence graph representation \(G_I\) of \(W\) which is defined as follows: \(V(G_I) = V_W \cup C_W\), and \(E(G_I) = \{ab \mid a \in V_W, b \in C_W, W[a, b] = 1\}\). Figure 2 later on can also be used as an example of the representation.

Intuitively, each “1” entry in \(W\) will correspond to an edge in \(G_I\), and hence “structurally sparse” incidence graphs correspond to settings where most data is unknown. There is a well-studied hierarchy of structural graph parameters which measure, in a certain sense, the sparsity of graphs (see, e.g., Figure 1 in (Bodlaender et al., 2013)). Treewidth will be our parameter of choice here, as the best known and also most general parameter in this hierarchy. However, it is worth noting that the use of the incidence graph may provide a new perspective on previous algorithms for clustering problems—for instance, the so-called covering number parameter (Ganian et al., 2018; Eiben et al., 2021b) is simply the vertex cover number of the complement of \(G_I\).

**Theorem 1.** **Bounded-Domain** MCME is fixed-parameter tractable when parameterized by \(k + \text{tw}(G_I)\).
Proof. We begin by applying the well-established 5-approximation algorithm for treewidth (Bodlaender et al., 2016) to compute a nice tree-decomposition of $G_I$ of width $q \leq 5 \tw(G_I)$ in time $2^q(n + d)$. Let $r$ be the root of $T$. Given a node $t$ of $T$, let $V_t$ and $C_t$ be the sets of vectors and coordinates in $\chi_t$, respectively. Moreover, let $C^t_t$ and $V^t_t$ be the restrictions of $\chi^t_t$ to sets of coordinates and vectors correspondingly.

To prove the theorem, we will design a leaf-to-root dynamic programming algorithm which will compute and store a set of records at each node $t$ of $T$, whereas once we ascertain the records for $r$ we will have the information required to output a correct answer. Intuitively, the records will store the cluster centers restricted to the coordinates that appear in the bag, the partition of the vectors in the bag into clusters and the sum of minimum distances from vectors in $V^t_t$ to the cluster centers along the coordinates in $C^t_t$.

Formally, the records will have the following structure. We call a pair (cent, part) a snapshot in $t$ if the following holds:

- cent : $[k] \times C_t \rightarrow D$,
- part : $V_t \rightarrow [k]$.

Let $S(t)$ be the set of all snapshots of $t$. The record $R_t$ of $t$ is then a mapping from $S(t)$ to the set $\mathbb{R}^+$ of non-negative reals. Observe that $|S(t)| \leq |D|^{k(q+1)kq+1}$.

To introduce the semantics of our records, let $B_t$ be the set of all matrices with row labels $V^t_t$, column labels $C^t_t$ and entries of domain $D$. Let $B_t$ be a matrix in $B_t$. We define the partial weighted distance from $B_t$ to $A$ in $t$ as follows:

$$pwd(B_t, t) = \sum_{v \in V^t_t} \sum_{c \in C^t_t} W[v, c] \cdot (A[v, c] - B_t[v, c])^2.$$  

For $B_t \in B_t$, we say that (cent, part) $\in S(t)$ is the snapshot of $B_t$ in $t$ if there is a cluster-assignment $\varphi$ w.r.t. $B_t$ such that the following conditions hold:

- part $= \varphi|_V$,
- for every $c \in C_t$ and $v \in V^t_t$, $B_t[v, c] = cent[\varphi(v), c]$.

Recall that the existence of a cluster-assignment implies, in particular, that $B_t$ has at most $k$ distinct rows. We are now ready to define the record $R_t$. For each snapshot (cent, part) $\in S(t)$, we set $R_t(cent, part) = \tau$ if there exists $B_t \in B_t$ such that:

- (cent, part) is the snapshot of $B_t$ in $t$,
- $pwd(B_t, t) = \tau$, and
- $V^t_t \in B_t$ such that (cent, part) is the snapshot of $B'_t$, $pwd(B'_t, t) \geq \tau$.

In this case we say that $B_t$ witnesses $R_t(cent, part) = \tau$.

Recall that for the root $r$, we assume $\chi(r) = \emptyset$. Hence $S(r)$ contains only one element ($\emptyset$, $\emptyset$), and $R_r(\emptyset, \emptyset) = 0$ is equal to the minimum value of $\|W \cdot (A - B)\|_F^2$ that can be achieved by any matrix $B$ with entries of domain $D$ containing at most $k$ distinct rows. In other words, the instance is a YES-instance if and only if $R_r(\emptyset, \emptyset) \leq 0$. To prove the theorem, it now suffices to show that the records can be computed in a leaf-to-root fashion by proceeding along the nodes of $T$.

We distinguish the following cases:

- **t is a leaf node.** Let $t = \{v\}$ where $v$ is a vector. By definition, $S(t) = \{(\emptyset, part) : [1] \rightarrow [k]\}$ and $R_t(\emptyset, part) = 0$ for every $(\emptyset, part) \in S(t)$ as there are no coordinates to sum over. Let $\chi(t) = \{c\}$ for some coordinate $c$. Then $S(t) = \{(\emptyset, part) : [k] \times [1] \rightarrow D\}$ and $R_t(\emptyset, \emptyset) = 0$ for each $(\emptyset, \emptyset) \in S(t)$.

- **t is a forget node.** Let $t'$ be the child of $t$ in $T$ and $\chi(t) = \chi(t') \setminus \{c\}$ for some vector $v$. We set $R_t(cent, part) := \min_{v,i \in V_t} \chi(cent, part \cup (v, i))$ for each $(cent, part) \in S(t)$. For correctness, it will be useful to observe that $R_t(\emptyset, \emptyset) = \tau$, if $R_{t'}(cent, part) = \tau$, then there exists a matrix $B_t$ which witnesses this. But then $B_t$ also admits a snapshot (cent, part $\cup (v, i)$) at $t'$ for some $i \in [k]$ and witnesses $R_{t'}(cent, part \cup (v, i)) \leq \tau$. So in our algorithm $R_t(cent, part) \leq R_{t'}(cent, part \cup (v, i)) \leq \tau$. On the other hand $R_t(cent, part) = \tau$, then there exists a snapshot (cent, part $\cup (v, i)$) at $t'$ for some $i \in [k]$ such that $R_{t'}(cent, part \cup (v, i)) = \tau$. Hence, we can correctly set $R_t = R_t'$. If $\chi(t) = \chi(t') \setminus \{c\}$ for some coordinate $c$, we set $R_t(cent, part)$ equal to $\min_{e \in R_{t'}(cent, part)}$ is a restriction of $cent'$ to all coordinates except $c$. Correctness can be argued similarly to the case of a forgotten vector.

- **t is an introduce node (introducing a vector).** Let $t'$ be the child of $t$ in $T$ and let $\chi(t) = \chi(t') \cup \{v_0\}$ for some vector $v_0$. Fix a snapshot (cent, part) in $S(t)$, let $s = \text{part}(v_0)$ and $s' = \text{part}(\emptyset, t)$. We will denote by $\Delta_0$ the sum of distances from $v_0$ to the $i$-th cluster center along the coordinates in $C_t$, i.e., $\Delta_0 = \sum_{c \in C_t} W[v_0, c] \cdot (A[v_0, c] - cent[i, c])^2$. We set $R_t(cent, part) = R_{t'}(cent, part') + \Delta_0$. For correctness, assume that $R_t(cent, part) = \tau = R_{t'}(cent, part') + \Delta_0$. Construct a matrix $B_t$ from the witnesses $B_{t'}$ of $R_{t'}(cent, part')$ with a cluster-assignment $\varphi'$ by adding a new row with a label $v_0$ as follows. If $i = \varphi'(v)$ for some $v \in V_t$, we set $B_t[v_0, c] := B_{t'}[v, c]$. Otherwise, we define $B_t[v_0, c] := cent[i, c]$ for $c \in C_t$ and $B_t[v_0, c] := 0$ in the rest of coordinates $c$. Note that in both cases $B_t[v_0, c] = cent[i, c]$ for every $c \in C_t$; in the second case it follows from the definition, while in the first one we have $B_t[v_0, c] := B_{t'}[v, c] = cent[\varphi'(v), c] = cent[i, c]$. The matrix $B_t$ with a cluster-
assignment $\varphi = \varphi' \cup \{(v_0, i)\}$ has a snapshot (cent, part) in $t$ and $\pwd(B_t, t) = \sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] \cdot \varphi_B[v, c])^2$ is $\pwd(B_t, t)^2 = \pwd(B_t, t') + \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] \cdot \varphi_B[v, c])^2$.

As $\pwd(B_t, t') = \mathcal{R}_v(\text{cent, part})$, $W[v, c] = 0$ for every forgotten coordinate $c$ and $B_t[v_0, c] := \text{cent}[i, c]$ for every $c \in C_t$, we have $\pwd(B_t, t) = \mathcal{R}_v(\text{cent, part'}) + \Delta_0 = \tau$. So $B_t$ witnesses that $\mathcal{R}_v(\text{cent, part}) \leq \tau$.

On the other hand, assume that $\mathcal{R}_v(\text{cent, part}) = \tau$. Then there exists a matrix $B_t$ in $B_t$ admitting the snapshot (cent, part) in $t$ such that $\pwd(B_t, t) = \tau$. Let $\varphi$ be the corresponding cluster-assignment w.r.t. $B_t$. We construct $B_t'$ from $B_t$ by deletion of the row $v_0$. Then $B_t'$ with cluster-assignment $\varphi' = \varphi \setminus \{(v_0, i)\}$ has a snapshot (cent, part') in $t$ and witnesses that $\mathcal{R}_v(\text{cent, part'}) \leq \pwd(B_t, t') + \Delta_0 = \tau$. Therefore in our algorithm $\mathcal{R}_0^\tau \leq \mathcal{R}_v(\text{cent, part'}) + \Delta_0 \leq \tau$. Hence, we can correctly set $\mathcal{R}_v = \mathcal{R}_0^\tau$.

$t$ is an introduce node (introducing a coordinate). Let $t'$ be the child of $t$ in $T$ and let $\chi(t) = \chi(t') \cup \{(c_0, 0)\}$ for some coordinate $c_0$. Fix a snapshot (cent, part) in $S(t)$, we will denote by $\Delta_0$ the sum of distances from $v \in V_t^+$ to the corresponding cluster center along the coordinate $c_0$, i.e., $\Delta_0 = \sum_{v \in V_t^+} W[v, c_0] \cdot (A[v, c_0] \cdot \text{cent}[\text{part}(v), c_0])^2$. Let $\varphi'$ be the restriction of cent to all the coordinates except $c_0$. We set $\mathcal{R}_v^\tau(\text{cent, part}) = \mathcal{R}_v(\text{cent, part'}) + \Delta_0$. For correctness, assume that $\mathcal{R}_v^\tau(\text{cent, part}) = \tau = \mathcal{R}_v(\text{cent, part'}) + \Delta_0$. We construct a matrix $B_t$ from a witness $B_{t'}$ of $\mathcal{R}_v(\text{cent, part'})$ with cluster-assignment $\varphi'$ by adding a column $c_0$. For every $v \in V_t^+$, we set $B_t[v, c] = B_{t'}[v, c]$ for all $c \in C_t^i$ and $B_t[v, c_0] = \text{cent}[\varphi'(v), c_0]$. Then $B_t$ with the same cluster-assignment $\varphi'$ has a snapshot (cent, part) in $t$ and $\pwd(B_t, t) = \pwd(B_{t'}, t') + \Delta_0 = \mathcal{R}_v(\text{cent, part'}) + \Delta_0 = \tau$. Therefore, $B_t$ witnesses that $\mathcal{R}_v(\text{cent, part}) \leq \tau$. On the other hand, assume that $\mathcal{R}_v(\text{cent, part}) = \tau$. Then there exists a matrix $B_t$ in $B_t$ with the snapshot (cent, part) in $t$ such that $\pwd(B_t, t) = \tau$. Let $B_t'$ be obtained from $B_t$ by deletion of the column with the label $c_0$. Then $B_t'$ witnesses $\mathcal{R}_v(\text{cent, part'}) \leq \tau - \Delta_0$, so in our algorithm $\mathcal{R}_v^\tau(\text{cent, part}) = \mathcal{R}_v(\text{cent, part'}) + \Delta_0 \leq \tau$. Hence, we can correctly set $\mathcal{R}_v = \mathcal{R}_0^\tau$.

$t$ is a join node. Let $t_1, t_2$ be the two children of $t$ in $T$, recall that $\chi(t_1) = \chi(t_2) = \chi(t)$ and $\chi^i_t \cap \chi^i_{t'} = \chi(t)$. For every (cent, part) in $S(t)$ we set $\mathcal{R}_v^\tau(\text{cent, part}) = \mathcal{R}_{t_1}(\text{cent, part}) + \mathcal{R}_{t_2}(\text{cent, part}) - \text{doublecount, where doublecount} = \sum_{v \in V_t} \sum_{c \in C_t} W[v, c] \cdot (A[v, c] \cdot \text{cent}[\text{part}(v), c])^2$.

For correctness, assume that $\mathcal{R}_v^\tau(\text{cent, part}) = \tau = \tau_1 + \tau_2 - \text{doublecount}$, where $\tau_1 = \mathcal{R}_{t_1}(\text{cent, part})$, $\tau_2 = \mathcal{R}_{t_2}(\text{cent, part})$. Let $B_t$ with the cluster-assignment $\varphi$, be the witness of $\mathcal{R}_t(\text{cent, part}) = \tau$, $i = 1, 2$. We obtain a matrix $B_t$ with cluster-assignment $\varphi = \varphi_1 \cup \varphi_2$ as a composition of $(B_1, \varphi_1)$ and $(B_2, \varphi_2)$. To check the consistency conditions, observe that the sets of common row and column labels of $B_1$ and $B_2$ are $V_t$ and $C_t$ correspondingly. Recall that $\varphi_1[v], \varphi_2[v]$ = $\varphi_2[v]$. Moreover, if $v_1 \in V_t$ and $v_2 \in V_t$ are such that $\varphi_1(v_1) = \varphi_2(v_2)$, then we have $B_1[v_1, c] = \text{cent}[\varphi_1(v_1), c] = \text{cent}[\varphi_2(v_2), c] = B_2[v, c]$ for every $c \in C_t$.

Note that for every $v \in V_t$, $\varphi(v) = \varphi_1(v) = \varphi_2(v)$. Pick $c \in C_t$ and $v \in V_t^+$, then $v \in V_t^+$ for some $i \in [1, 2]$ and $\varphi(v) = \varphi_1(v) = \varphi_2(v)$. Therefore, $(\text{cent, part})$ is a snapshot of $B_t$ in $t$. Recall that $\pwd(B_t, t) = \sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] - B_t[v, c])^2$. Here $W[v, c] = 0$ for every $v \in V_t^+$ and $c \not\in C_t^i$, $i = 1, 2$. So $\pwd(B_t, t) = \sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 + \sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 = \pwd(B_{t_1}, t_1) + \pwd(B_{t_2}, t_2) - \text{doublecount} = \tau_1 + \tau_2 - \text{doublecount} = \tau$. Hence $B_t$ witnesses $\mathcal{R}_t(\text{cent, part}) \leq \tau$.

For the converse, assume that $\mathcal{R}_t(\text{cent, part}) = \tau$ and $B_t$ is a matrix witnessing this. Let $B_i$ be the restriction of $B_t$ to rows $V_t^i$ and columns $C_t^i$, $i = 1, 2$. Then $B_1$ and $B_2$ have a snapshot (cent, part) and in our algorithm $\mathcal{R}_v^\tau(\text{cent, part}) = \mathcal{R}_{t_1}(\text{cent, part}) + \mathcal{R}_{t_2}(\text{cent, part}) - \text{doublecount} \leq \pwd(B_1) + \pwd(B_2) - \text{doublecount} = \pwd(B) = \tau$. Hence the resulting record $\mathcal{R}_t = \mathcal{R}_0^\tau$ is correct, which concludes the correctness proof of the algorithm.

To bound the runtime of the algorithm, observe that at each node $t$ we compute the record $\mathcal{R}_t$ for $|S(t)| \leq |D|^{k|\mathcal{G}|} = |D|^{O(k\cdot \text{tw}(G))}$ entries, where each entry is computed in time at most quadratic in $\text{tw}(G)$. Since the tree-decomposition is nice and has at most $O(n + d)$ nodes, the runtime of the algorithm is upper-bounded by $(n + d) \cdot |D|^{O(k\cdot \text{tw}(G))}$.

Towards proving fixed-parameter tractability without involving $k$, we consider the case where $k$ is significantly larger than $\text{tw}(G)$. We prove that this case always admits a "perfect" solution, which can also be computed efficiently.

**Lemma 2 (⋆).** If $k \geq (\text{tw}(G) + 1) \cdot |D|^{\text{tw}(G) + 1}$ then there exists a matrix $B$ over domain $D$ containing at most $k$ distinct rows such that $\|W \circ (A - B)\|^2_F = 0$. Moreover, given a nice tree-decomposition of width $q$ such that $k \geq q \cdot |D|^q$, such a matrix $B$ can be computed in time $(n + d) \cdot |D|^{O(q)}$.

**Proof Sketch.** We will prove the second claim of the lemma, from which the first one follows. To this end, let us assume that we are given a nice tree-decomposition $\mathcal{T} = (T, \chi)$ of
At this point, we can prove the main result of this section. We denote the root of $T$ by $r$ and define $C_t$, $V_t$, $C_t^i$ and $V_t^i$ analogously to the proof of Theorem 1. We will explicitly construct $q \cdot |D|^q$ cluster centers yielding a zero sum of squares, so that after processing the node $t$, each of the centers is defined on coordinates $c \in C_t^i$. The remaining clusters will be empty, thus for the rest of the proof we assume $k = q \cdot |D|^q$.

Formally, we will attach to every node $t$ the pair $(R_t, \alpha_t)$ where $R_t$ is the matrix with the row labels $[k]$ and the column labels $C_t^i$, and $\alpha_t$ is a mapping from $V_t^i$ to $[k]$. Intuitively, $R_t$ stores the restrictions of cluster centers to processed coordinates, while $\alpha_t$ assigns a cluster index to every row label $v \in V_t^i$. Throughout the dynamic programming procedure, we will maintain the following invariants:

1. $\alpha_t(u) \neq \alpha_t(v)$ for any two distinct $u$ and $v$ from $V_t$,
2. for every $\omega \in D^{C_t}$, there are precisely $\frac{k}{|D|^{|C_t|}}$ indices $i \in [k]$ such that $R_t[i][C_t] = \omega$, where $R_t[i][C_t]$ is the restriction of $R_t[i]$ to the coordinate set $C_t$,
3. $\sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] - R_t[\alpha_t(v), c])^2 = 0$.

To complete the proof, it remains to show that the records $(R_t, \alpha_t)$ can be computed in a leaf-to-root fashion via dynamic programming along the nodes of the tree-decomposition. Indeed, once we obtain the record $(R_r, \alpha_r)$ for the root node $r$, we can define the solution matrix $B$ by setting $B[v] = R_r[\alpha_r(v)]$; this matrix has at most $k$ distinct rows and the third invariant yields that $\sum_{v \in V_A} \sum_{c \in C_A} W[v, c] \cdot (A[v, c] - B[v, c])^2 = 0$. \hfill $\square$

At this point, we can prove the main result of this section.

**Theorem 3.** **Bounded-Domain MCME** is fixed-parameter tractable when parameterized by $\text{tw}(G_I)$.

**Proof.** Given an instance of Bounded-Domain MCME, we begin by constructing the incidence graph $G_I$ and then applying the known 5-approximation algorithm to compute a tree-decomposition of $G_I$ of width $q - 1 \leq 5 \cdot \text{tw}(G_I)$ in time $2^{O(\text{tw}(G_I))} \cdot (n + d)$. At this point we proceed by comparing $q$ and $k$, as follows:

- if $k \geq q \cdot |D|^q$, then we can correctly output “Yes” (along with a suitable witness) by using Lemma 2;
- on the other hand, if $k \leq q \cdot |D|^q$ then we instead invoke Theorem 1 to solve the instance.

The running time in the former case is upper-bounded by $(n + d) \cdot |D|^{O(\text{tw}(G_I))}$, while in the latter case the bound is $(n + d) \cdot |D|^{O(\text{tw}(G_I))} \leq (n + d) \cdot |D|^{(k \cdot \text{tw}(G_I))}$. \hfill $\square$

### 4. Handling Real-Valued Matrices with Treewidth

We now turn our attention to Real-Valued MCME. As was mentioned in the Introduction, a fixed-parameter algorithm for the problem parameterized by the incidence treewidth of $W$ would resolve a long-standing open problem as a special case, and so far remains beyond reach. However, we can still make tangible progress on the problem by considering the treewidth of a different representation of $W$. In particular, the primal graph $G_P$ of $W$ is a graph containing one vertex for each row of $W$, where two rows $a, b$ are adjacent if and only if there is a column $c$ such that $W[a, c] = W[b, c] = 1$. An example comparing primal and incidence graphs is provided in Figure 2.

Our aim in this section is to prove the next theorem.

**Theorem 4 (⋆).** Real-Valued MCME is fixed-parameter tractable when parameterized by $\text{tw}(G_P)$.

Before we proceed to the proof, we will first introduce some additional notation that will be useful in the context of real-valued matrices. For every $V' \subseteq V_A$, let $\mu(V', c) = \min_{x \in A} W[x, c] \cdot c - A[x, c] \cdot (A[i, c] - x)^2$ and let $\Delta_c(V') = \mu(\mu(V', c), c)$. Intuitively, if vectors from $V'$ form a cluster then $\Delta_c(V')$ is a minimal sum of distances to the cluster center along the coordinate $c$.

**Observation 5 (⋆).** For each coordinate $c$, if $V'' = \{z \in V' \mid W[z, c] = 1\} \neq \emptyset$ then

\[
\mu(V', c) = \sum_{v \in V''} A[v, c] \quad \text{and} \quad \Delta_c(V') = \sum_{v \in V''} (A[v, c] - 1)^2 \cdot \frac{\sum_{v \in V''} A[v, c]^2}{|V''|}.
\]

Now we are ready to present the proof of Theorem 4.

**Proof Sketch for Theorem 4.** As our initial step, we once again apply the known approximation algorithm (Bodlaender et al., 2016) to compute a nice tree-decomposition $(T, \chi)$.
of $G_P$ of width $q \leq 5 \text{tw}(G_P)$. We keep the notations $r$, $V_t$, and $V_t^+$ from the previous section. Note that now the bags in the tree-decomposition contain only vectors, so $V_t = \chi(t)$. In contrast to the previous section, for every node $t$ we denote by $C_t^+$ the set of all processed in $t$ coordinates, i.e., coordinates $c$ such that $W[v, c] \neq 0$ for some $v \in V_t^+ \setminus V_t$. A crucial observation is the following: if $c \in C_t^+$, then for all vectors $v \notin V_t^+$ it holds that $W[v, c] = 0$, otherwise $v$ and $v_0$ would appear in the same bag. In other words, values of the cluster centers on $c \in C_t^+$ are not meaningful for vectors introduced outside of $T_t$. We will design a leaf-to-root dynamic programming algorithm which computes a set of records $R_t$ at each node $t$ of $T$. For each way $p$ of partitioning the vectors in the bag $V_t$ into at most $k$ clusters, the record $R_t$ stores the minimum cost of clustering the vectors of $V_t^+$ in the coordinates of $C_t^+$, considering only the partitions of $V_t^+$ that extend $p$.

For the root $r$ of $T$, $R_r(\emptyset)$ is equal to the desired value $\min_{\hat{\delta}} ||W \circ (A - B)||_F^2$, so it suffices to show how to compute the records at each node in $T$ from the records of its children. Let $t$ be an introduce node with a child $t'$ such that $V_t = V_{t'} \cup \{v\}$, denote by $p'$ the restriction of $p$ to $V_{t'}$. We set $R_t(p) = R_{t'}(p')$ since $C_t^+ = C_{t'}^+$ and $W[v, c] = 0$ for each $c \in C_t^+$. If $t$ is a join node with children $t_1$ and $t_2$, observe that $W[v, c] = 0$ for each $v \in V_{t_1}^+ \setminus V_t$, $c \in C_{t_2}^+$, and for each $v \in V_{t_2}^+ \setminus V_t$, $c \in C_{t_1}^+$. Furthermore, $C_t^+$ is necessarily disjoint from $C_{t_1}^+ \cup C_{t_2}^+$, and $R_t(p) = R_{t_1}(p) + R_{t_2}(p)$.

Finally, for a forget node $t$ with a child $t'$ such that $V_t \cup \{v\} = V_{t'}$, branch on all admissible partitions $p'$ of $V_{t'}$ that extend $p$. Start with $\Delta(p') = 0$, for each coordinate $c \notin C_{t'}^+$ such that $W[v, c] = 1$, and for each equivalence class $[x]_{p'} \in [p']$, add $\Delta_c([x]_{p'})$, the cost of clustering the rows $[x]_{p'}$ in the column $c$, to $\Delta(p')$. Since $C_{t'}^+ \setminus C_t^+$ are precisely the coordinates above, and no row outside of the bag $V_t$ contributes to these coordinates, the new partial weighted distance is $R_{t'}(p') + \Delta(p')$, assuming that $p'$ is the right choice. Therefore, we set $R_t(p) = \min_{p'}(R_{t'}(p') + \Delta(p'))$.

As for the running time, in each step there are at most $\text{tw}(G_P)^{O(\text{tw}(G_P))}$ partitions of the bag and hence the total running time required to process all forget nodes can in total be upper-bounded by $d \cdot \text{tw}(G_P)^{O(\text{tw}(G_P))} \leq (n + d) \cdot \text{tw}(G_P)^{O(\text{tw}(G_P))}$. The latter bound can then easily be shown to hold for all other kinds of nodes.

5. An Incidence-Graph Based Algorithm for Real-Valued MCME

While Theorem 4 significantly expands the previously known boundaries of tractability for MCME, the algorithm’s performance strongly depends on the structural properties of the primal graph. In general, primal graph representations are known to be denser than incidence graph representations, and this may make them unsuitable for the application of structure-based algorithms on certain instances (see, e.g., the example below Theorem 6).

As our final result, we show that although a fixed-parameter algorithm for Real-Valued MCME parameterized by $\text{tw}(G)$ remains beyond our reach, we can exploit a different parameter of the incidence graph to achieve fixed-parameter tractability—namely, the local feedback edge number.

**Theorem 6 (⋆).** Real-Valued MCME is fixed-parameter tractable when parameterized by $\text{lfen}(G_I)$.

We note that it is not difficult to show that $\text{lfen}(G_I)$ and $\text{tw}(G)$ are pairwise incomparable parameterizations. Indeed, an $n \times 1$ mask consisting only of “1” entries has $\text{lfen}(G_I) = 0$ but $\text{tw}(G) = n - 1$. On the other hand, consider, for some integer $m$, an $(m + 1) \times 2m$ mask $W$ such that the first row $v_0$ of $W$ consists only of “1” entries, while for each $i \in [m]$ the $(i + 1)$-th row has “1” entries on precisely two positions: $i$ and $i + m$. Then $G_P$ is a star with center in $v_0$, so $\text{tw}(G) = 1$. However, $G_I$ consists of $m$ edge-disjoint cycles intersecting in $v_0$. It is then easy to observe that the local feedback edge number of $v_0$ with respect to any spanning tree of $G_I$ is $m$.

We proceed by introducing some additional terminology that will be useful in the coming arguments. Let $T$ be a fixed rooted spanning tree of $G$ such that $\text{lfen}(G, T) = \text{lfen}(G)$, denote the root by $r$. For $t \in V(T)$, let $T_t$ be the subtree of $T$ rooted at $t$. We define the boundary $\delta(t)$ of $t$ to be the set of endpoints of all edges in $G$ with precisely one endpoint in $V(T_t)$ (observe that the boundary can never have a size of 1). $t$ is called closed if $|\delta(t)| \leq 2$ and open otherwise.

**Proposition 7 (Ganian and Korchevna (2021)).**

1. For every closed child $t'$ of $t$ in $T$, it holds that $\delta(t') = \{t, t'\}$ and $\ell t'$ is the only edge between $V(T_{t'})$ and $V(G) \setminus V(T_t)$ in $G$.
2. $|\delta(t)| \leq 2 \text{lfen}(G) + 2$.
3. Let $\{t_i | i \in [j]\}$ be the set of all open children of $t$ in $T$. Then $j \leq 2 \text{lfen}(G)$ and $\delta(t) \subseteq \bigcup_{i=1}^{j} \delta(t_i) \cup \{t\} \cup N^G(t)$.

To prove Theorem 6, we will provide a leaf-to-root dynamic programming algorithm which stores information about optimal partitionings of $\delta(t)$ into clusters. On a very intuitive level, Point 1 of Proposition 7 allows the algorithm to handle the closed children in a greedy manner, Point 2. ensures that the size of the records is bounded, and Point 3. is used in the dynamic programming step to compute records for a parent based on the records of its children. Furthermore, one can observe that $|\bigcup_{i=1}^{j} \delta(t_i)|$ is upper-bounded by a
linear function of \(\lfen(t)\), which will be useful to ascertain the runtime bound for the algorithm.

**Observation 8.** For each node \(t\) in \(T\), \(|\bigcup_{i=1}^j \delta(t_i)| \leq 4\lfen(t) + 2\).

**Proof.** By Point 2. of Proposition 7, the number of vertices in \(\bigcup_{i=1}^j \delta(t_i)\) that belong to \(\delta(t)\) is at most \(|\delta(t)| \leq 2\lfen(G) + 2\). If \(v \in \delta(t_i) \setminus \delta(t)\) for some \(i \in [j]\), then \(v = t_i\). So the number of such \(v\) is at most \(j \leq 2\lfen(G)\) by Point 3. of Proposition 7. In total, \(|\bigcup_{i=1}^j \delta(t_i)| \leq 4\lfen(t) + 2\).

We are now ready to establish the claimed result by providing an algorithm for **REAL-VALUED** MCMC with running time \((kn^2 + d) \cdot \lfen(G)\), assuming a spanning tree \(T\) of minimum local feedback edge number is provided. We note that such a spanning tree can be computed by a fixed-parameter algorithm (Ganian & Korchemna, 2021).

**Proof Sketch for Theorem 6.** We will design a leaf-to-root dynamic procedure computing the set of records for every node of \(T\). Let \(V_t^i\) and \(C_t^i\) be the sets of vectors and coordinates of \(T_t\) correspondingly, we denote by \(V_t\) and \(C_t\) the restrictions of \(\delta(t)\) to vectors and coordinates respectively, then \(\delta(t) = V_t \cup C_t\). Analogously to the previous theorems, we define the record \(R_t\) of \(t\) as a mapping that associates each partition \(p\) of \(V_t\) with the optimal partial cost of clustering that agrees with \(p\) on \(\delta(t)\). The partial cost is computed only on the rows of \(V_t \cup V_t^i\) and the columns of \(C_t^i\).

Recall that for the root \(r\) of \(T\), the boundary is empty, in particular \(V_r = \emptyset\), and \(R_r(\emptyset)\) is equal to the minimum value of \(\|W \circ (A - B)\|_2\), that can be achieved by any real matrix \(B\) containing at most \(k\) distinct rows. Hence the instance is a YES-instance if and only if \(R_r(\emptyset) \leq t\). Thus, it only remains to carry out the computation of the records in each node from the records of its children, and this is the most technical part of the proof.

Intuitively, for a node \(t\) we consider separately its open and closed children, denote the former by \(t_1, \ldots, t_j\), and the latter by \(t_{j+1}, \ldots, t_m\). For each partition \(p\) of \(V_t\), we enumerate all possible extensions of \(p\) to a partition \(p^*\) of \(\bigcup_{i \in [j]} V_{t_i}\); the crucial observation here is that the number of different \(p^*\)'s is upper-bounded by \(\lfen(G)\). Now, if \(t\) is a vertex node, every coordinate in \(C_t^i\) is contained in the subtree of one of the children of \(t\), and the cost of this coordinate is accounted-for in the record of the child. It thus only remains to sum up the records of the children, taking each time the respective partition on \(V_{t_i}\) for a child \(t_i\). For an open child the partition is explicitly a restriction of \(p^*\) on \(V_{t_i}\), and for a closed child, as \(\delta(t_i) = \{t, t_i\}\), \(V_{t_i} = \{t\}\) and the partition of \(V_{t_i}\) is always trivial.

More work has to be done if \(t\) is a coordinate node: we have to account for the distances in the new coordinate \(t\). For a closed child \(t_i\) there is only interaction between the coordinate \(t\) and the row \(t_i\), the rest of the subtree is independent. Therefore, the rows \(t_{j+1}, \ldots, t_m\) can be clustered in any way, to minimize the cost in the coordinate \(t\). This one-dimensional subproblem can be solved by a separate dynamic programming in time \(O(km^2)\). For the open children, the exact partitioning on the boundary vectors is again fixed, and the dynamic programming can be enhanced to respect the given partitioning on these additional vectors, at the cost of an extra \(2^{O(\lfen(G)\})}\) time factor.

### 6. Concluding Remarks

While our algorithmic results are specifically designed to deal with **MEANS CLUSTERING WITH MISSING ENTRIES**, it would be interesting to see whether the approaches and techniques developed here can be applied to other clustering variants or, e.g., the related task of low-rank matrix completion. Still, on the theoretical side the by far most prominent problem that is relevant to this research direction is the complexity of **k-MEANS CLUSTERING** for real-valued matrices when parameterized by the number of columns. A W[1]-hardness result for this problem would immediately exclude the existence of a fixed-parameter algorithm for **REAL-VALUED** MCMC parameterized by the incidence treewidth of the mask, while a fixed-parameter algorithm could potentially open up the way towards tractability.

It would also be interesting to see how the considered sparsity parameters behave in practical settings. In particular, even though direct implementations of our exact algorithms with runtime guarantees are unlikely to outperform state-of-the-art heuristics, it may be possible to exploit these parameters to guide or otherwise improve existing methods.

### Acknowledgements

Robert Ganian, Thelka Hamm, Viktoriia Korchemna, and Kirill Simonov acknowledge support by the Austrian Science Fund (FWF, projects Y1329 and P31336). Karolina Okraska acknowledges support by the European Research Council, grant agreement No 714074; parts of this work were performed while visiting TU Wien, Vienna, Austria.

### References

Aloise, D., Deshpande, A., Hansen, P., and Popat, P. 


The Complexity of k-Means Clustering when Little is Known
(Appendix: Full Version)

Robert Ganian *1 Thekla Hamm *1 Viktoriia Korchemna *1 Karolina Okrasa *2 Kirill Simonov *1

Abstract
In the area of data analysis and arguably even in machine learning as a whole, few approaches have been as impactful as the classical k-means clustering. Here, we study the complexity of k-means clustering in settings where most of the data is not known or simply irrelevant. To obtain a more fine-grained understanding of the tractability of this clustering problem, we apply the parameterized complexity paradigm and obtain three new algorithms for k-means clustering of incomplete data: one for the clustering of bounded-domain (i.e., integer) data, and two incomparable algorithms that target real-valued data. Our approach is based on exploiting structural properties of a graphical encoding of the missing entries, and we show that tractability can be achieved using significantly less restrictive parameterizations than in the complementary case of few missing entries.

1. Introduction
k-means clustering is a fundamental task in machine learning and data analysis, one that has been the focus of extensive empirical as well as theoretical research. In general, the aim in k-means clustering is to partition the rows in an input matrix A into k clusters and compute one center per cluster so as to minimize the within-cluster sum of squares—or, when viewed as a decision problem, achieve a within-cluster sum of squares of at most a specified target value ℓ. Depending on the setting, A could be real-valued (Lloyd, 1982; Moshkovitz et al., 2020; Fomin et al., 2021) or contain integers from a bounded domain; for instance, Fomin et al. (Fomin et al., 2018) studied the case where A was binary, and Ganian et al. (Ganian et al., 2018) the bounded-domain case where ℓ = 0.

It is well known that k-means clustering is NP-complete, even when restricted to k = 2 clusters (Drineas et al., 2004; Aloise et al., 2009). On the theoretical side, a prominent approach used to circumvent such lower bounds is the use of the parameterized complexity paradigm, which provides a more fine-grained look into the complexity of problems. Since its inception, parameterized complexity has been successfully applied to a multitude of problems relevant to machine learning and artificial intelligence such as, e.g., Bayesian network learning (Ordyniak & Szeider, 2013; Grüttemeier et al., 2021; Ganian & Korchemna, 2021), integer linear programming (Ganian & Ordyniak, 2018; 2019; Dvoráček et al., 2021), principal component analysis (Simonov et al., 2019; Dahiya et al., 2021), and of course clustering (Cohen-Addad et al., 2018; Fomin et al., 2018).

Apart from classical clustering problems, one direction that has been gaining traction in recent years is the study of clustering in settings where some of the data is not known or simply irrelevant—a task that essentially combines data completion with clustering (Wang et al., 2019). Indeed, several authors have studied the parameterized complexity of various clustering problems for data with missing entries (Ganian et al., 2018; Koana et al., 2020; 2021; Eiben et al., 2021b), and Eiben et al. (Eiben et al., 2021a) obtained a fixed-parameter approximation algorithm specifically for (k-)MEANS CLUSTERING WITH MISSING ENTRIES (MCME).

While their approaches and techniques differ, all of these works target the case where nearly all entries are known, i.e., where the unknown entries only occur in a precisely defined “sparse” way. On the other hand, very little is known about the complexity of MCME when the number of unknown entries is large; such situations are of practical relevance in, e.g., recommender systems and predictive analytics. For example, in the classical Netflix Prize challenge where the task was to predict user ratings for movies based on previous ratings, only about 1% of the user-movie pairs were orig-

* Equal contribution 1 Algorithms and Complexity Group, TU Wien, Austria 2 Institute of Informatics, University of Warsaw, Poland. Correspondence to: Robert Ganian <rganian@gmail.com>, Thekla Hamm <thamm@ac.tuwien.ac.at>, Viktoriia Korchemna <vkorchemna@ac.tuwien.ac.at>, Kirill Simonov <ksimonov@ac.tuwien.ac.at>, Karolina Okrasa <k.okrasa@mini.pw.edu.pl>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

2 See Preliminaries for a brief introduction.
The Complexity of k-Means Clustering when Little is Known

Originally supplied with a rating. The central mission of this article is to push the boundaries of tractability for MCME to instances where known entries are sparse.

**Contribution.** A natural approach for handling instances of MCME with many missing entries would be to invert the parameterizations that have been developed for tackling instances where almost all entries are known. For instance, Eiben et al. (Eiben et al., 2021b) and Ganian et al. (Ganian et al., 2018) obtained several fixed-parameter clustering algorithms by using a parameter called the covering number, which is the minimum number of rows and columns needed to cover all the unknown entries. Equivalently, if we use an auxiliary binary matrix \( W \) (the mask) to specify which entries of \( A \) are relevant/known, the covering number is simply the minimum number of rows and columns needed to cover all the \( 0 \)'s in \( W \). It would be tempting to instead parameterize by the number of rows and columns needed to cover all the \( 1 \)'s in \( W \); such an “inverse covering number” would lead to fixed-parameter algorithms targeting instances where all known entries occur in a few columns and rows.

However, we show that in our setting it is in fact possible to do much better than that. Our approach is based on using the incidence graph representation of \( W \), which is the graph containing one vertex for each column and for each row of \( W \) and where an edge connects row \( a \) with column \( b \) if and only if \( W[a, b] = 1 \). Observe that the inverse covering number considered in the previous paragraph would simply be the size of a minimum vertex cover in \( W \), while the covering number is the size of a minimum vertex cover in the complement of \( W \). It is worth noting that in graph-theoretic settings, the size of a minimum vertex cover is considered a highly restrictive parameterization, one that is used primarily when more desirable parameterizations fail. On the other hand, the by far most prominent parameter used for graphs is treewidth (Robertson & Seymour, 1984), which has found ubiquitous applications throughout computer science and which is smaller (i.e., less restrictive as a parameter) than the size of a minimum vertex cover. As our first result, we use the treewidth of the incidence graph of the mask \( W \) (the incidence treewidth) to prove:

**Result 1:** Bounded-domain MCME is fixed-parameter tractable when parameterized by the incidence treewidth of the mask.

Result 1 is noteworthy not only due to the use of treewidth instead of the inverse covering number, but also because incidence treewidth is the only parameter required to achieve tractability. In particular, unlike previous algorithms for clustering incomplete (bounded-domain or real-valued) data by Eiben et al. (Eiben et al., 2021a;b), Ganian et al. (Ganian et al., 2018), and Koana et al. (Koana et al., 2020; 2021), Result 1 can also be applied to instances where the number \( k \) of clusters is large. As we will see later, we will be able to retain this benefit in all our algorithms.

Unfortunately, an extension of Result 1 towards real-valued instances seems difficult at this point. Indeed, a fixed-parameter algorithm for real-valued MCME parameterized by the incidence treewidth of the mask would imply, as a special case, fixed-parameter tractability parameterized by the number \( d \) of columns. However, the existence of a fixed-parameter algorithm even for \(-M\)EANS (corresponding to the restriction of MCME where all entries are known) when parameterized by \( d \) is a long-standing open problem dating back to Inaba et al.’s celebrated XP algorithm parameterized by \( k + d \) (Inaba et al., 1994).

Instead, we show that one can obtain fixed-parameter algorithms for real-valued MCME by using the treewidth of a different graph representation of \( W \). In particular, we use the primal graph—a well-established counterpart to the incidence graph representation in settings such as Boolean satisfiability (Samer & Szeider, 2009), constraint satisfaction (Samer & Szeider, 2010) or integer programming (Ganian & Ordyniak, 2019). In our context, the primal graph of \( W \) contains a vertex for each row of \( W \), with edges connecting pairs of rows that share at least one coordinate which is known/relevant according to the mask, and the primal treewidth is simply the treewidth of this graph.

**Result 2:** Real-valued MCME is fixed-parameter tractable when parameterized by the primal treewidth of the mask.

Note that, due to the nature of the primal graph representation, the primal treewidth is a more restrictive parameter than the incidence treewidth (in the sense of being bounded on fewer classes of instances). In fact, the primal treewidth of a mask may in certain cases be unboundedly large even when its incidence graph is just a tree. It would hence be useful to also have an algorithm for real-valued MCME that can exploit properties of the incidence graph, even for properties that are more restrictive than incidence treewidth.

However, the same obstacle towards extending Result 1 to the real-valued setting also applies to all traditional structural parameters that are, intuitively, based on small vertex separators in the graph—including pathwidth (Robertson & Seymour, 1983), treedepth (Nesetril & de Mendez, 2012), the feedback vertex number e.g., (Mertzios et al., 2020) and even the vertex cover number e.g., (Gaspers & Najeebullah, 2019; Fellows et al., 2018), which is the most restrictive vertex separator based parameter. In spite of this, as our third and final result we identify a parameter of the incidence graph that yields fixed-parameter tractability of real-valued MCME: the recently introduced local feedback edge number (Ifen) (Ganian & Korchemnaya, 2021). On a high level,
We note that the parameters used in Results 2 and 3 are when parameterized by the local feedback edge number of the incidence graph of the mask.

We note that the parameters used in Results 2 and 3 are orthogonal and the gap between them may be arbitrarily large. Because of this, both results are incomparable and give rise to different tractable classes for the problem.

2. Preliminaries

For an integer \( i \), we let \([i] = \{1, 2, \ldots, i\}\) and \([i]_0 = [i] \cup \{0\}\). On the other hand, if \( p \) is an equivalence relation on domain \( X \), we denote by \([p]\) the set of equivalence classes of \( p \) and by \( |p|\) the cardinality of \( p \), i.e., the number of equivalence classes of \( p \). For every \( x \in X \), \([x]_p \in [p] \) is the equivalence class containing \( x \).

### Notation and Problem Formulation.

Let \( A \) be a matrix with \( n \) rows and \( d \) columns. We use \( V_A \) and \( C_A \) to denote the set of row and column indices of \( A \), i.e., \( V_A = [n] \) and \( C_A = [d] \). Let \( M_A = \max_{v \in V_A} \max_{c \in C_A} |A[v, c]| \) for two matrices \( A, B \). We denote by \( A - B \) the entry-wise subtraction of the two matrices, i.e., \( (A - B)[i, j] = A[i, j] - B[i, j] \), and similarly by \( A \circ B \) the entry-wise product of the two matrices, i.e., \( (A \circ B)[i, j] = A[i, j] \cdot B[i, j] \). For a matrix \( A \in \mathbb{R}^{n \times d} \) we denote its squared Frobenius norm by \( |A|_F^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} A[i, j]^2 \). We can now formalize our problems of interest (abbreviated as MCME):

**MEANS CLUSTERING WITH MISSING ENTRIES**

**Input:** Matrix \( A \in D^{n \times d} \) over a domain \( D \subseteq \mathbb{R} \), binary matrix \( W \) (the mask), integers \( k \) and \( \ell \).

**Task:** Determine if there exists a matrix \( B \) over \( D \) containing at most \( k \) distinct rows such that \( \|W \circ (A - B)\|_F^2 \leq \ell \).

We distinguish between BOUNDED-DOMAIN MCME and REAL-VALUED MCME depending on whether \( D = \mathbb{Z}_0 \) for some fixed integer \( z \), or \( D \) is the set of reals. In particular, for \( z = 1 \) this coincides with BINARY \( k \)-MEANS CLUSTERING (Fomin et al., 2018). In this case, minimizing the Frobenius norm also minimizes the sum of Hamming distances for each row to its cluster center. Using the Frobenius norm also for \( z > 1 \) is not only consistent with the continuous case, but also allows us to capture further applications, e.g., when the data matrix to cluster is composed of user ratings each taking a small numeric value. While we state MCME as a decision problem for complexity-theoretic reasons, every algorithm presented here is constructive and can output a solution matrix \( B \) if one exists.

In this formulation, the distinct rows of \( B \) are the centers of the sought-after clusters while \( \ell \) is the target upper-bound on the sum of squares. For \( V' \subseteq V_A \), we call a mapping \( \varphi : V' \to [k] \) a cluster-assignment w.r.t. \( B \) if for every \( v_1, v_2 \in V' \) such that \( \varphi(v_1) = \varphi(v_2) \), it holds that \( B[v_1] = B[v_2] \). Notice that the existence of a cluster assignment w.r.t. a matrix \( B' \) implies that \( B' \) has at most \( k \) distinct rows. The clusters associated with \( \varphi \) are then sets of the form \( \{v \in V \mid \varphi(v) = i\} \) for some \( i \in [k] \).

In our algorithms, we often construct the solution matrices dynamically by “merging” smaller matrices; we formalize this procedure below. Let \( \{B_i \mid i \in [m]\} \) be a set of matrices, where each \( B_i \) has row labels \( V_i \), column labels \( C_i \), and is associated with a cluster-assignment \( \varphi_i : V_i \to [k] \) and the following two consistency conditions hold:

- \( \varphi_i(v) = \varphi_j(v) \) for every \( v \in V_i \cap V_j \), \( i, j \in [m] \,
- if \( \varphi_i(v_i) = \varphi_j(v_j) \) for some \( v_i \in V_i \) and \( v_j \in V_j \), then \( B_i[v_i, c] = B_j[v_j, c] \) for every \( c \in C_i \cap C_j \).

Let \( \varphi = \bigcup_{i \in [m]} \varphi_i \). We define the composition of \( (B_i, \varphi_i), i \in [m] \), to be the matrix \( B^* \) with row labels \( V = \bigcup_{i \in [m]} V_i \), column labels \( C = \bigcup_{i \in [m]} C_i \) and entries as follows. For every \( v \in V \) and \( c \in C \), pick any \( i \in [m] \) such that \( c \in C_i \) and there exists \( v_i \in V_i \) with \( \varphi_i(v_i) = \varphi(v) \). We set \( B^*[v, c] = B_i[v_i, c_i] \). If there is no such \( i \in [m] \), we simply set \( B^*[v, c] \) to some arbitrary but uniform default value—in our case, we will always use 0. Observe that if both consistency conditions hold, then both \( B^* \) and \( \varphi \) are well-defined and that \( \varphi \) is a cluster-assignment w.r.t. \( B^* \).

**Claim 1.** \( B^* \) is well-defined and admits the cluster-assignment \( \varphi \).

**Proof.** To see that \( B^* \) is well-defined, pick \( v \in V \) and \( c \in C_i \cap C_j \). If there exist \( v_i \in V_i \) and \( v_j \in V_j \) such that \( \varphi_i(v_i) = \varphi(v) \) and \( \varphi_j(v_j) = \varphi(v) \), then \( \varphi_i(v_i) = \varphi_j(v_j) \), so \( B_i[v_i, c] = B_j[v_j, c] \) by the consistency conditions. Moreover, \( \varphi \) is the cluster-assignment w.r.t. \( B^* \). Indeed, consider \( u, v \in V \) with \( \varphi(u) = \varphi(v) \) and \( c \in C \). If \( c \in C_i \) for some \( i \in [m] \) such that there exists \( v_i \in V_i \) with \( \varphi(v_i) = \varphi(u) \), then \( B^*[v, c] = B_i[v_i, c] = B^*[u, c] \). If there is no such \( i \in [m] \), we have \( B^*[v, c] = 0 = B^*[u, c] \).

Observe that if \( C_i \cap C_j = \emptyset \) for every \( i \neq j \), we can skip the second consistency condition: in this case it just requires \( B_i[v] = B_i[u] \) whenever \( \varphi_i(v) = \varphi_i(u) \), which holds as \( \varphi_i \) is the cluster-assignment w.r.t. \( B_i \).

**Parameterized Complexity.** In parameterized algorithms (Downey & Fellows, 2013; Cygan et al., 2015), the running-time of an algorithm is studied with respect to a parameter \( k \in \mathbb{N}_0 \) and input size \( n \). The basic idea is to find a parameter that describes the structure of the instance.
such that the combinatorial explosion can be confined to this parameter. In this respect, the most favorable complexity class is FPT (fixed-parameter tractable), which contains all problems that can be solved in time $f(k) \cdot n^{O(1)}$, where $f$ is a computable function. Algorithms with this running-time are called fixed-parameter algorithms.

Sparse Matrices. It follows from definition of MCME that entries of the input matrix $A$ and the target matrix $B$ are only relevant when the respective entry of the mask $W$ is set to “1”. We thus assume that only those entries are stored explicitly; in particular, the size of the input is therefore linear in the number of non-zero entries of $W$. All the structural parameters in our considerations automatically imply that the mask $W$ is sparse, in the sense that it has at most $O_k(n + d)$ non-zero entries, where the constant under $O_k(\cdot)$ depends on the respective parameter $k$. All algorithms in this paper except one are linear-time in the number of non-zero entries of the mask, and the above allows us to state their running times in the form $(n + d) \cdot f(k)$, where $f$ is a certain function of the respective parameter $k$.

Treewidth. A nice tree-decomposition $T$ of a graph $G = (V, E)$ is a pair $(T, \chi)$, where $T$ is a tree (whose vertices we call nodes) rooted at a node $r$ and $\chi$ is a function that assigns each node $t$ a set $\chi(t) \subseteq V$ such that:

- For every edge $uv \in E$ there is a node $t$ where $u, v \in \chi(t)$.
- For every vertex $v \in V$, the set of nodes $t$ satisfying $v \in \chi(t)$ forms a subtree of $T$.
- $|\chi(t)| = 1$ for every leaf $t$ of $T$ and $|\chi(r)| = 0$.
- There are only three kinds of non-leaf nodes in $T$:
  - Introduce node: a node $t$ with exactly one child $t'$ such that $\chi(t) = \chi(t') \cup \{v\}$ for some vertex $v \not\in \chi(t')$.
  - Forget node: a node $t$ with exactly one child $t'$ such that $\chi(t) = \chi(t') \setminus \{v\}$ for some vertex $v \in \chi(t')$.
  - Join node: a node $t$ with two children $t_1, t_2$ such that $\chi(t) = \chi(t_1) = \chi(t_2)$.

The width of a nice tree-decomposition $(T, \chi)$ is $\max_{t \in V(T)}(|\chi(t)| - 1)$, and the treewidth of the graph $G$, denoted $\tw(G)$, is the minimum width of a nice tree-decomposition of $G$. We let $T_r$ denote the subtree of $T$ rooted at a node $t$, and we use $\chi^+_t$ to denote the set $\bigcup_{t' \in V(T_r)} \chi(t')$. The sets $\chi(t)$ are commonly called bags.

Fixed-parameter algorithms are known for computing nice tree-decompositions of optimal width and linearly many nodes (Bodlaender, 1996; Kloks, 1994), albeit more efficient fixed-parameter approximation algorithms are often used to achieve better running times (Bodlaender et al., 2016).

Figure 1. Example of a graph $G$ with the spanning tree $T$ (marked in thick red) such that $\lfen(G) = \lfen(G, T) = 2$. The feedback edge number of $G$, i.e., its edge deletion distance to acyclicity, is exactly the number of black edges and can be made arbitrarily large in this fashion while preserving $\lfen(G) = 2$.

Local Feedback Edge Number. The local feedback edge number was recently introduced by Ganian and Korčemna (2021) as an edge-cut based restriction of treewidth that acts as a “local” measure of the size of a feedback edge set; they used the parameter to obtain fixed-parameter algorithms for learning Bayesian networks and polytrees.

For a connected graph $G = (V, E)$ and a spanning tree $T$ of $G$, the local feedback edge set at $x \in V$ is defined as $E_{loc}(x) = \{yz \in E \setminus E(T) \mid \text{the unique path between } y \text{ and } z \text{ in } T \text{ contains } x\}$. The local feedback edge number of $(G, T)$ (denoted $\lfen(G, T)$) is then equal to $\max_{x \in V} |E_{loc}(x)|$, and the local feedback edge number of $G$ is simply the smallest local feedback edge number among all possible spanning trees of $G$, i.e., $\lfen(G) = \min_T \lfen(G, T)$ is a spanning tree of $G$. An example of a graph $G$ with $\lfen(G) = 2$ is provided in Figure 1.

3. Solving Bounded-Domain MCME by Exploiting the Mask

Since BOUNDED-DOMAIN MCME is NP-complete (Drineas et al., 2004; Aloise et al., 2009), it is natural to attempt and circumvent this lower bound by exploiting structural measures of the inputs to obtain fixed-parameter tractability. Eiben et al. (Eiben et al., 2021a) recently obtained a fixed-parameter approximation algorithm for BOUNDED-DOMAIN requiring the simultaneous parameterization by three measures: the number of clusters, the desired approximation ratio, and a technical measure that captures the sparsity of the missing data (i.e., the “0” entries in $W$).

In this section, we will present a fixed-parameter algorithm for BOUNDED-DOMAIN MCME which is aimed at the supplementary case where the relevant data is sparse. To formalize this, we consider the incidence graph representation $G_I$ of $W$ which is defined as follows: $V(G_I) = V_W \cup C_W$, and $E(G_I) = \{ab \mid a \in V_W, b \in C_W, W[a, b] = 1\}$. Figure 2 later on can also be used as an example of the representation.

Intuitively, each “1” entry in $W$ will correspond to an edge in $G_I$, and hence “structurally sparse” incidence graphs correspond to settings where most data is unknown. There is a well-studied hierarchy of structural graph parameters
which measure, in a certain sense, the sparsity of graphs (see, e.g., Figure 1 in (Bodlaender et al., 2013)). Treewidth will be our parameter of choice here, as the best known and also most general parameter in this hierarchy. However, it is worth noting that the use of the incidence graph may provide a new perspective on previous algorithms for clustering problems—for instance, the so-called covering number parameter (Ganian et al., 2018; Eiben et al., 2021b) is simply the vertex cover number of the complement of $G_I$.

The main goal of this section is to establish the fixed-parameter tractability of MCME parameterized by $tw(G_I)$. As a first step towards this goal, we obtain a dynamic programming algorithm that handles the simpler case where $k$ is also part of the parameterization.

**Theorem 1.** **Bounded-Domain MCME is fixed-parameter tractable when parameterized by** $k + tw(G_I)$.

**Proof.** We begin by applying the well-established 5-approximation algorithm for treewidth (Bodlaender et al., 2016) to compute a nice tree-decomposition of $G_I$ of width $q \leq 5 tw(G_I)$ in time $2^{O(tw(G_I))}(n + d)$. Let $T$ be the root of $T$. Given a node $t$ of $T$, let $V_t$ and $C_t$ be the sets of vectors and coordinates in $\chi(t)$, respectively. Moreover, let $C_t^+$ and $V_t^+$ be the restrictions of $\chi_t$ to sets of coordinates and vectors correspondingly.

To prove the theorem, we will design a leaf-to-root dynamic programming algorithm which will compute and store a set of records at each node $t$ of $T$, whereas once we ascertain the records for $r$ we will have the information required to output a correct answer. Intuitively, the records will store the cluster centers restricted to the coordinates that appear in the bag, the partition of the vectors in the bag into clusters and the sum of minimum distances from vectors in $V_t^+$ to the cluster centers along the coordinates in $C_t^+$.

Formally, the records will have the following structure. We call a pair $(\text{cent}, \text{part})$ a snapshot in $t$ if there holds:

- $\text{cent} : [k] \times C_t \to D$,
- $\text{part} : V_t \to [k]$.

Let $S(t)$ be the set of all snapshots of $t$. The record $R_t$ of $t$ is then a mapping from $S(t)$ to the set $\mathbb{R}^+$ of non-negative reals. Observe that $|S(t)| \leq |D|^{k(q+1)}k^{q+1}$.

To introduce the semantics of our records, let $B_t$ be the set of all matrices with row labels $V_t^+$, column labels $C_t^+$ and entries of domain $D$. Let $B_t$ be a matrix in $B_t$. We define the *partial weighted distance* from $B_t$ to $A$ in $t$ as follows:

$$\text{pwd}(B_t, t) = \sum_{v \in V_t^+} \sum_{c \in C_t^+} W[v, c] \cdot (A[v, c] - B_t[v, c])^2.$$ 

For $B_t \in B_t$, we say that $(\text{cent}, \text{part}) \in S(t)$ is the snapshot of $B_t$ in $t$ if there is a cluster-assignment $\varphi$ w.r.t. $B_t$ such that the following conditions hold:

- part = $\varphi|_V$,
- for every $c \in C_t$ and $v \in V_t^+$, $B_t[v, c] = \text{cent}[\varphi(v), c]$.

Recall that the existence of a cluster-assignment implies, in particular, that $B_t$ has at most $k$ distinct rows. We are now ready to define the record $R_t$. For each snapshot $(\text{cent}, \text{part}) \in S(t)$, we set $R_t(\text{cent}, \text{part}) = \tau$ if there exists $B_t \in B_t$ such that:

- $(\text{cent}, \text{part})$ is the snapshot of $B_t$ in $t$,
- $\text{pwd}(B_t, t) = \tau$, and
- $\forall B_t' \in B_t$ such that $(\text{cent}, \text{part})$ is the snapshot of $B_t'$, $\text{pwd}(B_t', t) \geq \tau$.

In this case we say that $B_t$ witnesses $R_t(\text{cent}, \text{part}) = \tau$.

Recall that for the root $r$, we assume $\chi(r) = \emptyset$. Hence $S(r)$ contains only one element $(\emptyset, \emptyset)$, and $R_r(\emptyset, \emptyset) = 0$ is equal to the minimum value of $\|W \circ (A - B)\|_F^2$ that can be achieved by any matrix $B$ with entries of domain $D$ containing at most $k$ distinct rows. In other words, the instance is a YES-instance if and only if $R_r(\emptyset, \emptyset) \leq \ell$. To prove the theorem, it now suffices to show that the records can be computed in a leaf-to-root fashion by proceeding along the nodes of $T$.

We distinguish the following cases:

- $t$ is a leaf node. Let $\chi(t) = \{v\}$ where $v$ is a vector. By definition, $S(t) = \{(\emptyset, \emptyset)\} \text{ part} : [1] \to [k]\} \text{ and } R_t(\emptyset, \emptyset) = 0$ for every $(\emptyset, \emptyset) \in S(t)$ as there are no coordinates to sum over. Let $\chi(t) = \{c\}$ for some coordinate $c$. Then $S(t) = \{(\text{cent}, \emptyset)\} \text{ cent} : [k] \times [1] \to D\} \text{ and } R_t(\text{cent}, \emptyset) = 0$ for each $(\text{cent}, \emptyset) \in S(t)$.

- $t$ is a forget node. Let $t'$ be the child of $t$ in $T$ and $\chi(t) = \chi(t') \setminus \{c\}$ for some vector $v$. We set $R_t(\text{cent}, \text{part}) := \min_{B_t \in B_t} \chi(t') \setminus \{c\} \text{ R}_t(\text{cent}, \text{part} \cup (v, i))$ for each $(\text{cent}, \text{part}) \in S(t)$. For correctness, it will be useful to observe that $B_t$ is $B_{t'}$. If $R_{t'}(\text{cent}, \text{part}) = \tau$, then there exists a matrix $B_t$ which witnesses this. But then $B_t$ also admits a snapshot $(\text{cent}, \text{part} \cup \{(v, i)\})$ at $t'$ for some $i \in [k]$ and witnesses $R_{t'}(\text{cent}, \text{part} \cup (v, i)) \leq \tau$. So in our algorithm $R_{t'}(\text{cent}, \text{part}) \leq R_{t'}(\text{cent}, \text{part} \cup (v, i)) \leq \tau$. If on the other hand $R_{t'}(\text{cent}, \text{part}) = \tau$, then there exists a snapshot $(\text{cent}, \text{part} \cup \{(v, i)\})$ at $t'$ for some $i \in [k]$ such that $R_{t'}(\text{cent}, \text{part} \cup \{(v, i)\}) = \tau$. $R_t(\text{cent}, \text{part}) \leq \tau$ now follows from the existence of a matrix witnessing the value of $R_{t'}(\text{cent}, \text{part} \cup \{(v, i)\})$. Hence, we can correctly set $R_t = R_{t'}$.

If $\chi(t) = \chi(t') \setminus \{c\}$ for some coordinate $c$, we set $R_t(\text{cent}, \text{part})$ equal to $\min \{R_{t'}(\text{cent}, \text{part})\}$ cent is a restriction of $\text{cent}$ to all coordinates except $c$. Correctness can be argued similarly to the case of a forgotten vector.

- $t$ is an introduce node (introducing a vector). Let $t'$ be
the child of \( t \) in \( T \) and let \( \chi(t) = \chi(t') \cup \{v_0\} \) for some vector \( v_0 \). Fix a snapshot \((\text{cent}, \text{part})\) in \( S(t) \), let \( i = \text{part}(v_0) \) and \( t' = \text{part}(v_0, t) \). We will denote by \( \Delta_0 \) the sum of distances from \( v_0 \) to the \( i \)-th cluster center along the coordinates in \( C_i \), i.e., \( \Delta_0 = \sum_{c_i \in C_i} W[v_0, c_i] \cdot (A[v_0, c_i] - \text{cent}[i, c_i])^2 \). We set \( \mathcal{R}_0(t, \text{cent}, \text{part}) = \mathcal{R}_0(t', \text{cent}, \text{part}) + \Delta_0 \). For correctness, assume that \( \mathcal{R}_0(t', \text{cent}, \text{part}) = \tau = \mathcal{R}_0(t', \text{cent}, \text{part}) + \Delta_0 \). Construct a matrix \( B_t \) from the witness \( B_{t'} \) of \( \mathcal{R}_0(t', \text{cent}, \text{part}) \) with a cluster-assignment \( \varphi' \) by adding a new row with a label \( v_0 \) as follows. If \( i = \varphi'(v) \) for some \( v \in V_{t'}^i \), we set \( B_t[v_0] = B_t[v, c] \). Otherwise we define \( B_t[v_0, c] = \text{cent}[i, c] \) for \( c \in C_i \) and \( B_t[v_0, c] := 0 \) in the rest of coordinates \( c \). Note that in both cases \( B_t[v_0, c] = \text{cent}[i, c] \) for every \( c \in C_i \). In the second case it follows from the definition, while in the first one we have \( B_t[v_0, c] := B_t[v, c] = \text{cent}[\varphi'(v), c] \) equals \( \text{cent}[i, c] \). The matrix \( B_t \) with this cluster-assignment \( \varphi = \varphi' \cup \{v_0\} \) has a snapshot \((\text{cent}, \text{part})\) in \( t \) and \( \text{pwd}(B_t, t) = \sum_{v \in V_t^i} \sum_{c_i \in C_i} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 = \text{pwd}(B_t, t') + \sum_{c_i \in C_i} W[v_0, c_i] \cdot (A[v_0, c_i] - B_t[v_0, c_i])^2 \). As \( \text{pwd}(B_t, t') = \mathcal{R}_0(t', \text{cent}, \text{part}) \), \( W[v_0, c] = 0 \) for every forgotten coordinate \( v \) and \( B_t[v_0, c] := \text{cent}[i, c] \) for every \( c \in C_i \), we have \( \text{pwd}(B_t, t) = \mathcal{R}_0(t, \text{cent}, \text{part}) + \Delta_0 = \tau \). So \( B_t \) witnesses that \( \mathcal{R}_t(\text{cent}, \text{part}) \leq \tau \).

On the other hand, assume that \( \mathcal{R}_t(\text{cent}, \text{part}) = \tau \). Then there exists a matrix \( B_t \) in \( B_t \), admitting the snapshot \((\text{cent}, \text{part})\) in \( t \) such that \( \text{pwd}(B_t, t) = \tau \). Let \( \varphi \) be the corresponding cluster-assignment w.r.t. \( B_t \). We construct \( B_{t'} \) from \( B_t \) by deletion of the row \( v_0 \). Then \( B_{t'} \) with cluster-assignment \( \varphi' = \varphi \setminus \{v_0\} \) has a snapshot \((\text{cent}, \text{part})\) in \( t \) and witnesses that \( \mathcal{R}_0(\text{cent}, \text{part}) \leq \text{pwd}(B_{t'}, t') = \tau - \Delta_0 \). Therefore in our algorithm \( \mathcal{R}_0(t', \text{cent}, \text{part}) + \Delta_0 \leq \tau \). Hence, we can correctly set \( \mathcal{R}_t = \mathcal{R}_0 \).

**t is a join node.** Let \( t_1, t_2 \) be the two children of \( t \) in \( T \), recall that \( \chi(t_1) = \chi(t_2) = \chi(t) \) and \( \chi^i_1 \cap \chi^i_2 = \chi(t) \). For every \( (\text{cent}, \text{part}) \) in \( S(t) \) we set \( \mathcal{R}_0(t', \text{cent}, \text{part}) = \mathcal{R}_t(t_1, \text{cent}, \text{part}) + \mathcal{R}_t(t_2, \text{cent}, \text{part}) - \text{doublecount} \), where doublecount = \( \sum_{v \in V_t} \sum_{c_i \in C_i} W[v, c] \cdot (A[v, c] - \text{cent}[\text{part}(v), c])^2 \).

For correctness, assume that \( \mathcal{R}_0(t', \text{cent}, \text{part}) = \tau = \tau_1 + \tau_2 - \text{doublecount} \), where \( \tau_1 = \mathcal{R}_t(t_1, \text{cent}, \text{part}) \), \( \tau_2 = \mathcal{R}_t(t_2, \text{cent}, \text{part}) \). Let \( B \) be the matrix with cluster-assignment \( \varphi = \varphi_1 \cup \varphi_2 \) as a composition of \((B_1, \varphi_1)\) and \((B_2, \varphi_2)\). To check the consistency conditions, observe that the sets of common row and column labels of \( B_1 \) and \( B_2 \) are \( V_t \) and \( C_t \) correspondingly. Recall that \( \varphi_1|_{V_t} = \text{part} = \varphi_2|_{V_t} \). Moreover, if \( v_1 \in V_t \) and \( v_2 \in V_t \) are such that \( \varphi_1(v_1) = \varphi_2(v_2) \), then we have \( B_1[v_1, c] = \text{cent}[\varphi_1(v_1), c] = \text{cent}[\varphi_2(v_2), c] = B_2[v_2, c] \) for every \( c \in C_t \).

Note that for every \( v \in V_t \), \( \varphi(v) = \varphi_1(v) \) for. Pick \( c \in C_t \) and \( v \in V_t^i \), then \( v \in V_t^i \) for some \( i \in \{1, 2\} \) and so \( B_t[v, c] = B_t[v, c] = \text{cent}[\varphi_1(v), c] = \text{cent}[\varphi_1(v), c] \). Therefore \((\text{cent}, \text{part})\) is a snapshot of \( B_t \) in \( t \). Recall that \( \text{pwd}(B_t, t) = \sum_{v \in V_t^i} \sum_{c_i \in C_t} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 \). Here \( W[v, c] = 0 \) for every \( v \in V_t^i \) and \( c \in C_t \), \( i = 1, 2 \). So \( \text{pwd}(B_t, t) = \sum_{v \in V_t^i} \sum_{c_i \in C_t} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 = \text{pwd}(B_1, t_1) + \text{pwd}(B_2, t_2) - \text{doublecount} = \tau_1 + \tau_2 - \text{doublecount} = \tau \). Hence \( B_t \) witnesses \( \mathcal{R}_t(\text{cent}, \text{part}) \) for every \( \text{pwd}(B_t, t) = \mathcal{R}_t(\text{cent}, \text{part}) \).

For the converse, assume that \( \mathcal{R}_t(\text{cent}, \text{part}) = \tau \) and \( B_t \) is a matrix witnessing this. Let \( B_t \) be the restriction of \( B_t \) to rows \( V_t^i \) and columns \( C_t^i \), \( i = 1, 2 \). Then \( B_1 \) and \( B_2 \) have a snapshot \((\text{cent}, \text{part})\) and in our algorithm \( \mathcal{R}_t(t', \text{cent}, \text{part}) = \mathcal{R}_t(t, \text{cent}, \text{part}) + \mathcal{R}_t(t, \text{cent}, \text{part}) - \text{doublecount} \leq \text{pwd}(B_1) + \text{pwd}(B_2) - \text{doublecount} = \text{pwd}(B) = \tau \) for every \( \text{pwd}(B) = \mathcal{R}_t(\text{cent}, \text{part}) \). Hence the resulting record \( \mathcal{R}_t \) is correct, which concludes the correctness proof of the algorithm.

To bound the runtime of the algorithm, observe that at each node \( t \) we compute the record \( \mathcal{R}_t \) for \( S(t) \) \( \leq |D|^{|k|} \cdot |D|^{|O(k \cdot \text{tw}(G_t))|} \) entries, where each entry is computed in time at most quadratic in \( \text{tw}(G_t) \). Since the tree-decomposition is nice and has at most \( \mathcal{O}(n + d) \)
nodes, the runtime of the algorithm is upper-bounded by \((n + d) \cdot |D|^O(k \cdot \text{tw}(G_1))\).

Towards proving fixed-parameter tractability without involving \(k\), we consider the case where \(k\) is significantly larger than \(\text{tw}(G_1)\). We prove that this case always admits a “perfect” solution, which can also be computed efficiently.

**Lemma 2.** If \(k \geq (\text{tw}(G_1) + 1) \cdot |D|^{\text{tw}(G_1) + 1}\), then there exists a matrix \(B\) over domain \(|D|\) containing at most \(k\) distinct rows such that \(\|W \circ (A - B)\|_2^2 = 0\). Moreover, given a nice tree-decomposition of width \(q - 1\) such that \(k \geq q \cdot |D|^q\), such a matrix \(B\) can be computed in time \((n + d) \cdot |D|^{O(q)}\).

**Proof.** We will prove the second claim of the lemma, from which the first one follows. To this end, let us assume that we are given a nice tree-decomposition \(T = (T, \chi)\) of \(G_1\) of width \(q - 1\) and \(k \geq q \cdot |D|^q\). We denote the root of \(T\) by \(r\) and define \(C_t, V_t, C_t^i\) and \(V_t^i\) analogously to the proof of Theorem 1. We will explicitly construct \(q \cdot |D|^q\) cluster centers yielding a zero sum of squares, so that after processing the node \(t\), each of the centers is defined on coordinates \(c \in C_t^i\). The remaining clusters will be empty, thus for the rest of the proof we assume \(k = q \cdot |D|^q\).

Formally, we will attach to every node \(t\) the pair \((R_t, \alpha_t)\) where \(R_t\) is the matrix with the row labels \(k\) and the column labels \(C_t^i\), and \(\alpha_t\) is a mapping from \(V_t^i\) to \(k\). Intuitively, \(R_t\) stores the restrictions of cluster centers to processed coordinates, while \(\alpha_t\) assigns a cluster index to every row label \(v \in V_t^i\). Throughout the dynamic programming procedure, we will maintain the following invariants:

1. \(\alpha_t(u) \neq \alpha_t(v)\) for any distinct \(u\) and \(v\) from \(V_t^i\).
2. for every \(\omega \in D^{C_t^i}\), there are precisely \(\frac{k}{|D|^{C_t^i} - 1}\) indices \(i \in [k]\) such that \(R_t[i][C_t] = \omega\), where \(R_t[i][C_t]\) is the restriction of \(R_t[i]\) to the coordinate set \(C_t^i\).
3. \(\sum_{v \in V_t^i} \sum_{c \in C_t^i} W[v, c] \cdot (A[v, c] - R_t[\alpha_t(v), c])^2 = 0\).

Assume that the invariants hold in the root \(r\), let us define the matrix \(B\) by setting \(B[v] = R_r[\alpha_r(v)]\). Then \(B\) has at most \(k\) distinct rows and the third invariant yields that \(\sum_{v \in V_A} \sum_{c \in C_A} W[v, c] \cdot (A[v, c] - B[v, c])^2 = 0\). In particular, for \(q = \text{tw}(G_1) + 1\) we get the statement of the lemma. Hence it is sufficient to construct \((R_t, \alpha_t)\) in a leaf-to-root fashion, for each of the following cases.

**t is a leaf node.** If \(\chi(t) = \{v_0\}\) for some row label \(v_0\), we set \(\alpha_t(v_0) = 1\) and \(R_t[i] = 0\) for every \(i \in [k]\). Otherwise, \(\chi(t) = \{c\}\) for a coordinate \(c\). Then \(\alpha_t\) should be the empty mapping, for every \(i \in [k]\) we set \(R_t[i, c]\) equal to some \(\gamma \in D\) so that every \(\gamma \in D\) appears in \(R_t\) precisely \(\frac{k}{|D|}\) times. It is easy to see that all the invariants are satisfied.

**t is a forget node.** Let \(t'\) be the child of \(t\) in \(T\), we keep \(R_{t'} = R_t\) and \(\alpha_{t'} = \alpha_t\). The first and third invariants are obviously preserved. In case \(\chi(t) = \chi(t') \setminus \{c_0\}\) for a coordinate \(c_0\), we have \(C_{t'} = C_t \cup \{c_0\}\). By the second invariant for \(t'\), for every \(\gamma \in D\) and \(\omega \in D^{C_t}\) the number of \(i \in [k]\) with \(R_{t'}[i][C_t] = \omega\) and \(R_{t}[i][c_0] = \gamma\) is \(\frac{k}{|D|^{C_t}}\). Therefore the number of \(i \in [k]\) with \(R_{t'}[i][C_t] = \omega\) is precisely \(\frac{k}{|D|^{C_t}} = \frac{k}{|D|^{C_{t'}}}\).

**t is an introduce node (introducing a vector).** Let \(t'\) be the child of \(t\) in \(T\) and \(\chi(t) = \chi(t') \cup \{v_0\}\) for some row label \(v_0\). Then we set \(R_{t'} = R_t\) and \(\alpha_{t'} = \alpha_t \cup \{v_0, i_0\}\) where \(i_0 \in [k]\) is such that \(R_{t'}[i_0][C_t] = A[v_0][C_t]\) and \(\alpha_{t'}(v) \neq i_0\) for every \(v \in V_t\). Such an index always exists: by the second invariant for \(t'\) there are \(\frac{k}{|D|^{C_{t'}}} \geq \frac{k}{|D|^{C_t}} = q > |V_t|\) indices \(i\) such that \(R_{t'}[i][C_t] = A[v_0][C_t]\), so at least one of them is not equal to any \(\alpha_t(v), v \in V_t\). As the third invariant holds in \(t'\), we have \(\sum_{v \in V_{t'}} \sum_{c \in C_{t'}} W[v, c] \cdot (A[v, c] - R_{t'}[\alpha_{t'}(v), c])^2 = \sum_{c \in C_{t'}} W[v_0, c] \cdot (A[v_0, c] - R_{t'}[\alpha_{t'}(v_0), c])^2\). Here \(W[v_0, c] = 0\) for every \(c \in C_{t'} \setminus C_t\) because such \(c\) never appears in the same bag with \(v_0\). Moreover, for \(c \in C_t\), \(R_{t'}[\alpha_{t'}(v)], = R_t[i_0, c] = A[v_0, c]\), so every summand is equal to zero.

**t is an introduce node (introducing a coordinate).** In case \(t'\) is the child of \(t\) in \(T\) and \(\chi(t) = \chi(t') \cup \{c_0\}\) for some coordinate \(c_0\), we set \(\alpha_{t'} = \alpha_t \cup \{c_0\}\), \(R_{t'}[i][c_0] = R_t[i][c_0]\) for every \(i \in [k]\) and define \(R_{t'}\) in the column \(c_0\) as follows. For every vector \(\omega \in D^{C_{t'}}\), let \(g_\omega^i\) be the set of indices \(i \in [k]\) such that \(R_{t'}[i][C_t] = \omega\). Recall that by the second invariant for \(t'\), \(g_\omega^{t'} = \frac{k}{|D|^{C_{t'}}}\). We subdivide \(g_\omega^i\) into \(|D|\) groups \(\{g_\omega^i \mid \gamma \in D\}\) of the same size \(\frac{k}{|D|^{C_{t'}}} \geq \frac{k}{|D|^{C_{t}}} = q \geq |V_t|\) (potentially one of the groups can contain \(\alpha_{t'}(v)\) for all \(v \in V_t\)). For every \(\gamma \in D\) and \(i \in g_\omega^i\) we then set \(R_{t'}[i, c_0] = \gamma\) and by this satisfy the second invariant in \(t\). The following validates the third one:

\[
\sum_{v \in V_{t'}} \sum_{c \in C_{t'}} W[v, c] \cdot (A[v, c] - R_{t'}[\alpha_{t'}(v), c])^2 = \sum_{v \in V_t} W[v, c_0] \cdot (A[v, c_0] - R_{t'}[\alpha_t(v), c_0])^2 = \sum_{v \in V_t} W[v, c_0] \cdot (A[v, c_0] - R_t[\alpha_t(v), c_0])^2.
\]

Fix \(v \in V_t\), let \(\gamma = A[v][C_t]\). Then \(\alpha_{t'}(v) \neq \alpha_t(v)\), so \(\alpha_{t'}(v) \in g_\gamma\) with \(\gamma = A[v, c_0]\). Therefore \(R_{t'}[\alpha_{t'}(v), c_0] = \gamma = A[v, c_0]\), i.e., every summand is equal to zero.

**t is a join node.** Let \(t_1, t_2\) be the two children of \(t\) in \(T\), recall that \(\chi(t_1) = \chi(t_2) = \chi(t)\). By the well-known separation property of tree-decompositions, \(\chi_t \setminus \chi_{t_1} = \chi(t)\) (Downey & Fellows, 2013; Cygan et al., 2015). Without loss of generality we may assume that \(\alpha_{t_1}\) and \(\alpha_{t_2}\) coincide on \(V_t\) and \(R_{t_1}[i, c] = R_{t_2}[i, c]\) for every \(i \in [k]\) and
With the storage method above in mind, observe first that we now estimate the running time of our procedure. To further specify how the entries \(\alpha\) match with \(\beta\) on \(V\). For a join node, finding a suitable permutation of \(\alpha\) can be copied by reference in constant time each. From the child node. Since the entry at a child node is never forgotten, the row indices are simply copied over. For the forgotten row and column indices are simply copied from \(\beta\). Then, we can correctly define \(\alpha\) on \(V\). Observe that the desired time bound that is linear in terms of \(|V|\) is satisfied. To show the desired time bound that is linear in terms of \(|V|\) we need to show that the entries \(\alpha\) and \(\beta\) agree on \(V\). Applying \(\alpha\) to rearrange the rows of \(\beta\) takes time \(O(|V|)\) since we pass the representations of \(\beta\) by reference. Now, to compose \(\alpha\) with \(\beta\), instead of computing the image value-by-value which would take linear time, we do the following constant-time deferred application. For the linked list that stores the values of \(\alpha\) on \(V\), we consider another type of node that contains a single permutation \(\alpha\) of \([k]\). This node in interpreted as follows: any pair \((v, i)\) that comes after this node is deemed to represent the pair \((v, \pi(v,i))\) instead. Using permutation nodes, \(\pi\) can be applied to \(\alpha\) in \(O(k+q)\) by computing the values on \(V\) explicitly and putting a permutation node \(\pi\) at the beginning of the linked list for \(V\), and another permutation node \(\pi^{-1}\) at the end of this list. Finally, the union of \(\alpha\) and \(\beta\) is computed in time \(O(q)\) by appending the linked list that stores the images of \(\alpha\) on \(V\) to the linked list of \(\alpha\) on \(V\) and storing the values on \(V\) separately. In the same fashion, for each \(i \in [k]\) we construct the row \(R[i]\) by concatenating the respective linked lists of \(R[i]\) and \(R[i]\).

After processing the root node \(r\), the entries of \(B\) can be restored efficiently in the following way. Recall that at most \(O((n + d) \cdot q)\) entries of \(B\) are “1”, as the graph \(G\) has a tree-decomposition of width \(q - 1\); only the corresponding \(O((n + d) \cdot q)\) entries of \(B\) are relevant, others need not to be restored. First, in time \(O(kd)\) we retrieve the rows of \(R\) from the computed representation and store each row as an array indexed by the column indices. The cluster assignment can be retrieved from the representation of \(\alpha\) in time \(O((n + d) \cdot k)\) by traversing through the corresponding linked list while maintaining the composition of all occurred permutation nodes (observe that at most \(2(n + d)\) permutation nodes were created throughout the execution).

Afterwards each relevant entry \(B[v, c]\) can be computed in constant time, by retrieving the cluster assignment \(\alpha(v)\) of \(v\) and the value \(R[\alpha(v), c]\) from the respective row array. Since the number of nodes in the tree-decomposition is bounded by \(O(n + d)\) and processing of each node takes time \(O(k^2q)\), the total running time is thus bounded by

\[
O((n + d) \cdot k^2 q) = (n + d) \cdot |D|^{O(q)},
\]

as desired. □

At this point, we can prove the main result of this section.

**Theorem 3.** **Bounded-domain MCME is fixed-parameter tractable when parameterized by tw\((G)\).**

**Proof.** Given an instance of **Bounded-Domain MCME**, we begin by constructing the incidence graph \(G\) and then applying the known 5-approximation algorithm to compute a tree-decomposition of \(G\) of width \(q - 1 \leq 5 \cdot tw(G)\).
in time $\mathcal{O}(\text{tw}(G_I)) \cdot (n + d)$. At this point we proceed by comparing $q$ and $k$, as follows:

- if $k \geq q \cdot |D|^q$, then we can correctly output “Yes” (along with a suitable witness) by using Lemma 2;

- on the other hand, if $k \leq q \cdot |D|^q$ then we instead invoke Theorem 1 to solve the instance.

The running time in the former case is upper-bounded by $(n + d) \cdot |D|^\mathcal{O}(\text{tw}(G_I))$, while in the latter case the bound is $(n + d) \cdot |D|^\mathcal{O}(k \cdot \text{tw}(G_I)) \leq (n + d) \cdot |D|^\mathcal{O}(\text{tw}(G_I))$. \hfill $\Box$

4. Handling Real-Valued Matrices with Treewidth

We now turn our attention to REAL-VALUED MCME. As was mentioned in the Introduction, a fixed-parameter algorithm for the problem parameterized by the incidence treewidth of $W$ would resolve a long-standing open problem as a special case, and so far remains beyond reach. However, we can still make tangible progress on the problem by considering the treewidth of a different representation of $W$. In particular, the primal graph $G_P$ of $W$ is a graph containing one vertex for each row of $W$, where two rows $a, b$ are adjacent if and only if there is a column $e$ such that $W[a, e] = W[b, e] = 1$. An example comparing primal and incidence graphs is provided in Figure 2.

Our aim in this section is to prove the next theorem.

**Theorem 4.** REAL-VALUED MCME is fixed-parameter tractable when parameterized by $\text{tw}(G_P)$.

Before we proceed to the proof, we will first introduce some additional notation that will be useful in the context of real-valued matrices. For every $V' \subseteq V_A$, let $\mu(V', c) = \arg\min f(x)$ for $f(x) = \sum_{v \in V'} W[v, c] \cdot (A[v, c] - x)^2$ and let $\Delta_c(V') = f(\mu(V', c))$. Intuitively, if vectors from $V'$ form a cluster then $\Delta_c(V')$ is a minimal sum of distances to the cluster center along the coordinate $c$.

**Observation 5.** For each coordinate $c$, if $V'' = \{z \in V' \mid W[z, c] = 1\} \neq \emptyset$ then

$$\mu(V', c) = \sum_{v \in V''} A[v, c] \quad \text{and}$$

$$\Delta_c(V') = \sum_{v \in V''} (A[v, c])^2 - \frac{(\sum_{v \in V''} A[v, c])^2}{|V''|}.$$

**Proof.** As $W$ is a binary matrix, we can rewrite $f(x) = \sum_{v \in V''} (A[v, c] - x)^2 = |V''| x^2 - 2 \sum_{v \in V''} A[v, c] \cdot x + \sum_{v \in V''} (A[v, c])^2$. By assumption, $|V''| \neq 0$, so $f(x)$ is a quadratic function and attains a minimum at an extreme point, i.e., at $x = \frac{\sum_{v \in V''} A[v, c]}{|V''|}$. The minimal value of $f$ can be computed as $\sum_{v \in V''} (A[v, c])^2 - \frac{(\sum_{v \in V''} A[v, c])^2}{|V''|}$. \hfill $\Box$

Now we are ready to present the proof of Theorem 4.

**Proof of Theorem 4.** As our initial step, we once again apply the known approximation algorithm (Bodlaender et al., 2016) to compute a nice tree-decomposition $(T, \chi)$ of $G_P$ of width $q \leq 5 \text{tw}(G_P)$. We keep the notations $r, V_t, V_t^+$ from the previous section. Note that now the bags in the tree-decomposition contain only vectors, so $V_t = \chi(t)$. In contrast to the previous section, for every node $t$ we denote by $C_t^+$ the set of all processed in $t$ coordinates, i.e., coordinates $c$ such that $W[v, c] \neq 0$ for some $v \in V_t^+ \setminus V_t$. A crucial observation is the following: if $c \in C_t^+$, then for all vectors $v \not\in V_t^+$ it holds that $W[v, c] = 0$, otherwise $v$ and $v_c$ would appear in the same bag. In other words, values of the cluster centers on $c \in C_t^+$ are not meaningful for vectors introduced outside of $T_t$.

We will design a leaf-to-root dynamic programming algorithm which computes a set of records $R_t$ at each node $t$ of $T$. For each way $p$ of partitioning the vectors in the bag $V_t$ into at most $k$ clusters, the record $R_t$ stores the minimum cost of clustering the vectors of $V_t^+$ in the coordinates of $C_t^+$, considering only the partitions of $V_t^+$ that extend $p$.

Formally, for a node $t$ of $T$ let $P_t$ be the set of all equivalence relations $p \subseteq V_t \times V_t$ with at most $k$ equivalence classes. The record $R_t$ of $t$ is a mapping from $P_t$ to $\mathbb{R}^+$. Observe that $|P_t| \leq (q + 1)^q + 1$.

To introduce the semantics of our records, let $B_t$ be the set of all real matrices with the row labels $V_t^+$ and the column labels $C_t^+$. Let $B_t$ be a matrix in $B_t$. We define the partial weighted distance from $B_t$ to $A$ in $t$ as follows:

$$\text{pwd}(B_t, t) = \sum_{v \in V_t^+} \sum_{c \in C_t^+} W[v, c] \cdot (A[v, c] - B_t[v, c])^2.$$

We also denote the number of forgotten clusters by $fcl(B_t, t)$, that is, the number of distinct rows in $B_t$ not
equal to any $B_i[v]$, $v \in V_t$. We say that $p \in \mathcal{P}_t$ with $||p|| \leq k$ equivalence classes is the $t$-partition of $B_t$ if the following conditions hold:

- for every $v, w \in V_t$ such that $(v, w) \in p$, $B_i[v] = B_i[w]$,
- $\text{fcl}(B_t, t)$ is at most $k - ||p||$.

The last two conditions are in fact equivalent to existence of a cluster-assignment $\varphi$ w.r.t. $B_t$ such that for every $u, v \in V_t$, $\varphi(u) = \varphi(v)$ if and only if $(u, v) \in p$.

We are now ready to define the record $\mathcal{R}_t$. For each $p \in \mathcal{P}_t$, we set $\mathcal{R}_t(p) = \tau$ if there exists $B_t \in \mathcal{B}_t$ such that:

- $p$ is the $t$-partition of $B_t$,
- $\text{pwd}(B_t, t) = \tau$, and
- $\forall B'_t \in \mathcal{B}_t$ such that $p$ is the $t$-partition of $B'_t$, $\text{pwd}(B'_t, t) \geq \tau$.

In this case we say that $B_t$ witnesses that $\mathcal{R}_t(p) = \tau$. Observe that $\mathcal{R}_t(p) < \infty$ since any $p \in \mathcal{P}_t$ is a $t$-partition of the zero matrix, and that $\mathcal{R}_t(p)$ is well-defined (in the sense that $\inf_{B_t \in \mathcal{B}_t} \text{pwd}(B_t, t)$ is achieved on $B_t$), since $\text{pwd}(B_t, t)$ is minimized on a matrix constructed by Observation 5 from an optimal cluster-assignment, and there is only a finite number of those.

Recall that for the root $r$ of $T$, we assume $V_r = \emptyset$. Hence $\mathcal{P}_r$ contains only one element $\emptyset$, and $\mathcal{R}_r(\emptyset)$ is equal to the minimum value of $\|W \circ (A - B)\|^2_F$ that can be achieved by any real matrix $B$ containing at most $k$ distinct rows. Indeed, if some coordinate $c$ is not processed in $r$, then $W[v, c] = 0$ for every row label $v$. Let us extend a witness $B_t$ of $\mathcal{R}_r$ to a matrix $B$ with column set $C_A$ by setting $B[v, c] = 0$ for every row label $v$ and every $c \notin C_A$. Then $\mathcal{R}_r(\emptyset) = \sum_{v \in V_t} \sum_{c \in C_A} W[v, c] \cdot (A[v, c] - B[v, c])^2 = \sum_{v \in V_t} \sum_{c \in C_A} W[v, c] \cdot (A[v, c] - B[v, c])^2$. Hence the instance is a YES-instance if and only if $\mathcal{R}_r(\emptyset) \leq \ell$.

We will show how to compute the records in a leaf-to-root fashion by proceeding along the nodes of $T$, in each of the following cases:

$t$ is a leaf node. Let $V_t = \{v\}$, then $\mathcal{P}_t$ contains only one equivalence relation $p = \{(v, v)\}$ and $\mathcal{R}_t(p) = 0$ as there are no processed coordinates.

$t$ is a forget node. Let $t'$ be the child of $t$ in $T$ and $V_t = V_{t'} \cup \{v_0\}$ for some vector $v_0$, we denote $C_{new} = C_t^l \setminus C_{t'}^l$. By branching over all $p' \in \mathcal{P}(t')$ such that $p'|_{V_t \times V_t} = p$, we compute

$$\mathcal{R}_t^0(p) = \min_{p'} (\mathcal{R}_{t'}(p') + \Delta(p')),$$

where

$$\Delta(p') = \sum_{[v] \in [p']} \sum_{c \in C_{new}} \Delta_c([v]).$$

For correctness, assume that $\mathcal{R}_t^0(p) = \tau = \mathcal{R}_{t'}(p') + \Delta(p')$ for some $p' \in \mathcal{P}(t')$ such that $p'|_{V_t \times V_t} = p$. Construct a matrix $B_t$ from the witness $B_{t'}$ of $\mathcal{R}_{t'}(p')$ by adding a new column for every label $c \in C_{new}$ as follows. For every $v \in V_{t'}$, we set $B_t[v, c] = \mu([v], c)$. If $u \in V_t \setminus V_{t'}$ is such that $B_t[u] = B_t[v]$ for some $v \in V_{t'}$, we set $B_t[u, c] = \mu([v], c)$. Otherwise we set $B_t[u, c] = 0$. To see that $p$ is a $t$-partition of $B_t$, assume firstly that there is $v \neq v_0$ such that $(v_0, v) \in p'$, then $\text{fcl}(B_t, t) = \text{fcl}(B_{t'}, t')$ and $||p|| = ||p'||$. Otherwise $\text{fcl}(B_t, t) \leq \text{fcl}(B_{t'}, t') + 1$ and $||p|| = ||p'|| - 1$; in any case $p$ is a $t$-partition of $B_t$. Recall that

$$\text{pwd}(B_t, t) = \sum_{v \in V_t} \sum_{c \in C_{new}} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 = \text{pwd}(B_{t'}, t') + \sum_{v \in V_{t'}} \sum_{c \in C_{new}} W[v, c] \cdot (A[v, c] - B_t[v, c])^2.$$  

As $C_{new} \cap C_{t'}^l = \emptyset$, we have $W[v, c] = 0$ for all $c \in C_{new}$ and $v \in V_{t'} \setminus V_{t'}$. Therefore,

$$\text{pwd}(B_t, t) = \text{pwd}(B_{t'}, t')$$

$$+ \sum_{v \in V_{t'}} \sum_{c \in C_{new}} W[v, c] \cdot (A[v, c] - \mu([v], c))^2$$

$$= \mathcal{R}_{t'}(p') + \Delta(p') = \tau,$$

and thus $B_t$ witnesses that $\mathcal{R}_t(p) \leq \tau$.

On the other hand, assume that $\mathcal{R}_t(p) = \tau$ and $B_t$ is a matrix in $B_t$ witnessing this. Let $B_{t'}$ be obtained from $B_t$ by deleting the columns with labels in $C_{new}$. We construct $p' \in \mathcal{P}(t')$ from $p$ by adding $v_0$ to an equivalence class of arbitrary $v \in V_{t'}$ such that $B_t[v_0] = B_t[v]$. If there is no such $v$, we create a new equivalence class $\{v_0\}$. In any case $p'$ is a $t'$-partition of $B_{t'}$: if $||p'|| = ||p||$ then $\text{fcl}(B_{t'}, t') = \text{fcl}(B_t, t)$, in case $||p'|| = ||p|| + 1$ we have $\text{fcl}(B_{t'}, t') = \text{fcl}(B_t, t) - 1$. Therefore $B_{t'}$ witnesses that $\mathcal{R}_{t'}(p') \leq \text{pwd}(B_{t'}, t')$

$$= \tau - \sum_{v \in V_{t'}} \sum_{c \in C_{new}} W[v, c] \cdot (A[v, c] - B_t[v, c])^2$$

$$\leq \tau - \sum_{v \in [p']} \sum_{c \in C_{new}} \Delta_c([v]) = \tau - \Delta(p').$$

So in our algorithm $\mathcal{R}_t^0(p) \leq \mathcal{R}_{t'}(p') + \Delta(p') \leq \tau$. Hence, we can correctly set $\mathcal{R}_t = \mathcal{R}_t^0$.

$t$ is an introduce node. Let $t'$ be the child of $t$ in $T$ and let $V_t = V_{t'} \cup \{v_0\}$ for some vector $v_0$. For every $p \in \mathcal{P}(t)$, we set $\mathcal{R}_t^1(p) = \mathcal{R}_{t'}(p')$ where $p' = p|_{V_{t'} \times V_{t'}}$. For correctness, it will be useful to observe that $C_t^l = C_{t'}^l$. Recall from
the first paragraph of the proof that $W[v_0, c] = 0$ for every $c \in C_1$. Assume $\mathcal{R}_t(p) = \tau$ with a witness $B_t$, let us obtain $B_t$ from $B_t$ by deleting the row with label $v_0$. Then $B_t$ admits the $t'$-partition $p'$ and $\text{pwd}(B_t, t') = \text{pwd}(B_t, t)$, so $B_t$ witnesses that $\mathcal{R}_t(p') \leq \tau$.

If on the other hand $\mathcal{R}_t(p') = \tau$ and $B_t$ is a matrix witnessing this, let us add to $B_t$ a row with the label $v_0$ so that $p$ is a $t$-partition of the resulting matrix $B_t$. For this, if $(v_0, v) \in p$ for some $v \in V_t$, we set $B_t[v_0] := B_t[v]$, which results in $|p| = |p'|$ and $\text{fcl}(B_t, t) = \text{fcl}(B_t, t')$. Otherwise $|p| = |p'| + 1$; we make $B_t[v_0]$ equal to any row of $B_t$ that is not among $\{B_t[v] \mid v \in V_t\}$ (if such a row exists) and by this achieve $\text{fcl}(B_t, t) = \text{fcl}(B_t, t') = 1$.

If all the rows of $B_t$ are among $\{B_t[v] \mid v \in V_t\}$, we set $B_t[v_0]$ equal to any of them. Then $p$ is a $t$-partition of $B_t$ and witnesses $\mathcal{R}_t(p) \leq \tau$, so we can correctly set $\mathcal{R}_t = \mathcal{R}_t^0$.

$t$ is a join node. Let $t_1, t_2$ be the children of $t$ in $T$. For every $p$ in $P_t$ we set $\mathcal{R}_t^0(p) = \mathcal{R}_{t_1}(p) + \mathcal{R}_{t_2}(p)$. For correctness, assume that $\mathcal{R}_{t_i}^0(p) = \tau = \tau_1 + \tau_2$ where $\tau_1 = \mathcal{R}_{t_1}(p)$, $\tau_2 = \mathcal{R}_{t_2}(p)$. Let $B_t$ be a witness of $\mathcal{R}_{t_i}(p)$ and let $\varphi_i$ be the cluster-assignment w.r.t. $B_t$ such that for every $u, v \in V_t$, $\varphi_1(u) = \varphi_2(v)$ if and only if $(u, v) \in p$, $i = 1, 2$. Without loss of generality we may assume that $\varphi_1|V_t = \varphi_2|V_t$. As $V_{t_1} \cap V_{t_2} = V_t$ and $C_{t_1} \cap C_{t_2} = \emptyset$, there exists a composition of $(B_t, \varphi_1)$ and $(B_t, \varphi_2)$, we denote it by $B_s$. Recall that $B_s$ admits the $t$-partition $\varphi = \varphi_1 \cup \varphi_2$, in particular, $p$ is a $t$-partition of $B_s$. As $W[v, c] = 0$ for every $c \in C_t^c$ and $v \notin V_t^c$, $i = 1, 2$, we have

$$\text{pwd}(B_s, t) = \sum_{v \in V_t} \sum_{c \in C_t^c} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 + \sum_{v \in V_t^c} \sum_{c \in C_t^c} W[v, c] \cdot (A[v, c] - B_t[v, c])^2 = \text{pwd}(B_t, t_1) + \text{pwd}(B_t, t_2) = \tau_1 + \tau_2 = \tau.$$ 

Hence $B_t$ witnesses that $\mathcal{R}_t(p) \leq \tau$.

For the converse, assume that $\mathcal{R}_t(p) = \tau$ and $B_t$ is a matrix witnessing this. Let $B_t$ be the restriction of $B$ to rows $V_t^c$ and columns $C_t^c$, $i = 1, 2$. Then $p$ is a $t$-partition of $B_1$ and $B_2$ and in our algorithm $\mathcal{R}_t^0(p) = \mathcal{R}_{t_1}(p) + \mathcal{R}_{t_2}(p) \leq \text{pwd}(B_1) + \text{pwd}(B_2) = \text{pwd}(B_t) = \tau$. So the resulting record $\mathcal{R}_t = \mathcal{R}_t^0$ is correct, which concludes the correctness proof of the algorithm.

To estimate the time complexity, recall that in a forget node we branch over at most $(q + 1)^{\tau+1} = tw(G_P)^\Omega(tw(G_P))$ partitions $p'$. Computing $\Delta(p')$ requires time at most $|C_{\text{new}}| \cdot (q + 1)^2 \leq |C_{\text{new}}| \cdot \Omega(tw(G_P)^2)$. Observe that each column appears in $C_{\text{new}}$ of at most one forget node in $T$, as all rows that have a value of “1” in the same column form a clique. Thus, the processing time of all forget nodes is upper-bounded by $d \cdot tw(G_P)^\Omega(tw(G_P))$. For the rest of the nodes, it is easy to see that the processing time of each node is upper-bounded by $tw(G_P)^\Omega(tw(G_P))$. Since the nice tree-decomposition $T$ has $\mathcal{O}(n)$ nodes, the total running time is at most $(n + d) \cdot tw(G_P)^\Omega(tw(G_P))$.

5. An Incidence-Graph Based Algorithm for Real-Valued MCME

While Theorem 4 significantly expands the previously known boundaries of tractability for MCME, the algorithm’s performance strongly depends on the structural properties of the primal graph. In general, primal graph representations are known to be denser than incidence graph representations, and this may make them unsuitable for the application of structure-based algorithms on certain instances (see, e.g., the example below Theorem 6).

As our final result, we show that although a fixed-parameter algorithm for REAL-VALUED MCME parameterized by $tw(G_1)$ remains beyond our reach, we can exploit a different parameter of the incidence graph to achieve fixed-parameter tractability—namely, the local feedback edge number.

**Theorem 6.** REAL-VALUED MCME is fixed-parameter tractable when parameterized by $\text{lfen}(G_1)$.

We note that it is not difficult to show that $\text{lfen}(G_1)$ and $tw(G_P)$ are pairwise incomparable parameterizations. Indeed, an $n \times 1$ mask consisting only of “1” entries has $\text{lfen}(G_1) = 0$ but $tw(G_P) = n - 1$. On the other hand, consider, for some integer $m$, an $\lfloor m \times 2 \lfloor m$ mask $W$ such that the first row $v_0$ of $W$ consists only of “1" entries, while for each $i \in [m]$ the $(i + 1)$-th row has “1" entries on precisely two positions: $i$ and $i + m$. Then $G_P$ is a star with center in $v_0$, so $tw(G_P) = 1$. However, $G_1$ consists of $m$ edge-disjoint cycles intersecting in $v_0$. It is then easy to observe that the local feedback edge number of $v_0$ with respect to any spanning tree of $G_1$ is $m$.

We proceed by introducing some additional terminology that will be useful in the coming arguments. Let $T$ be a fixed rooted spanning tree of $G$ such that $\text{lfen}(G, T) = \text{lfen}(G)$, denote the root by $r$. For $t \in V(T)$, let $T_t$ be the subtree of $T$ rooted at $t$. We define the boundary $\delta(t)$ of $t$ to be the set of endpoints of all edges in $G$ with precisely one endpoint in $V(T_t)$ (observe that the boundary can never have a size of 1). $t$ is called closed if $|\delta(t)| \leq 2$ and open otherwise.

**Proposition 7** (Ganian and Korchemna (2021)).

1. For every closed child $t'$ of $t$ in $T$, it holds that $\delta(t') = \{t, t'\}$ and $tt'$ is the only edge between $V(T_t)$ and $V(G) \setminus V(T_{t'})$ in $G$.
2. $|\delta(t)| \leq 2 \lfen(G) + 2$.
3. Let $\{t_i \mid i \in [j]\}$ be the set of all open children of $t$ in $T$. 

Then \( j \leq 2 \text{lfen}(G) \) and \( \delta(t) \subseteq \bigcup_{i=1}^{j} \delta(t_i) \cup \{t\} \cup N_G(t) \).

To prove Theorem 6, we will provide a leaf-to-root dynamic programming algorithm which stores information about optimal partitionings of \( \delta(t) \) into clusters. On a very intuitive level, Point 1. of Proposition 7 allows the algorithm to handle the closed children in a greedy manner. Point 2. ensures that the size of the records is bounded, and Point 3. is used in the dynamic programming step to compute records for a parent based on the records of its children. Furthermore, one can observe that \( |\bigcup_{i=1}^{j} \delta(t_i)| \) is upper-bounded by a linear function of \( \text{lfen}(t) \), which will be useful to ascertain the runtime bound for the algorithm.

**Observation 8.** For each node \( t \) in \( T \), \( |\bigcup_{i=1}^{j} \delta(t_i)| \leq 4 \text{lfen}(t) + 2 \).

**Proof.** By Point 2. of Proposition 7, the number of vertices in \( \bigcup_{i=1}^{j} \delta(t_i) \) that belong to \( \delta(t) \) is at most \( |\delta(t)| \leq 2 \text{lfen}(G) + 2 \). If \( v \in \delta(t_i) \setminus \delta(t) \) for some \( i \in [j] \), then \( v = t_i \). So the number of such \( v \) is at most \( j \leq 2 \text{lfen}(G) \) by Point 3. of Proposition 7. In total, \( |\bigcup_{i=1}^{j} \delta(t_i)| \leq 4 \text{lfen}(t) + 2 \).

We are now ready to establish the claimed result by providing an algorithm for **Real-Valued MCME** with running time \((kn^2 + d) \cdot \text{lfen}(G_j)^O(\text{lfen}(G_j))\), assuming a spanning tree \( T \) of minimum local feedback edge number is provided. We note that such a spanning tree can be computed by a fixed-parameter algorithm (Ganian & Korchemna, 2021).

**Proof of Theorem 6.** We will design a leaf-to-root dynamic programming procedure computing the set of records for every node of \( T \). Intuitively, the record in \( t \) will store the minimum sum of distances of processed vectors, computed along the processed coordinates, to their cluster centers for every possible partition of vectors in \( \delta(t) \) into clusters.

Formally, let \( V_t^+ \) and \( C_t^+ \) be the sets of vectors and coordinates of \( T_t \) correspondingly, we denote by \( V_t \) and \( C_t \) the restrictions of \( \delta(t) \) to vectors and coordinates respectively, then \( \delta(t) = V_t \cup C_t \). Let \( P_t \) be the set of all equivalence relations \( p \subseteq V_t \times V_t \) with at most \( k \) equivalence classes. The record \( R_t \) of \( t \) is a mapping from \( P_t \) to \( \mathbb{R}^+ \). Observe that \( |P_t| \leq \text{lfen}(G_j)^O(\text{lfen}(G_j)) \).

To introduce the semantics of our records, let \( B_t \) be the set of all real matrices with the row labels \( V_t^+ \cup V_t \) and the column labels \( C_t^+ \). Let \( B_t \) be a matrix in \( B_t \). We define the **partial weighted distance** from \( B_t \) to \( A \) in \( t \) as follows:

\[
\text{pwd}(B_t, t) = \sum_{v \in V_t \cup V_t^+} \sum_{c \in C_t^+} W[v, c] \cdot (A[v, c] - B_t[v, c])^2.
\]

For a matrix \( B_t \) with the row labels \( V_t^+ \cup V_t \) and the column labels \( C_t^+ \), we say that \( p \in P_t \) is the \( t \)-**partition** of \( B_t \) if the following conditions hold:

- for every \( v, w \in V_t \) such that \( (v, w) \in p \), \( B_t[v] = B_t[w] \);
- the number \( \text{fcl}(B_t, t) \) of pairwise different rows in \( B_t \) not equal to any \( B[v], v \in V_t \), is at most \( k - |p| \).

Note that the existence of a \( t \)-partition automatically implies that \( B_t \) has at most \( k \) distinct rows. We are now ready to define the record \( R_t \). For each \( p \in P_t \), we set \( R_t(p) = \tau \) if there exists \( B_t \in B_t \) such that:

- \( p \) is the \( t \)-partition of \( B_t \);
- \( \text{pwd}(B_t, t) = \tau \), and
- \( \forall B'_t \in B_t \) such that \( p \) is the \( t \)-partition of \( B'_t \), \( \text{pwd}(B'_t, t) \geq \tau \).

In this case we say that \( B_t \) **witnesses** that \( R_t(p) = \tau \).

Recall that for the root \( r \in T \), the boundary is empty, in particular \( V_r = \emptyset \), so \( P_r \) contains only one element \( \emptyset \), and \( R_r(\emptyset) \) is equal to the minimum value of \( \|W \circ (A - B)\|_F^2 \) that can be achieved by any real matrix \( B \) containing at most \( k \) distinct rows. Hence the instance is a **YES-instance** if and only if \( R_r(\emptyset) \leq \ell \). We will compute the records in a leaf-to-root fashion by proceeding along the nodes of \( T \).

Note that for any leaf vector \( v \), there are no coordinates in \( T_v \) and therefore \( R_v(p) = 0 \) for every \( p \in P_v \).

Let \( t = c \) be a (maybe leaf) coordinate with children \( v_1, \ldots, v_m \) where \( v_1, \ldots, v_j \) are open and the rest are closed. Recall from Proposition 7 that \( j \leq 2 \text{lfen}(G_j) \). We assume that the closed children are sorted by the coordinate \( c \), i.e., \( A[v_i, c] \leq A[v_{i+1}, c], i \in [m - 1] \setminus [j] \). Let us denote \( V_0 = \{v|(v, c) \in E(G_j) \setminus E(T)\} \) and for each \( i \in [m] \), \( V_i = V_0 \setminus C_i \). We observe that \( V_i \subseteq E_{\in\{j\}0} \).

For every \( p_0 \in P_t \), we initiate by setting \( R_t(p_0) = dnM_A + |\bigcup_{i \in [j]} V_i| - \sum_{v \in V_t} (A[v, c] - B_t[v, c])^2 \). In every branch we construct a partition \( p^* \) on the domain \( D^* = \bigcup_{i \in [m]} V_i \) extending \( p \) to the set of closed children.

To find the optimal (i.e., minimizing the sum of distances in the coordinate \( c \)) subdivision, we will compute the dynamic programming table \( \mathcal{M}(i, S, k_{cl}) \) for every subset \( S \subseteq [p] \) and non-negative integers \( i \in [j, m] \) and \( k_{cl} \leq k - |p| \), that stores the minimum sum of distances along the coordinate \( c \) for the closed children \( v_{j+1}, \ldots, v_i \) if they are partitioned into \( |S| + k_{cl} \) clusters with the following property: \( k_{cl} \) clusters do not contain any open children, and for each equivalence
Observe that if \( \varphi \) the matrix \( \tau \) ple back-tracking allows to construct the partition \( M_k \) for \( i \neq j \) and then for remaining triples \( S \subseteq [p], i \in [j + 1, m] \) and \( k \leq k - ||p|| \) compute:

\[
M_1(i, S, k) = \min_{\dot i \leq i \leq \min\{i, k-1\}} \left( M(i', S, k) + \Delta_c(v_{i',i}) \right),
\]

\[
M_2(i, S, k) = \min_{\dot i \leq i \leq \min\{i, k-1\}} \left( M(i', S \setminus \{v_i\}, k) + \Delta_c([v] \cup v_{i',i}) \right),
\]

where \( v_{i',i} = \{ v_y : y \in [i] \setminus [i'] \} \). As special cases, we set \( M_1(i, S, k) = d(m_A + 1) \) if \( k = 0 \), and \( M_2(i, S, k) = d(m_A + 1) \) if \( S = \emptyset \). Intuitively, \( M_1 \) corresponds to the case where the children in \( v_{i',i} \) form a separate cluster, while \( M_2 \) represents the addition of \( v_{i',i} \) to the cluster of open children from \( [i] \). \( M_2 \) also captures an option when no closed children are added to the cluster of \( [v] \), namely, when \( i = i' \). We set \( M(i, S, k) = \min(M_1(i, S, k), M_2(i, S, k)) \).

When all the entries of \( M \) are computed, let \( \Delta_c(p) = \min_{0 \leq k \leq k - ||p||} \left( M(m, [p], k) \right) - M(m, [p], k_0) \). A simple back-tracking allows to construct the partition \( p^* \) of \( D^* \) extending \( p \) such that \( [p^*] = \{ p \} + k_0 \) and \( \sum_{v \in [p]} \Delta_c([v]) = \Delta_c(p) \). We try to improve \( R^0(p_0) = \min_{\{p^* \}} \sum_{v \in [p]} R_{v,c}(p^*) \).

For correctness, assume that in our algorithm \( R^0(p) = \tau = \sum_{v \in [p]} R_{v,c}(p^*) \). For every \( i \in [j] \), let \( p_i = p^* \) and let \( B_i \) be the witness of \( R_{v,c}(p_i) \). We define the matrix \( B_0 \) with the only column label \( c \) and row labels \( V_0 \) by setting \( B_0(v,c) = \mu([v],c) \). Then, in particular, \( \sum_{w \in [v]} V_0 \cdot (A(v,c) - B_0(v,c))^2 = \Delta_c([v]) \) for every \( v \in [p^*] \). For every \( i \in [m_0] \), \( p_i = p^* \) and therefore we can construct cluster-assignments \( \varphi_i \) w.r.t. \( B_i \) such that

- for every \( u, v \in V_i \), \( \varphi_i(u) = \varphi_i(v) \) if and only if \( (u,v) \in p^*, i \in [m_0] \);
- for every \( v \in V_{i_1} \cap V_{i_2} \), \( \varphi_{i_1}(v) = \varphi_{i_2}(v) \), \( i_1, i_2 \in [m_0] \).

Note that the sets of column labels of \( B_i \) are pairwise disjoint, so there exists a composition \( B_i \) of \( (B_i, \varphi_i) \), \( i \in [m_0] \). Recall that \( B_i \) admits the cluster-assignment \( \varphi_i = \bigcup_{j \in [m_0]} \varphi_i \), has row labels \( \bigcup_{j \in [m_0]} (V_i \cup V_j) \cup V_0 = V_i \cup V_j \) and column labels \( \bigcup_{j \in [m_0]} C_i \cup \{c\} = C_i \). Let us compute \( \text{pwd}(B_i, t) = \sum_{v \in V_i \cup V_j} \sum_{c \in C_i} W[v,c] \cdot (A[v,c] - B_i[v,c])^2 \).

Observe that if \( c \in C_i \) and \( v \notin V_i \cup V_j \), then \( W[v,c] = 0, i \in [m] \). The set \( v \) such that \( W[v,c] = 1 \) is precisely \( V_0 \). Therefore \( \text{pwd}(B_i, t) \) can be computed as \( \sum_{i \in [m]} \sum_{v \in V_i \cup V_j} \sum_{c \in C_i} W[v,c] \cdot (A[v,c] - B_i[v,c])^2 + \sum_{v \in V_0} W[v,c] \cdot (A[v,c] - B_i[v,c])^2 = \sum_{i \in [m]} \text{pwd}(B_i, v_i) + \sum_{i \in [m]} \text{pwd}(B_i, v_{i',i}) \).

For the converse, assume that \( R_{v,c}(p_0) = \tau \) and \( B_i \) is a matrix witnessing this. There exists a cluster-assignment \( \varphi \) w.r.t. \( B_i \) such that for every \( u, v \in V_i \), \( \varphi(u) = \varphi(v) \) if and only if \( (u,v) \in p_0 \). We define the equivalence relation \( p^* \) on domain \( D^* \) as follows: \( (u,v) \in p^* \) if and only if \( \varphi(u) = \varphi(v) \). In particular, \( p^* \) extends \( p_0 \) and \( ||p|| \leq k \).

Claim 2. If \( u \) and \( v \) are closed children of \( c \) and \( A[u,c] \leq A[v,c] \) but \( B_i[u,c] \geq B_i[v,c] \), then there exists a witness \( B_i \in B_i \) of \( R_{v,c}(p_0) \leq \tau \) with cluster-assignment \( \varphi' \) such that:

- \( \varphi'(u) = \varphi(u), \varphi'(v) = \varphi(u) \);
- \( \varphi' \) coincides with \( \varphi \) outside of \( V_i \cup V_j \);
- \( B_i[u,c] = B_i[v,c], B_i[v,c] = B_i[u,c] \);
- \( B_i'[v_i,c] = B_i[v_i,c] \) for every \( v_i \notin \{u,v\}, i \in [m] \).

Proof. Let us define \( \varphi' \) as follows:

- if \( w \in V_i \cup V_j \) and \( \varphi(w) = \varphi(u) \), then \( \varphi'(w) = \varphi(u) \);
- if \( w \in V_i \cup V_j \) and \( \varphi(w) = \varphi(v) \), then \( \varphi'(w) = \varphi(v) \);
- for the rest of \( w \in V_i \cup V_j \), \( \varphi'(w) = \varphi(w) \).

We define the matrix \( B_i' \) with row and column labels \( V_i \cup V_j \) and \( C_i \) such that

- if \( \varphi'(w) \neq \varphi'(v) \) and \( \varphi'(w) \neq \varphi(u) \), we set \( B_i'[w] = B_i[w], w \in V_i \cup V_j \);
- \( B_i'[v,c] = B_i[v,c] \) for \( c \in C_i \cup C_i \);
- \( B_i'[u,c] = B_i[u,c] \) for the rest of \( c \);
- \( B_i'[v,c] = B_i[u,c] \) for \( c \in C_i \cup C_i \);
- \( B_i'[v,c] = B_i[u,c] \) for the rest of \( c \);
- \( \varphi'(w) = \varphi'(x) \), we set \( B_i'[w] = B_i'[x], x \in \{u,v\} \).

From the last definition it is clear that \( \varphi' \) is a cluster-assignment w.r.t. \( B_i' \). Observe that \( \varphi' \) coincides with \( \varphi \) on \( V_i \), in particular, \( p_0 \) is a \( t \)-partition of \( B_i' \). Let us compare \( \text{pwd}(B_i, t) \) and \( \text{pwd}(B_i', t) \). Summands corresponding to \( w \) with \( \varphi'(w) \neq \varphi'(v) \) and \( \varphi'(w) \neq \varphi(u) \) obviously coincide. Consider \( w \neq v \) with \( \varphi'(w) = \varphi(v) \). If \( w \in V_i \cup V_j \), then \( \varphi(w) = \varphi(v) \) and therefore for \( c \in C_i \cup C_i \), we have \( B_i[w,c] = B_i[v,c] = B_i[w,c] \). For \( c \notin C_i \cup C_i \), \( B_i[w,c] = B_i[w,c] = 0 \) as \( v \) and \( u \) are closed children. Assume that \( w \notin V_i \cup V_j \), then \( \varphi(w) = \varphi(u) \). For \( c \in C_i \cup C_i \), we have \( W[w,c] = 0 \). For the rest of \( c \), \( B_i'[w,c] = B_i'[v,c] = B_i[u,c] = B_i[w,c] \), so the
summands corresponding to \( w \) coincide. Similarly they coincide for \( w \neq u \) with \( \varphi'(w) = \varphi'(u) \). In particular, 
\[ B'_i[v_i, c] = B_i[v_i, c] \] for every \( v_i \notin \{ u, v \}, i \in [m] \). The only difference may occur in the summands corresponding to \( u \) and \( v \). Moreover, the only coordinate \( c' \) where \( W[v, c'] = 1 \) and \( B'_i[v, c'] \neq B_i[v, c'] \), is \( c' = c \), similarly for \( u \).

So comparing \( \text{pwd}(B_i, t) \) and \( \text{pwd}(B'_i, t) \) is in fact comparing \( (A[v, c] - B_i[v, c])^2 + (A[u, c] - B_i[u, c])^2 \) and \( (A[v, c] - B'_i[v, c])^2 + (A[u, c] - B'_i[u, c])^2 \). It is easy to see that the second sum is not larger then the first, see Figure 3 for the illustration.

![Figure 3](image_url)

In fact, Claim 2 allows us to assume without loss of generality that every cluster intersects the set of closed children of \( c \) by some \( v_{i_1, i_2} \), \( i_1 \leq i_2 \). Indeed, pick any closed children \( v_i \) and \( v_{i'} \) of \( c \) with \( i < i' \), then \( A[v_i, c] \leq A[v_{i'}, c] \). We will say that the pair \( (v_i, v_{i'}) \) is bad if one of the following holds:

- \( B_i[v_i, c] > B_i[v_{i'}, c] \);
- \( B_i[v_i, c] = B_i[v_{i'}, c], \varphi(v_i) \neq \varphi(v_{i'}) \) but there exists \( i'' > i' \) such that \( \varphi(v_i) = \varphi(v_{i''}) \).

Observe that if there are any bad pairs, we can choose one of them so that application of the Claim 2 to it decreases the number of bad pairs. Applying Claim 2 iteratively until there remain no bad pairs, we finally obtain a witness \( B^*_i \) of \( R_i(p_0) = \tau \) with cluster-assignment \( \varphi^* \) such that any cluster of \( \varphi^* \) intersects the set of closed children of \( c \) in some \( v_{i_1, i_2} \). Further we assume \( B^*_i = B_i, \varphi^* = \varphi \). By optimality of \( B_i \), we conclude that \( \sum_{v \in [p]} W[u, c] \cdot (A[u, c] - B_i[u, c])^2 = \Delta_c([v]) \) for every \( v \in [p]^* \). Every such case is captured by the computation scheme of the dynamic table \( \mathcal{M} \), so we have \( \mathcal{M}(m, [p], k_{C_i}^0) \leq \sum_{v \in [p]^*} \Delta_c([v]) \) where \( p \) is the restriction of \( p^* \) to \( D \setminus v_{i_1, i_2} \) and \( k_{C_i}^0 = |[p]^*| - |[p]| \).

Let \( B_i \) be the restriction of \( B^*_i \) to rows \( V_i \cup V_{i^*} \) and columns \( C_i, i \in [m] \). Then \( p^* \cap p_i \) is \( v_i \)-partition of \( B_i \) and hence in our algorithm

\[
R_i^0(p_0) \leq \sum_{i \in [m]} R_{C_i}(p^*_v) + \mathcal{M}(m, [p], k_{C_i}^0) \\
\leq \sum_{i \in [m]} \text{pwd}(B_i, v_i) + \sum_{v \in [p]^*} \Delta_c([v]) \\
= \text{pwd}(B_i, t) = \tau.
\]

It remains to consider the case where \( t = v \) is a vector with children \( c_1, \ldots, c_m \) in \( T \), where \( c_1, \ldots, c_j \) are open and the rest are closed. We will use the notations \( V_i = V_{c_i}, C_i = C_{c_i}, V^k_i = V_{c_i}^k \) and \( C_i^0 = C_{c_i}^0 \), \( i \in [m] \). For every \( p_0 \in \mathcal{P}_t \), we initiate by setting \( R_i(p_0) = ndM_i + 1 \) and then branch over equivalence relations \( p \) on \( \bigcup_{i \in [j]} V_i \cup \{ v \} \supseteq V_i \) such that \( |[p]| \leq k \) and \( p|_{V_i} = p_0 \). In every branch we set \( R_i^0(p_0) := \min(R_{C_i}(p_0), \sum_{i \in [m]} R_{C_i}(p_i)) \) where \( p_i = p|_{V_i} \), for \( i \in [j] \) and \( p_i = \{ (v, v) \} \) for \( i \in [m] \setminus [j] \).

For correctness, assume that in our algorithm \( R^0_i(p_0) = \tau = \sum_{i \in [m]} R_{C_i}(p_i) \) and let \( B_i \) be the witness of \( R_{C_i}(p_i) \). We define \( \varphi \) on the domain of \( p \) and with values in \([k]\) by setting \( \varphi(u) = \varphi(u) \) if and only if \( (u, u) \in p \). Recall that \( p_i = p|_{V_i} \), for \( i \in [j] \) and \( p_i = \{ (v, v) \} \) for \( i \in [m] \setminus [j] \), so there exist cluster-assignments \( \varphi_i \) w.r.t. \( B_i \) which agree with \( \varphi \) on intersections of their domains.

Sets of columns of \( B_i \) are pairwise disjoint, so there exists a composition matrix \( B_i \) with cluster-assignment \( \bigcup_{i \in [m]} \varphi_i = \varphi \), row labels \( \bigcup_{i \in [m]} (V_i \cup V_i^k) = V_i \cup V_i^k \) and column labels \( \bigcup_{i \in [m]} C_i^0 = C_i^0 \). Let us compute \( \text{pwd}(B_i, t) = \sum_{u \in V_i \cup V_i^k} \sum_{c \in C_i^0} W[u, c] \cdot (A[u, c] - B_i[u, c])^2 \) where \( C_i = \bigcup_{i \in [m]} C_i^0 \) and \( W[u, c] = 0 \) whenever \( c \in C_i^0 \) and \( u \notin V_i \cup V_i^k \). Therefore \( \text{pwd}(B_i, t) = \sum_{i \in [m]} \sum_{u \in V_i \cup V_i^k} \sum_{c \in C_i^0} W[u, c] \cdot (A[u, c] - B_i[u, c])^2 \) and \( R_i(p_0) = \tau \) if and only if \( (u, v) \in p \) if and only if \( \varphi(u) = \varphi(v) \). Let \( B_i \) be the restriction of \( B_i \) to
rows $V_i \cup V_i^\perp$ and columns $C_i^k$, $i \in [m]$, then $p_i = p|_{V_i}$ is $c_i$-partition of $B_i$, $i \in [j]$. Hence in our algorithm $\mathcal{R}_d(p_0) \leq \sum_{i \in [j]} \mathcal{R}_c(p_i) + \sum_{i \in [m] \setminus [j]} \mathcal{R}_c([v, v']) \leq \sum_{i \in [m]} \text{pwd}(B_i, c_i) = \text{pwd}(B_t, t) = \tau$. Hence the resulting record $\mathcal{R}_d = \mathcal{R}_d^0$ is correct, which concludes the correctness proof of the algorithm.

Let us estimate the time for processing coordinates $t = c$. For a fixed $c$ with $m_c$ children, we branch over equivalence relations $p$ on domain $\bigcup_{i \in [j]} V_i \subseteq \delta(c) \cup \bigcup_{i \in [j]} \delta(v_i)$. By Observation 8, the size of the domain is upper-bounded by $O(\lceil \text{len}(G_I) \rceil)$, so there are at most $\lceil \text{len}(G_I) \rceil^{O(\lceil \text{len}(G_I) \rceil)}$ branches. In every branch, we start from computing $\Delta_c([v] \cup v_i', i')$ and $\Delta_c(v_i', i')$ for each $[v] \in [p]$ and $i' < i$ from $[m_c] \setminus [j]$. Taking into account Observation 5, all the computations can be performed in time at most $m_c^2 \cdot |[p]|$. After this, every entry of $\mathcal{M}$ can be calculated in time $O(m_c \cdot |[p]|) \leq m_c \cdot O(\lceil \text{len}(G_I) \rceil)$. $\mathcal{M}$ has size at most $m_c \cdot 2^{|[p]|} \leq m_c \cdot 2^{O(\lceil \text{len}(G_I) \rceil)}$ and therefore can be computed for all the branches in time $m_c^2 \cdot |[p]|^{O(\lceil \text{len}(G_I) \rceil)}$. As $\sum_{c \in C_A} m_c^2 \leq (\sum_{c \in C_A} m_c)^2 \leq n^2$, processing all the coordinates $c$ takes time of at most $n^2 k \cdot \text{len}(G_I)^{O(\lceil \text{len}(G_I) \rceil)}$. Record for a vector $v$ with $m_v$ children can be computed in $m_v \cdot \text{len}(G_I)^{O(\lceil \text{len}(G_I) \rceil)}$, which yields the time $d \cdot \text{len}(G_I)^{O(\lceil \text{len}(G_I) \rceil)}$ for processing all the vectors. Therefore the total running time of the algorithm is upper-bounded by $(kn^2 + d) \cdot \text{len}(G_I)^{O(\lceil \text{len}(G_I) \rceil)}$. □

6. Concluding Remarks

While our algorithmic results are specifically designed to deal with means clustering with missing entries, it would be interesting to see whether the approaches and techniques developed here can be applied to other clustering variants or, e.g., the related task of low-rank matrix completion. Still, on the theoretical side the by far most prominent problem that is relevant to this research direction is the complexity of $k$-means clustering for real-valued matrices when parameterized by the number of columns. A W[1]-hardness result for this problem would immediately exclude the existence of a fixed-parameter algorithm for real-valued MCME parameterized by the incidence treewidth of the mask, while a fixed-parameter algorithm could potentially open up the way towards tractability.

It would also be interesting to see how the considered sparsity parameters behave in practical settings. In particular, even though direct implementations of our exact algorithms with runtime guarantees are unlikely to outperform state-of-the-art heuristics, it may be possible to exploit these parameters to guide or otherwise improve existing methods.

Acknowledgements

Robert Ganian, Thekla Hamm, Viktoriia Korchemnna, and Kirill Simonov acknowledge support by the Austrian Science Fund (FWF, projects Y1329 and P31336). Karolina Okrasa acknowledges support by the European Research Council, grant agreement No 714704; parts of this work were performed while visiting TU Wien, Vienna, Austria.

References


