Graph Clustering Problems Under the Lens of Parameterized Local Search

Jaroslav Garvardt
Institute of Computer Science, Friedrich Schiller University Jena, Germany

Nils Morawietz
Institute of Computer Science, Friedrich Schiller University Jena, Germany

André Nichterlein
Technische Universität Berlin, Germany

Mathias Weller
Technische Universität Berlin, Germany

Abstract

Cluster Editing is the problem of finding the minimum number of edge-modifications that transform a given graph G into a cluster graph G', that is, each connected component of G' is a clique. Similarly, in the Cluster Deletion problem, we further restrict the sought cluster graph G' to contain only edges that are also present in G. In this work, we consider the parameterized complexity of a local search variant for both problems: LS Cluster Deletion and LS Cluster Editing. Herein, the input also comprises an integer k and a partition $\mathcal{C}$ of the vertex set of G that describes an initial cluster graph G^*, and we are to decide whether the "k-move-neighborhood" of G^* contains a cluster graph G' that is "better" (uses less modifications) than G^*. Roughly speaking, two cluster graphs G_1 and G_2 are k-move-neighbors if G_1 can be obtained from G_2 by moving at most k vertices to different connected components.

We consider parameterizations by $k + \ell$ for some natural parameters ℓ, such as the number of clusters in $\mathcal{C}$, the size of a largest cluster in $\mathcal{C}$, or the cluster-vertex-deletion number (cvd) of G. Our main lower-bound results are that LS Cluster Editing is W[1]-hard when parameterized by k even if $\mathcal{C}$ has size two and that both LS Cluster Deletion and LS Cluster Editing are W[1]-hard when parameterized by $k + \ell$, where ℓ is the size of the largest cluster of $\mathcal{C}$. On the positive side, we show that both problems admit an algorithm that runs in $k^{\mathcal{O}(k)} \cdot \text{cvd}^{3k} \cdot n^{\mathcal{O}(1)}$ time and either finds a better cluster graph or correctly outputs that there is no better cluster graph in the k-move-neighborhood of the initial cluster graph.

As an intermediate result, we also obtain an algorithm that solves Cluster Deletion in $\text{cvd}^{\text{cvd}} \cdot n^{\mathcal{O}(1)}$ time.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases parameterized local search, permissive local search, FPT, W[1]-hardness

Digital Object Identifier 10.4230/LIPIcs.IPEC.2023.20

Funding *Jaroslav Garvardt*: Partially supported by the Carl Zeiss Foundation within the project "Interactive Inference".
Nils Morawietz: Partially supported by the DFG, project OPERAH, KO 3669/5-1.

1 Introduction

Graph-based data clustering is a fundamental task with numerous applications [40]. Within this broad setting, we focus on the approach of modfying an input graph into a cluster graph (that is, a disjoint union of cliques) with as few edge modifications as possible. Herein, edges may be deleted or inserted, leading to the well-known Cluster Editing or Correlation

Clustering [2, 4, 41] problem. If only edge deletions are allowed, then the problem is called Cluster Deletion [41]. One important advantage of the graph-modification approach is that the number of clusters is not part of the input but is determined implicitly.

Cluster Deletion and, in particular, Cluster Editing are highly relevant in practice, with application areas ranging from bioinformatics [4] to data mining [2] and psychology [46]. Unfortunately, it is NP-hard to decide whether a given graph is at most a given number of modifications away from being a cluster graph [41]. Therefore, efforts have been made to circumvent this hardness. One particularly successful approach are parameterized algorithms [8, 13, 18, 23, 26, 37]. Given the amount of research and the practical relevance, it is no surprise that Cluster Editing was selected as the problem for the sixth installment of the parameterized implementation challenge PACE 2021 [32]. The results revealed the strength of local search for Cluster Editing: The top ten submissions in the heuristic track all involve local search. Moreover, the top three submissions always returned a solution less than 1.001 times larger than the best known solution, that is, the relative error is below 10^{-3}. This is in stark contrast to the best known polynomial-time approximation having an approximation factor of 2.06 [12], thus having a relative error that is three orders of magnitude larger!

In this work we complement the results of the PACE 2021 heuristic track with a theoretical study of the local search problems associated with Cluster Editing and Cluster Deletion. More precisely, we study the following question: Can we improve a given clustering[1] of the input graph G by "moving" at most k vertices to different clusters? Many local search algorithms submitted to PACE 2021 try to move vertices between clusters to improve their solution [3, 7, 22, 32, 42]. For Cluster Editing we are free to move any vertex in any cluster (inserting missing edges within a cluster and deleting edges between clusters) while, for Cluster Deletion, we have to ensure that there are no missing edges within any cluster we create. The respective local search versions of the problems are called LS-Cluster Editing and LS-Cluster Deletion (see Section 2 for precise problem definitions).

Related Work. In the more general setting, our work fits into the theme of *parameterized local search*. Unfortunately, most parameterized local search problems turn out to be W[1]-hard with respect to their local search radius k [9, 15, 17, 21, 27, 28, 33, 39, 43]. Consequently, one often tries to combine k with some structural parameter ℓ in the hope of obtaining an FPT algorithm. Experimental evaluation showed that such algorithms often achieve very good solutions quickly [19, 20, 25, 29, 31].

Dörnfelder et al. [15] already proved the W[1]-hardness of a local search version of Cluster Editing with a different local neighborhood: they search for a better solution by modifying at most k edges in the given solution. Recently, Luo et al. [38] considered the closely related Dynamic Cluster Editing problem, in which a given clustering $\mathcal{C}$ for a graph G has to be adapted while the graph G changes dynamically, keeping the modification distance between them low. In this setting, the main question is whether minor changes to $\mathcal{C}$ are sufficient to produce a good clustering for the (slightly) changed new graph. Since, in this setting, the old graph G is actually irrelevant for the computational problem, there are only minor technical differences to LS-Cluster Editing. Luo et al. [38] measure the distance to $\mathcal{C}$ by the number of vertices moving to a different cluster (they call this "matching distance") – the same measure we use. They analyzed the parameterized complexity of

[1] A clustering can be viewed as an equivalence relation ("which vertices will end up in the same cluster?") or, equivalently, a partition of the vertex set of G.

Dynamic Cluster Editing with respect to the solution size ℓ (number of edges to modify in G) and the number k of changes allowed to be done to the given solution. They obtained W[1]-hardness for each of the parameters k and ℓ individually and fixed-parameter tractability with respect to the combined parameter $k + \ell$.

Besides the work on Dynamic Cluster Editing, there are numerous works on Cluster Editing, including search-tree based algorithms [8, 23], kernelizations [10, 13, 18, 26], and above lower-bound parameterizations [6, 37]. For Cluster Deletion, there are search-tree based algorithms [30, 45] and problem kernels [10, 11].

Our Results. We consider the number k of vertices allowed to be moved to different clusters, since this number can be expected to be small (mostly one-digit values in PACE 2021 local search submissions). We show that LS-Cluster Deletion and LS-Cluster Editing are – like many other local search problems – W[1]-hard with respect to the local search radius k, so we try to combine k with different structural parameters. If a parameter combination $k+\ell$ allows for fixed-parameter tractability, then we aim for running times of the form $\ell^{\mathcal{O}(k)} \cdot n^{\mathcal{O}(1)}$ as advocated by Komusiewicz and Morawietz [34]. The motivation is that k is expected to be much smaller than even ℓ which, in turn, is hopefully considerably smaller than n. Thus, the resulting algorithms are expected to be very efficient in practice, which is particularly important since local search subroutines are called excessively in the solvers (see for example the PACE 2021 solvers [32]).

We combine k with the maximum degree Δ, the maximum size of any clique in the given solution, or the cluster vertex deletion number cvd, that is, the number of vertices to remove to obtain a cluster graph. In Section 3, we present algorithms with running times of the form $(\ell \cdot k)^{\mathcal{O}(k)} n^{\mathcal{O}(1)}$ for $\ell = \Delta$ (see Theorem 3.3) and $\ell = \text{cvd}$ (see Corollary 3.8 and Theorem 3.9). As an intermediate result, we obtain an algorithm solving Cluster Deletion in $\text{cvd}^{\text{cvd}} \cdot n^{\mathcal{O}(1)}$ time (see Theorem 3.6). We complement the algorithms for LS-Cluster Deletion and LS-Cluster Editing with lower bounds in Section 4. In particular, we show that LS-Cluster Editing is W[1]-hard with respect to the sum of k, the maximum cluster size, and the degeneracy of the input graph G (see Theorem 4.2) and that LS-Cluster Editing is W[1]-hard with respect to k in the restricted case that the given clustering consists of only two clusters (see Theorem 4.5). LS-Cluster Deletion is also W[1]-hard with respect to the sum of k and the maximum cluster size (see Theorem 4.6). Moreover, the employed reductions also show that neither LS-Cluster Editing nor LS-Cluster Deletion can be solved in $f(k) \cdot n^{o(k)}$ time unless the ETH fails. This shows that in the above mentioned algorithms the $\mathcal{O}(k)$ in the exponent cannot be replaced by $o(k)$.

Due to space restriction, proofs of statements marked with ($\star$) are deferred to a full version.

2 Preliminaries

For details about relevant definitions of parameterized complexity such as fixed-parameter tractability, W[1]-hardness, parameterized reductions, kernelization, and ETH, we refer to the standard monographs [14, 16].

Let X and Y be sets. We denote by $A \oplus B := (A \setminus B) \cup (B \setminus A)$ the *symmetric difference* between A and B. A *partition* $\mathcal{P}$ of X is a collection of non-empty and pairwise disjoint subsets of X such that $\cup_{P \in \mathcal{P}} P = X$. Moreover, for an integer k, we denote by $\binom{X}{k}$ the collection of all size-k subsets of X. Let $G = (V, E)$ be a graph. For a vertex set $S \subseteq V$, we denote by $E_G(S) := \binom{S}{2} \cap E$ the edges of G between the vertices of S. If G is clear from the

context, we may omit the subscript. A partition $\mathcal{C}$ of V is called a *clustering of* G. Each vertex set C of $\mathcal{C}$ is called a *cluster*. For a clustering $\mathcal{C}$, we denote by $E(\mathcal{C}) := \bigcup_{C\in\mathcal{C}} \binom{C}{2}$ the edges inside the clusters of $\mathcal{C}$.

We call a function $\chi\colon V \to \mathbb{N}$ a *coloring* of V. We say that color $i \in \mathbb{N}$ is *used by* χ if there is at least one vertex $v \in V$ with $\chi(v) = i$. For a coloring χ, let $\mathcal{C}_\chi$ denote the clustering of G where for each color i used by χ, $\chi^{-1}(i) = \{v' \in V \mid \chi(v') = i\}$ is a cluster of $\mathcal{C}_\chi$. We call χ a *cluster-coloring* of $\mathcal{C}_\chi$ and we call $\mathcal{C}_\chi$ the *clustering* of χ. Note that each clustering has infinitely many cluster-colorings and that all these cluster-colorings are identical with respect to isomorphism. Here, two colorings χ and χ' are *isomorphic* if there is a bijection $f\colon \mathbb{N} \to \mathbb{N}$ such that $\chi = f \circ \chi'$.

Let χ and χ' be colorings of V. We denote by $\mathrm{Move}(\chi, \chi') := \{v \in V \mid \chi(v) \neq \chi'(v)\}$ the vertices that receive different colors under χ and χ'. Moreover, we set $\mathrm{move}(\chi, \chi') := |\mathrm{Move}(\chi, \chi')|$. Two colorings χ and χ' are *k-move-neighbors* if $\mathrm{move}(\chi, \chi') \leq k$. Analogously, two clusterings $\mathcal{C}$ and $\mathcal{C}'$ are *k-move-neighbors* if there is a cluster-coloring χ of $\mathcal{C}$ and a cluster-coloring χ' of $\mathcal{C}'$ such that χ and χ' are k-move-neighbors. We also denote by $\mathrm{move}(\mathcal{C}, \mathcal{C}')$ the smallest integer k such that $\mathcal{C}$ and $\mathcal{C}'$ are k-move-neighbors. Note that $\mathrm{move}(\mathcal{C}, \mathcal{C}')$ can be computed in polynomial time [38]. For a clustering $\mathcal{C}$ we define $\mathrm{cost}(\mathcal{C}) := |E(\mathcal{C}) \oplus E|$. A clustering $\mathcal{C}'$ is *improving* over a clustering $\mathcal{C}$, if $\mathrm{cost}(\mathcal{C}') < \mathrm{cost}(\mathcal{C})$.

For a function $f\colon A \to B$ and $C \subseteq A$ we denote by $f|_C$ the function f restricted to C.

▶ **Observation 2.1.** *Let G be a graph and let $\mathcal{C}$ and $\mathcal{C}'$ be clusterings of G with $\mathrm{move}(\mathcal{C}, \mathcal{C}') \leq k$. Then, $|\mathcal{C} \cap \mathcal{C}'| \geq |\mathcal{C}| - 2k$.*

In this work, we consider the parameterized complexity of the following problems.

LS-Cluster Deletion
Input: An undirected graph $G = (V, E)$, an integer k and a clustering $\mathcal{C}$ of G with $E(\mathcal{C}) \subseteq E$.
Question: Is there an improving k-move-neighbor $\mathcal{C}'$ of $\mathcal{C}$ for G with $E(\mathcal{C}') \subseteq E$?

LS-Cluster Editing
Input: An undirected graph $G = (V, E)$, an integer k and a clustering $\mathcal{C}$ of G.
Question: Is there an improving k-move-neighbor $\mathcal{C}'$ of $\mathcal{C}$ for G?

Further, we also analyze the *permissive* version of both problems, that is, the problem, where we want to find any better solution or correctly output that the given solution has no improving k-move-neighbor. Thus, the permissive problem variants allow to return better clusterings even if these are not k-move-neighbors of $\mathcal{C}$.

3 Algorithms for Permissive Problem variants

In this section, we present our algorithmic results. We start with the parameterization maximum degree Δ and k. A first observation allows us to assume that any given clustering in our instance of LS-Cluster Editing consists exclusively of clusters that have diameter at most two in the input graph. Note that for LS-Cluster Deletion, the input clusters in the given clustering must all have diameter one, that is, they are cliques in the input graph.

▶ **Observation 3.1.** *Let G be a graph and let $\mathcal{C}$ be a clustering of G. If there is a cluster $C \in \mathcal{C}$ that has diameter at least three in G, then there is an improving 1-move-neighbor $\mathcal{C}'$ of $\mathcal{C}$.*

Proof. Let $u, v \in C$ be two vertices of distance at least three in G. Thus, we have $N(u) \cap N(v) = \emptyset$. Assume without loss of generality that $|N(v) \cap C| \leq |N(u) \cap C|$.

We show that $\mathcal{C}' := (\mathcal{C} \setminus \{C\}) \cup \{C \setminus \{v\}, \{v\}\}$ is an improving 1-move-neighbor of $\mathcal{C}$. Clearly, $\mathcal{C}'$ is a 1-move-neighbor of $\mathcal{C}$ as only v moves to a previously empty cluster. Moreover, this move costs $|N(v) \cap C|$ edge deletions but saves at least $|N(u) \cap C| + 1$ many edge insertions. Since $|N(v) \cap C| \leq |N(u) \cap C|$, it follows that $\mathcal{C}'$ is improving over $\mathcal{C}$. ◀

With the knowledge that the given clusters are connected, it is not too hard to see that we can restrict ourselves to looking for solutions where the vertices that change clusters are not too far apart from each other; otherwise we can simply ignore parts of the changes and obtain a better clustering by moving even fewer vertices. The formal statement is as follows:

▶ **Lemma 3.2** (⋆). *Let G be a graph and $\mathcal{C}$ a clustering and $\mathcal{C}'$ an improving clustering for $\mathcal{C}$. Let $C_1, \ldots, C_q$ be the clusters in $\mathcal{C}$ that change in $\mathcal{C}'$ (that is, $\{C_1, \ldots, C_q\} := \mathcal{C} \setminus \mathcal{C}'$). If $G[C_1 \cup \ldots \cup C_q]$ is not connected, then there is an improving clustering $\mathcal{C}^*$ of $\mathcal{C}$ with $\mathrm{move}(\mathcal{C}, \mathcal{C}^*) < \mathrm{move}(\mathcal{C}, \mathcal{C}')$. Moreover, if $E(\mathcal{C}') \subseteq E$, then $E(\mathcal{C}^*) \subseteq E$.*

With Observation 3.1 and Lemma 3.2 we can show that checking for solutions for LS-Cluster Editing and LS-Cluster Deletion boils down to finding subgraphs of bounded size in graphs of bounded degrees for which we can apply an algorithm of Komusiewicz and Sommer [35]. Overall this results in the following statement.

▶ **Theorem 3.3** (⋆). *LS-Cluster Editing and LS-Cluster Deletion can be solved in $\Delta^{8k}(6ek)^{2k+1}n^{\mathcal{O}(1)}$ time.*

We remark that an algorithm solving LS-Cluster Editing or LS-Cluster Deletion in $f(\Delta) \cdot n^{\mathcal{O}(1)}$ time is unlikely: Setting $k := n$ and applying this algorithm repeatedly until no improvement is found would solve Cluster Editing or Cluster Deletion in $f(\Delta) \cdot n^{\mathcal{O}(1)}$ time; this is unlikely as both problems are NP-hard for constant maximum degree [36].

Parameterization by cluster vertex deletion number and k. In the remainder of the section we provide two algorithms exploiting small modulators to cluster graphs – one for LS-Cluster Deletion in Section 3.1 and one for LS-Cluster Editing in Section 3.2. For both of our algorithms, we use the following notation. We say that a vertex set $M \subseteq V$ is a *(cluster) modulator* of G if $G[V \setminus M]$ is a cluster graph. Let $\mathcal{B}$ be the connected components of $G[V \setminus M]$. We call each clique $B \in \mathcal{B}$ a *bag*. We denote by $\mathrm{cvd}(G)$ the cluster vertex deletion number, that is, the size of a smallest cluster modulator of G. If the graph G is clear from the context, we may simply write cvd.

3.1 An Algorithm for LS-Cluster Deletion

Let $G = (V, E)$ be a graph and let $\mathcal{C}$ be a clustering of G. Moreover, let $S \subseteq V$ and let $\mathcal{C}_S$ be a clustering of $G[S]$. We say that $\mathcal{C}$ *extends* $\mathcal{C}_S$ if for each cluster $C \in \mathcal{C}_S$ there is a cluster $C' \in \mathcal{C}$ with $C' \cap S = C$.

▶ **Lemma 3.4.** *Let $G = (V, E)$ be a graph and let M be a cluster modulator of G. Moreover, let $\mathcal{C}_M$ be a given clustering of $G[M]$ with $E(\mathcal{C}_M) \subseteq E$. In $3^{|\mathcal{C}_M|} \cdot n^{\mathcal{O}(1)}$ time, one can find a best clustering $\mathcal{C}$ of G among all clusterings $\mathcal{C}'$ of G with $E(\mathcal{C}') \subseteq E$ that extend $\mathcal{C}_M$.*

Proof. Note that for each clustering $\mathcal{C}$ of G that extends $\mathcal{C}_M$, the edges between vertices of M in both $E(\mathcal{C})$ and $E(\mathcal{C}_M)$ are equal. Consequently, the task can also be reformulated as follows: find a clustering $\mathcal{C}^*$ of G that maximizes the number of edges having at least one endpoint in $V \setminus M$ among all clusterings $\mathcal{C}$ of G with $E(\mathcal{C}) \subseteq E$ that extend $\mathcal{C}_M$. In the following, we describe a dynamic program solving this reformulated task.

Fix an arbitrary ordering of the bags and let B_i denote the ith bag of $\mathcal{B}$ according to this ordering. The dynamic programming table T has entries of type $T[X, i]$ with $X \subseteq \mathcal{C}_M$ and $i \in [0, |\mathcal{B}|]$. For $X \subseteq \mathcal{C}_M$ and $i \in [0, |\mathcal{B}|]$, let V_X denote the union of all clusters of X and let V_X^i denote the union of V_X and the first i bags. The entry $T[X, i]$ stores the maximal number of edges having at least one endpoint in $V_X^i \setminus M$ of any clustering $\mathcal{C}_X^i$ of $G[V_X^i]$ with $E(\mathcal{C}) \subseteq E$ that extends X. Intuitively, this entry stores the best way to distribute the vertices of the first i bags among the clusters of X.

To compute an entry $T[X, i]$, we iterate over all subsets X' of X and check for the best way to distribute the vertices of the first $i-1$ bags among the clusters of X' and the best way to distribute the vertices of the ith bag among the clusters of $X \setminus X'$.

Formally, for each $X \subseteq \mathcal{C}_M$, we set $T[X, 0] := 0$ and for each $i \in [1, |\mathcal{B}|]$, we set

$$T[X, i] := \max_{X' \subseteq X} T[X', i-1] + \text{gain}_i^{X \setminus X'}.$$

Here, $\text{gain}_i^{X \setminus X'}$ is the number of edges having at least one endpoint in bag B_i of the best way to distribute the vertices of B_i among the clusters of $X \setminus X'$. This recurrence is correct because no cluster C in a clustering $\mathcal{C}$ of G can contain vertices of two distinct bags without violating $E(\mathcal{C}) \subseteq E$.

In the following, we describe how to compute gain_i^Y for each $i \in [1, |\mathcal{B}|]$ and each $Y \subseteq \mathcal{C}_M$.

▷ Claim 3.5. In $3^{|\mathcal{C}_M|} \cdot n^{\mathcal{O}(1)}$ time, the values gain_i^Y can be computed for all $i \in [1, |\mathcal{B}|]$ and all $Y \subseteq \mathcal{C}_M$.

Proof. The computation of this value relies on the following observation: Let $\mathcal{C}$ be a best clustering of $G[V_Y \cup B_i]$ with $E(\mathcal{C}) \subseteq E$ that extends Y. Moreover, let C be a largest cluster of $\mathcal{C}$. Then C contains all vertices of B_i that are adjacent to all vertices of $C \cap M$. This is true, since if there would be a cluster C' in $\mathcal{C}$ containing a vertex v of $\{v \in B_i \mid C \subseteq N(v)\}$, then $\mathcal{C}' := (\mathcal{C} \setminus \{C, C'\}) \cup \{C \cup \{v\}, C' \setminus \{v\}\}$ is a clustering with $E(\mathcal{C}') \subseteq E$ that improves over $\mathcal{C}$. The properties of $\mathcal{C}'$ hold since C is a largest cluster of $\mathcal{C}$ and B_i is a clique in G.

Hence, to solve this intermediate task, we can branch which cluster C of $Y \cup \{\emptyset\}$ will be extended to be the largest cluster in $\mathcal{C}$, add all vertices of B_i to C that may fit into this cluster and solve the task recursively.

This can also be done by a dynamic program. We introduce the dynamic programming table D_i with entries of type $D_i[Y, Z]$ with $Y \subseteq \mathcal{C}_M$ and $Z \subseteq Y$.

For each set $Z \subseteq Y$, we let $\text{Remain}_Y^Z := B_i \setminus \left(\bigcup_{C \in Y \setminus Z} \{v \in B_i \mid C \subseteq N(v)\}\right)$ denote the set of vertices of B_i that do not fit in any cluster of $Y \setminus Z$. The entry $D_i[Y, Z]$ stores the maximal number of edges having at least one endpoint in Remain_Y^Z of any clustering $\mathcal{C}_Z^i$ of $G[\text{Remain}_Y^Z \cup \bigcup_{C \in Y} C]$ with $E(\mathcal{C}) \subseteq E$ that extend Z.

For each $Y \subseteq \mathcal{C}_M$, we set $D_i[Y, \emptyset] := \binom{|\text{Remain}_Y^\emptyset|}{2}$ and for each non-empty $Z \subseteq Y$, we set

$$D_i[Y, Z] := \max\left(\binom{|\text{Remain}_Y^Z|}{2}, \max_{C \in Z}\left(\binom{|\text{Fit}_C|}{2} + |\text{Fit}_C| \cdot |C| + D_i[Y, Z \setminus \{C\}]\right)\right),$$

where $\text{Fit}_C := \{v \in \text{Remain}_Y^Z \mid C \subseteq N(v)\}$ denotes the set of vertices of Remain_Y^Z that fit into the cluster C.

This recurrence is correct by the above observation: in each optimal clustering $\mathcal{C}_Z^i$ of $G[\text{Remain}_Y^Z \cup \bigcup_{C\in Y} C]$ with $E(\mathcal{C}) \subseteq E$ that extend Z, there is a largest cluster $C' \in \mathcal{C}_Z^i$ that contains all vertices of $\text{Fit}_{C'\cap M}$.

Finally, for each $Y \subseteq \mathcal{C}_M$, we set $\text{gain}_i^Y := D_i[Y,Y]$. Since for each $i \in [1,|\mathcal{B}|]$, the table D_i contains $3^{|\mathcal{C}_M|}$ entries and each such entry can be computed in $n^{\mathcal{O}(1)}$ time, the values gain_i^Y can be computed for all $i \in [1,|\mathcal{B}|]$ and all $Y \subseteq \mathcal{C}_M$ in the stated running time.

Moreover, note that a corresponding clustering can be computed via traceback. ◁

Let $\mathcal{C}^*$ be any best clustering of G among all clusterings $\mathcal{C}$ of G with $E(\mathcal{C}) \subseteq E$ that extend $\mathcal{C}_M$. Then, the number of edges of $E(\mathcal{C}^*)$ having at least one endpoint in any bag is stored in $T[\mathcal{C}_M, |\mathcal{B}|]$. Moreover, a corresponding clustering can be found via traceback.

Since for each $i \in [0,|\mathcal{B}|]$ and each $X \subseteq \mathcal{C}_M$, the table entry $T[X,i]$ can be computed in $2^{|X|}\cdot n^{\mathcal{O}(1)}$ time and there are $2^{|\mathcal{C}_M|}$ choices for X, each table entry of T can be computed in time $\sum_{i=1}^{|\mathcal{C}_M|}\binom{|\mathcal{C}_M|}{i}\cdot 2^i\cdot n^{\mathcal{O}(1)} \subseteq 3^{|\mathcal{C}_M|}\cdot n^{\mathcal{O}(1)}$. ◀

Note that this implies the following FPT-algorithm for Cluster Deletion when parameterized by cvd: First, compute a minimum cluster modulator M of G in $1.811^{\text{cvd}}\cdot n^{\mathcal{O}(1)}$ time [44]. Second, iterate over all possible clusterings of $G[M]$ and apply the algorithm behind Lemma 3.4. This implies a running time of $1.811^{\text{cvd}}\cdot \mathbf{B}_{\text{cvd}}\cdot n^{\mathcal{O}(1)}$, where $\mathbf{B}_{\text{cvd}}$ is the cvd-th Bell number which denotes the number of partitions of a set of size cvd. Since for each $n\in\mathbb{N}$, $\mathbf{B}_n < (\frac{n}{\ln(n+1)})^n$ [5], this implies the following.

▶ **Theorem 3.6.** *Cluster Deletion can be solved in* $\text{cvd}^{\text{cvd}}\cdot n^{\mathcal{O}(1)}$ *time.*

Next, we show how we can use Lemma 3.4 to obtain a permissive algorithm for LS-Cluster Deletion when parameterized by cvd and k.

▶ **Theorem 3.7.** *Let G be a graph and let M be a given cluster modulator of G. Moreover, let $\mathcal{C}$ be a clustering of G with $E(\mathcal{C}) \subseteq E$ and let $k\in\mathbb{N}$. In $|M|^{2k}\cdot\binom{|M|}{k}\cdot k^k\cdot 3^{4k}\cdot n^{\mathcal{O}(1)}$ time, one can find a clustering $\mathcal{C}'$ of G with $E(\mathcal{C}') \subseteq E$ that is at least as good as a best clustering $\mathcal{C}^*$ of G with $E(\mathcal{C}^*)\subseteq E$ and $\text{move}(\mathcal{C},\mathcal{C}^*)\leq k$.*

Proof. Let $\mathcal{C}^*$ be a clustering of G that maximizes $|E(\mathcal{C}^*)|$ among all clusterings $\mathcal{C}''$ of G with $E(\mathcal{C}'')\subseteq E$ and $\text{move}(\mathcal{C},\mathcal{C}'')\leq k$. Moreover, let $\mathcal{M} := \{C\in\mathcal{C} \mid C\cap M\neq\emptyset\}$ and $\mathcal{M}^* := \{C\in\mathcal{C}^* \mid C\cap M\neq\emptyset\}$ denote the sets of clusters intersecting M, of $\mathcal{C}$ and $\mathcal{C}^*$, respectively.

Let $\mathcal{C}_M := \{C\cap M \mid C\in\mathcal{M}\}$ denote the clusters of $\mathcal{M}$ restricted to the vertices of M. Similarly, let $\mathcal{C}_M^* := \{C\cap M \mid C\in\mathcal{M}^*\}$. Note that $\text{move}(\mathcal{C},\mathcal{C}^*)\leq k$ implies $\text{move}(\mathcal{C}_M,\mathcal{C}_M^*)\leq k$. Hence, to find a clustering $\mathcal{C}'$ of G that is at least as good as $\mathcal{C}^*$, it suffices to enumerate all clusterings $\mathcal{C}'_M$ of $G[M]$ with $\text{move}(\mathcal{C}_M,\mathcal{C}'_M)\leq k$ (which includes $\mathcal{C}_M^*$) and to compute, for each such clustering $\mathcal{C}'_M$, any best clustering for G that extends $\mathcal{C}'_M$. As the latter task can be done in $3^{|\mathcal{C}'_M|}\cdot n^{\mathcal{O}(1)}$ time by Lemma 3.4, it remains to describe how one can enumerate all such clusterings $\mathcal{C}'_M$ of $G[M]$.

This can be done in $\binom{|M|}{k}\cdot(|\mathcal{M}|+k)^k\cdot n^{\mathcal{O}(1)}$ time by iterating over all possible subsets $M'\subseteq M$ of size k and iterating over all possible ways to move these k vertices into any of the clusters of $\mathcal{M}$ (including the clusters where these vertices came from) or opening a new cluster.

Hence, this algorithm runs in $\binom{|M|}{k}\cdot(|\mathcal{M}|+k)^k\cdot 3^{|\mathcal{M}|+k}\cdot n^{\mathcal{O}(1)}$ time. Note that this is not the desired running time since $|\mathcal{M}|$ occurs in the exponent of the running time and might be much larger than k.

To still obtain the desired running time, we perform some initial branching if $|\mathcal{M}| > 2k$. The idea behind this initial branching relies on Observation 2.1. Since at most k vertices were moved to obtain $\mathcal{M}^*$ from $\mathcal{M}$, Observation 2.1 implies $|\mathcal{M} \cap \mathcal{M}^*| \geq |\mathcal{M}| - 2k$. In other words, there is a subset $\mathcal{M}' \subseteq \mathcal{M}$ of size at most $2k$ such that $\mathcal{M} \setminus \mathcal{M}' \subseteq \mathcal{M} \cap \mathcal{M}^*$. This implies that all edge modifications having at least one endpoint in any cluster of $\mathcal{M} \setminus \mathcal{M}'$ are identical in both $E(\mathcal{C})$ and $E(\mathcal{C}^*)$. Hence, by applying for each subset $\mathcal{M}' \subseteq \mathcal{M}$ of size at most $2k$ the above described algorithm on the graph $G[V \setminus (\cup_{C \in \mathcal{M} \setminus \mathcal{M}'} C)]$, we find a clustering $\mathcal{C}'$ with $E(\mathcal{C}') \subseteq E$ which is at least as good as $\mathcal{C}^*$. This initial branching can be done in $|\mathcal{M}|^{2k} \cdot n^{\mathcal{O}(1)} \subseteq |M|^{2k} \cdot n^{\mathcal{O}(1)}$ time.

Since for each such branching-instance, there are at most $2k$ clusters containing vertices of M, the whole running time evaluates to $|M|^{2k} \cdot \binom{|M|}{k} \cdot (3k)^k \cdot 3^{3k} \cdot n^{\mathcal{O}(1)} = |M|^{2k} \cdot \binom{|M|}{k} \cdot k^k \cdot 3^{4k} \cdot n^{\mathcal{O}(1)}$ time. ◀

Since a cluster modulator can be 2-approximated in polynomial time [1], Theorem 3.7 implies the following:

▶ **Corollary 3.8.** *The permissive version of LS-Cluster Deletion can be solved in* $(k^k + 2^{\mathcal{O}(k)} \cdot \text{cvd}^{3k}) \cdot n^{\mathcal{O}(1)}$ *time.*

Proof. Let $I := (G = (V, E), k, \mathcal{C})$ be an instance of LS-Cluster Deletion. First, check in $1.811^k \cdot n^{\mathcal{O}(1)}$ time, whether G has a cluster modulator of size at most k [44]. If this is the case, one can find an optimal clustering $\mathcal{C}'$ for G with $E(\mathcal{C}') \subseteq E$ in time $\text{cvd}^{\text{cvd}} \cdot n^{\mathcal{O}(1)} \subseteq k^k \cdot n^{\mathcal{O}(1)}$ due to Theorem 3.6. Otherwise, $k < \text{cvd}$. In this case, one can 2-approximate a cluster modulator M in polynomial time [1] and find a clustering $\mathcal{C}'$ of G with $E(\mathcal{C}') \subseteq E$ that is at least as good as a best clustering $\mathcal{C}^*$ of G with $E(\mathcal{C}^*) \subseteq E$ and $\text{move}(\mathcal{C}, \mathcal{C}^*) \leq k$ in time $|M|^{2k} \cdot \binom{|M|}{k} \cdot k^k \cdot 3^{4k} \cdot n^{\mathcal{O}(1)}$ due to Theorem 3.7. Since $k < \text{cvd} \leq |M|$, we get that $\binom{|M|}{k} \cdot k^k \leq |M|^k \cdot \frac{k^k}{k!} \leq |M|^k \cdot 2^{\mathcal{O}(k)}$. Hence, the running time of the case $k < \text{cvd}$ evaluates to $2^{\mathcal{O}(k)} \cdot |M|^{3k} \cdot n^{\mathcal{O}(1)} \subseteq 2^{\mathcal{O}(k)} \cdot (2 \cdot \text{cvd})^{3k} \cdot n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k)} \cdot \text{cvd}^{3k} \cdot n^{\mathcal{O}(1)}$ time. In both cases, the running time is upper-bounded by $(k^k + 2^{\mathcal{O}(k)} \cdot \text{cvd}^{3k}) \cdot n^{\mathcal{O}(1)}$ time. ◀

3.2 An Algorithm for LS-Cluster Editing

In this subsection, we present a permissive algorithm for LS-Cluster Editing with a running time similar to the one for LS-Cluster Deletion.

▶ **Theorem 3.9.** *Let $G = (V, E)$ be a graph, let $\mathcal{C}$ be a clustering of G, and let $k \in \mathbb{N}$. In $2^{\mathcal{O}(k)} \cdot k^k \cdot \text{cvd}^{3k} \cdot n^{\mathcal{O}(1)}$ time, one can find a clustering that improves over $\mathcal{C}$ or correctly output that there is no clustering $\mathcal{C}'$ of G with $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$ that improves over $\mathcal{C}$.*

To give a better intuition for the following algorithm, we switch to the interpretation of LS-Cluster Editing where the initial solution is a cluster coloring $\chi_{\mathcal{C}}$ of $\mathcal{C}$. That is, if the given coloring $\chi_{\mathcal{C}}$ can be improved within the k-move-neighborhood, then we need to find *any* coloring χ^* improving over $\chi_{\mathcal{C}}$. This coloring χ^* is not required to be in the k-move-neighborhood of $\chi_{\mathcal{C}}$.

Let $I := (G = (V, E), k, \chi_{\mathcal{C}})$ be an instance of LS-Cluster Editing, let M be a cluster modulator of G. Let $\alpha \in \mathbb{N}$ be a color used by $\chi_{\mathcal{C}}$. We say that α is a *modulator color* if $\chi_{\mathcal{C}}^{-1}(\alpha) \cap M \neq \emptyset$. Otherwise, we say that α is a *bag color*. In the remainder of this section, we denote by col_{Mod} and col_{Bag} the set of modulator colors and bag colors of $\chi_{\mathcal{C}}$, respectively. Note that these sets of colors are defined with respect to the initial coloring $\chi_{\mathcal{C}}$ of the LS-Cluster Editing-instance I. Recall that each bag $B \in \mathcal{B}$ is a clique in G and a connected component of $G[V \setminus M]$. Fix an arbitrary ordering of the bags and let B_i denote the ith bag of $\mathcal{B}$ according to this ordering.

We first show that we can improve the initial coloring in polynomial time unless it has some properties that we will exploit in the following.

▶ **Observation 3.10.** *If there is a bag color α such that vertices of two distinct bags receive color α under $\chi_{\mathcal{C}}$, then one can find a coloring χ' of V in polynomial time that improves over $\chi_{\mathcal{C}}$.*

▶ **Observation 3.11.** *If there is a bag $B \in \mathcal{B}$ such that two vertices of B receive distinct bag colors under $\chi_{\mathcal{C}}$, then one can find a coloring χ' of V in polynomial time that improves over $\chi_{\mathcal{C}}$.*

Hence, we assume in the following, that for each bag color $\alpha \in \mathrm{col}_{\mathrm{Bag}}$, $\chi_{\mathcal{C}}^{-1}(\alpha)$ contains only vertices of a single bag and that for each bag $B_i \in \mathcal{B}$, there is at most one bag color $\alpha_i \in \mathrm{col}_{\mathrm{Bag}}$ with $\chi_{\mathcal{C}}^{-1}(\alpha_i) \subseteq B_i$.

Moreover, we assume in the following that $\mathrm{col}_{\mathrm{Mod}}$ has size $\mathcal{O}(k)$. In the final algorithm we use an initial branching – similar to the one used in the algorithm behind Theorem 3.7 – to ensure that this assumption is fulfilled.

Let χ' be a coloring of V and let χ_{int} be the coloring that agrees with $\chi_{\mathcal{C}}$ on all vertices of $V \setminus M$ and that agrees with χ' on all vertices of M. We call χ_{int} the *intermediate coloring* for χ'. The idea behind this definition is the following.

▶ **Observation 3.12.** *Let χ' be a coloring of V. Moreover, let χ_{int} be the intermediate coloring for χ'. It holds that* $\mathrm{move}(\chi_{\mathcal{C}}, \chi_{\mathrm{int}}) + \mathrm{move}(\chi_{\mathrm{int}}, \chi') = \mathrm{move}(\chi_{\mathcal{C}}, \chi')$.

Suppose that there is an improving k-move-neighbor χ' of $\chi_{\mathcal{C}}$. Then, to find a coloring of V that improves over $\chi_{\mathcal{C}}$, it is sufficient to iterate over all colorings χ_{int} with $\mathrm{move}(\chi_{\mathcal{C}}, \chi_{\mathrm{int}}) \leq k$ that agree with $\chi_{\mathcal{C}}$ on all vertices of $V \setminus M$, and to check whether there is a coloring χ' that improves over $\chi_{\mathcal{C}}$ with $\mathrm{move}(\chi_{\mathrm{int}}, \chi') \leq k$ such that χ_{int} and χ' agree on all vertices of M, that is, where χ_{int} is the intermediate coloring for χ'. Unfortunately, such an approach exceeds the desired running time for our algorithm since there are $n^{\mathcal{O}(k)}$ possibilities for the colorings χ_{int} due to the fact that there may be up to $\Theta(n)$ many bag colors and each vertex of M may receive any color. To obtain the desired running time, we instead only iterate over "template colorings". Here, a coloring χ_{temp} of V is a *template coloring* if

- $\mathrm{move}(\chi_{\mathcal{C}}, \chi_{\mathrm{temp}}) \leq k$,
- χ_{temp} agrees with $\chi_{\mathcal{C}}$ on all vertices of $V \setminus M$, and
- no vertex of M receives a color of $\mathrm{col}_{\mathrm{Bag}}$ under χ_{temp}.

For a template coloring χ_{temp}, let $\mathrm{col}_{\mathrm{Move}}$ denote the colors of $\mathbb{N} \setminus (\mathrm{col}_{\mathrm{Mod}} \cup \mathrm{col}_{\mathrm{Bag}})$ that are used by χ_{temp}. We call the colors of $\mathrm{col}_{\mathrm{Move}}$ the *moving colors* of χ_{temp}. Note that only vertices of M may receive a moving color under χ_{temp} and that there are at most k moving colors.

Figure 1 gives an overview over the considered types of colorings and they way we use them to find a coloring that improves over $\chi_{\mathcal{C}}$.

The idea behind template colorings is that a template coloring χ_{temp} may represent intermediate colorings for many colorings χ' in the following way: For a coloring χ' of V, we say that a template coloring χ_{temp} is *quasi-intermediate for* χ' if there is a "template recolor-function" $f\colon \mathbb{N} \to \mathbb{N}$ such that $f \circ \chi_{\mathrm{temp}}$ is the intermediate coloring for χ'. Herein, a function $f\colon \mathbb{N} \to \mathbb{N}$ is a *template recolor-function* if f preserves identity on all colors of $\mathbb{N} \setminus \mathrm{col}_{\mathrm{Move}}$ and where $f|_{\mathrm{col}_{\mathrm{Move}}}$ maps each color of $\mathrm{col}_{\mathrm{Move}}$ to some color of $\mathbb{N} \setminus \mathrm{col}_{\mathrm{Mod}}$ injectively. Essentially, this means that each of the moving colors of χ_{temp} may be identified with any bag color. Informally, this is due to the fact that each vertex that receives a moving

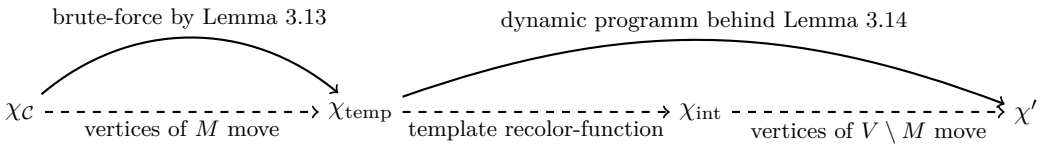

Figure 1 An overview over the four different kinds of considered colorings. For the initial coloring $\chi_{\mathcal{C}}$ and an improving coloring χ' with $\text{move}(\chi_{\mathcal{C}}, \chi') \leq k$, there is a template coloring χ_{temp} and an intermediate coloring χ_{int} such that there is a template recolor-function between χ_{temp} and χ_{int}. To find a coloring at least as good as χ', we first brute-force all possible choices of the template coloring χ_{temp} and afterwards search for the best coloring for which χ_{temp} is quasi-intermediate. This is done by a dynamic program that simultaneously finds the best template recolor-function and the best way to distribute the bag vertices by using at most k moves.

color under χ_{temp} already changed its color and we may move all vertices of that moving color together to any bag color while preserving the move-distance to $\chi_{\mathcal{C}}$. Note that, for each coloring χ' of V with $\text{move}(\chi_{\mathcal{C}}, \chi') \leq k$, there is a template coloring χ_{temp} which is quasi-intermediate for χ'. In contrast to intermediate colorings, we can show that we can enumerate a maximal set $\mathcal{X}$ of pairwise non-isomorphic template colorings in the desired running time.

▶ **Lemma 3.13.** *One can compute a maximal set $\mathcal{X}$ of pairwise non-isomorphic template colorings in time $(|\text{col}_{\text{Mod}}| + k)^k \cdot |M|^k \cdot n^{\mathcal{O}(1)}$.*

Proof. Recall that for each template coloring χ_{temp}, $\text{move}(\chi_{\mathcal{C}}, \chi_{\text{temp}}) \leq k$. Moreover, by the definition of a template coloring, $\text{Move}(\chi_{\mathcal{C}}, \chi_{\text{temp}}) \subseteq M$. Hence, to obtain a maximal set $\mathcal{X}$ of pairwise non-isomorphic template colorings, consider all possible subsets of M of size at most k and consider all possible ways of assigning colors of $\text{col}_{\text{Mod}} \cup X$ to these at most k vertices, where X is an arbitrary set of k colors from $\mathbb{N} \setminus (\text{col}_{\text{Mod}} \cup \text{col}_{\text{Bag}})$. Note that this can be done in the stated running time. ◀

Lemma 3.13 implies that, in order to find a coloring χ^* of V that improves over $\chi_{\mathcal{C}}$ (provided that there is such a coloring in the k-move-neighborhood of $\chi_{\mathcal{C}}$), it suffices to do the following: For each template coloring χ_{temp} in a maximal set of pairwise non-isomorphic template colorings, find the best coloring χ' in the k-move-neighborhood of $\chi_{\mathcal{C}}$ such that χ_{temp} is quasi-intermediate for χ'.

The latter task can be solved in $2^{\mathcal{O}(|\text{col}_{\text{Mod}}|+k)} \cdot n^{\mathcal{O}(1)}$ time.

▶ **Lemma 3.14 ($\star$).** *Let χ_{temp} be a template coloring. In $2^{\mathcal{O}(|\text{col}_{\text{Mod}}|+k)} \cdot n^{\mathcal{O}(1)}$ time, one can find a coloring of V that improves over $\chi_{\mathcal{C}}$ or correctly output that the k-move-neighborhood of $\chi_{\mathcal{C}}$ does not contain a coloring χ' such that χ' improves over $\chi_{\mathcal{C}}$ and where χ_{temp} is quasi-intermediate for χ'.*

We now conclude our permissive algorithm for LS-Cluster Editing. As we can switch between clusterings and cluster colorings in polynomial time, this also proves Theorem 3.9.

▶ **Theorem 3.15.** *Let $G = (V, E)$ be a graph, let $\chi_{\mathcal{C}}$ be a coloring of V, and let $k \in \mathbb{N}$. In $2^{\mathcal{O}(k)} \cdot k^k \cdot \text{cvd}^{3k} \cdot n^{\mathcal{O}(1)}$ time, one can find a coloring χ^* that improves over χ or correctly output that there is no coloring in the k-move-neighborhood of $\chi_{\mathcal{C}}$ that improves over $\chi_{\mathcal{C}}$.*

Proof. First, we 2-approximate a cluster modulator M for G in polynomial time [1]. Let $\mathcal{B}$ be the bags of $G[V \setminus M]$, let $\mathrm{col}_{\mathrm{Mod}}$ and $\mathrm{col}_{\mathrm{Bag}}$ be the modulator colors and bag colors of $\chi_{\mathcal{C}}$, respectively. If the condition of Observation 3.10 or Observation 3.11 applies, then we find a coloring χ^* of V that improves over χ in polynomial time. Hence, assume in the following that this is not the case.

As mentioned at the beginning of this subsection, we perform an initial branching step to ensure that the coloring $\chi_{\mathcal{C}}$ uses at most $2k$ modulator colors. Let χ' be the best coloring in the k-move-neighborhood of $\chi_{\mathcal{C}}$. Assume that χ' improves over $\chi_{\mathcal{C}}$. Since $\mathrm{move}(\chi_{\mathcal{C}}, \chi') \leq k$, Observation 2.1 implies that there is a subset $S \subseteq \mathrm{col}_{\mathrm{Mod}}$ of size at least $|\mathrm{col}_{\mathrm{Mod}}|-2k$ such that for each modulator color $\alpha \in S$, $\chi_{\mathcal{C}}^{-1}(\alpha) = {\chi'}^{-1}(\alpha)$. Hence, to find a coloring that improves over $\chi_{\mathcal{C}}$ it is sufficient to branch into all subsets $S \subseteq \mathrm{col}_{\mathrm{Mod}}$ of size at least $|\mathrm{col}_{\mathrm{Mod}}| - 2k$ and ask for a coloring $\widehat{\chi}'$ of $\widehat{V} := V \setminus (\cup_{\alpha \in S} \chi_{\mathcal{C}}^{-1}(\alpha))$ that improves over $\widehat{\chi} := \chi_{\mathcal{C}}|_{\widehat{V}}$ with respect to the subgraph $G[\widehat{V}]$. Note that this branching takes $|\mathrm{col}_{\mathrm{Mod}}|^{2k} \cdot n^{\mathcal{O}(1)} \leq |M|^{2k} \cdot n^{\mathcal{O}(1)}$ time.

Hence, in the following, we assume that $\mathrm{col}_{\mathrm{Mod}}$ has size at most $2k$. Next, we iterate over a maximal set $\mathcal{X}$ of pairwise non-isomorphic template colorings and compute for each such template coloring χ_{temp} a coloring χ^* which is at least as good as a best coloring χ'' in the k-move-neighborhood of $\chi_{\mathcal{C}}$ where χ_{temp} is quasi-intermediate for χ''. The latter task can be done in $2^{\mathcal{O}(|\mathrm{col}_{\mathrm{Mod}}|+k)} \cdot n^{\mathcal{O}(1)}$ time due to Lemma 3.14 for each template coloring $\chi_{\mathrm{temp}} \in \mathcal{X}$. Due to Lemma 3.13, $\mathcal{X}$ can be computed in $(|\mathrm{col}_{\mathrm{Mod}}| + k)^k \cdot |M|^k \cdot n^{\mathcal{O}(1)}$ time and has size $(|\mathrm{col}_{\mathrm{Mod}}| + k)^k \cdot |M|^k \cdot n^{\mathcal{O}(1)}$.

This algorithm is correct, since there is a template coloring χ'_{temp} such that χ'_{temp} is quasi-intermediate for χ'. Thus, in this way, we will find a coloring at least as good as χ'.

The whole algorithm runs in $|M|^{2k} \cdot (|\mathrm{col}_{\mathrm{Mod}}| + k)^k \cdot |M|^k \cdot 2^{\mathcal{O}(|\mathrm{col}_{\mathrm{Mod}}|+k)} \cdot n^{\mathcal{O}(1)}$ time. Since we ensured with the initial branching, that $\mathrm{col}_{\mathrm{Mod}}$ has size at most $2k$, this results in a running time of $2^{\mathcal{O}(k)} \cdot k^k \cdot |M|^{3k} \cdot n^{\mathcal{O}(1)}$ time. Finally, since M is a 2-approximated cluster modulator, $|M| \leq 2 \cdot \mathrm{cvd}(G)$. Hence, we obtain the stated running time of $2^{\mathcal{O}(k)} \cdot k^k \cdot \mathrm{cvd}(G)^{3k} \cdot n^{\mathcal{O}(1)}$ time. ◀

4 Lower Bounds

In this section we present several hardness results for LS-Cluster Deletion and LS-Cluster Editing. We obtain our hardness results for LS-Cluster Editing by reductions from restricted instances of Densest-k-Subgraph, which is defined as follows

Densest-k-Subgraph
Input: A graph $G = (V, E)$, integers k and d.
Question: Is there a subset $S \subseteq V$ of size exactly k such that $|E(S)| \geq \binom{k}{2} - d$?

Hence, we first show that Densest-k-Subgraph provides these hardness results on the desired restricted instances.

▶ **Theorem 4.1** (⋆). *Even if $d = \frac{k-1}{2}$, Densest-k-Subgraph is* W[1]*-hard when parameterized by both k and the degeneracy of G and cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f, unless the ETH fails. This holds even on instances where $|E(S)| < \binom{k-1}{2} - \frac{k-1}{4}$ for each vertex set S of size $k-1$.*

Based on these hardness results for Densest-k-Subgraph, we are now able to analyze the parameterized complexity of LS-Cluster Editing for the parameter combination of k plus the size of the largest cluster of the initial clustering $\mathcal{C}$.

▶ **Theorem 4.2.** *LS-Cluster Editing is* W[1]*-hard when parameterized by* $k + \ell +$ degen*, where* $\ell := \max_{C \in \mathcal{C}} |C|$ *and* degen *denotes the degeneracy of* G*. Moreover, unless the ETH fails, there is no computable function* f *such that LS-Cluster Editing can be solved in* $f(k+\ell) \cdot n^{o(k+\ell)}$ *time.*

Proof. We present a parameterized reduction from Densest-k-Subgraph with the restrictions listed in Theorem 4.1. Let $I = (G = (V, E), k, d)$ be an instance of Densest-k-Subgraph with $d = \frac{k-1}{2}$ such that $|E(S)| < \binom{k-1}{2} - \frac{k-1}{4}$ for each vertex set S of size $k-1$. We define an instance $I' := (G' := (V', E'), k, \mathcal{C})$ of LS-Cluster Editing with $\max_{C \in \mathcal{C}} |C| \in \mathcal{O}(k)$ as follows: We initialize G' as G and add for each vertex $v \in V$ a set K_v of $7k + \frac{k-3}{2}$ vertices to G' such that $\{v\} \cup K_v$ is a clique in G'. Additionally, we add a clique K^* of size $7k$ to G' and add edges to G', such that each vertex of V is adjacent to each vertex of K^*. Finally, we set $\mathcal{C} := \{K^*\} \cup \{\{v\} \cup K_v \mid v \in V\}$. Note that by definition of $\mathcal{C}$, each cluster $C \in \mathcal{C}$ is a clique in G'. The correctness proof is based on the following claim.

▷ Claim 4.3. Let $\mathcal{C}' := \{K^* \cup S\} \cup \{\{K_v\} \mid v \in S\} \cup \{\{v\} \cup K_v \mid v \in V \setminus S\}$ be a clustering of G' for some vertex set $S \subseteq V$. The improvement of $\mathcal{C}'$ over $\mathcal{C}$ is $2 \cdot |E_G(S)| - \binom{|S|}{2} - |S| \cdot \frac{k-3}{2}$.

Proof. We only have to consider the edges incident with at least one vertex of S in $E(\mathcal{C})$ and $E(\mathcal{C}')$. Let $F := E(\mathcal{C})$ and let $F' := E(\mathcal{C}')$. Note that the symmetric difference between F and F' are the edges of $F \oplus F' = \{\{v, x\} \mid v \in S, x \in K_v \cup K^*\} \cup \binom{S}{2}$. More precisely, $F \setminus F' = \{\{v, x\} \mid v \in S, x \in K_v\}$ and $F' \setminus F = \{\{v, x\} \mid v \in S, x \in K^*\} \cup \binom{S}{2}$. By construction, all edges of $F \oplus F'$ exist in G', except for the edges of $\binom{S}{2} \setminus E_{G'}(S) = \binom{S}{2} \setminus E_G(S)$. Hence, the improvement of $\mathcal{C}'$ over $\mathcal{C}$ is

$$\sum_{v \in S} (|K^*| - |K_v|) + |E_{G'}(S)| - \left(\binom{|S|}{2} - |E_{G'}(S)|\right) = 2 \cdot |E_G(S)| - \binom{|S|}{2} - |S| \cdot \frac{k-3}{2}. \qquad \triangleleft$$

Next, we show that I is a yes-instance of Densest-k-Subgraph if and only if I' is a yes-instance of LS-Cluster Editing.

($\Rightarrow$) Let S be a set of size k in G such that $|E_G(S)| \geq \binom{k}{2} - d$. We set $\mathcal{C}' := \{K^* \cup S\} \cup \{\{K_v\} \mid v \in S\} \cup \{\{v\} \cup K_v \mid v \in V \setminus S\}$. Note that $\text{move}(\mathcal{C}, \mathcal{C}') = k$. We show that $\mathcal{C}'$ is improving over $\mathcal{C}$. Due to Claim 4.3 and since $|E_G(S)| \geq \binom{k}{2} - d$ and $d = \frac{k-1}{2}$, the improvement of $\mathcal{C}'$ over $\mathcal{C}$ is at least

$$\binom{k}{2} - 2d - k \cdot \frac{k-3}{2} = \binom{k}{2} - k + 1 - k \cdot \frac{k-3}{2} = \binom{k}{2} - k \cdot \frac{k-1}{2} + 1 = 1.$$

Hence, I' is a yes-instance of LS-Cluster Editing.

($\Leftarrow$) Suppose that I' is a yes-instance of LS-Cluster Editing. Let $\mathcal{C}'$ be the best clustering for G' with $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$. Since I' is a yes-instance of LS-Cluster Editing, $\mathcal{C}' \neq \mathcal{C}$. We make some observations about the potential moves between $\mathcal{C}$ and $\mathcal{C}'$. The goal is to show that there is a set $S \subseteq V$ of size at most k such that $\mathcal{C}' = \{K^* \cup S\} \cup \{K_v \mid v \in S\} \cup \{\{v\} \cup K_v \mid v \in V \setminus S\}$. To this end, we show some intermediate results.

First, we show that for each vertex $v \in V$ there is a cluster $C \in \mathcal{C}'$ with $K_v \subseteq C$. Suppose that this is not the case. Hence, there is a vertex $v \in V$ and at least two clusters C_1 and C_2 in $\mathcal{C}'$ containing vertices of K_v. Since K_v has size more than k, at least one vertex of K_v is not moved. Assume without loss of generality that C_1 contains this vertex. Hence, each vertex of $C_2 \cap K_v$ moved to C_2. Thus, $C_2 \cap K_v$ contains at most k vertices. Let x be an arbitrary vertex of $C_2 \cap K_v$. Since K_v has size more than $4k$, C_1 contains at least $3k$ vertices of K_v and at most k vertices of $V' \setminus K_v$. By definition of K_v, the closed neighborhood of x is

exactly $K_v \cup \{v\}$. Hence, x has at most k neighbors in C_2, at least $3k$ neighbors in C_1 and at most k non-neighbors in C_1. Consequently, not moving x to C_2 yields a better clustering. Since $\mathcal{C}'$ is the best clustering with $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$, this is not possible.

Next, we show that there is a cluster C in $\mathcal{C}'$ with $K^* \subseteq C$. Suppose that this is not the case. Hence, there are at least two clusters C_1 and C_2 in $\mathcal{C}'$ containing vertices of K^*. Since K^* has size more than k, at least one vertex of K^* is not moved. Assume without loss of generality that C_1 contains this vertex. Hence, each vertex of $C_2 \cap K^*$ moved to C_2. Thus, $C_2 \cap K^*$ contains at most k vertices. Let x be an arbitrary vertex of $C_2 \cap K^*$. Since K^* has size more than $4k$, C_1 contains at least $3k$ vertices of K^* and at most k vertices of $V' \setminus K^*$. By definition of K^*, the closed neighborhood of x is exactly $K^* \cup V$. Hence, since each cluster in $\mathcal{C}$ contains at most one vertex of V, x has at most $k+1$ neighbors in C_2, at least $3k$ neighbors in C_1 and at most k non-neighbors in C_1. Consequently, not moving x to C_2 yields a better clustering. Since $\mathcal{C}'$ is the best clustering with $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$, this is not possible.

The above implies that only vertices of V moved to obtain $\mathcal{C}'$ from $\mathcal{C}$.

Next, we show that for each vertex $v \in V$, the cluster C of $\mathcal{C}'$ that contains v either contains all vertices of K^* or all vertices of K_v. Suppose that this is not the case and let v be a vertex of V such that the cluster $C \in \mathcal{C}'$ with $v \in C$ is not a superset of K^* and not a superset of K_v. Since v is only adjacent to at most one vertex in each cluster of $\mathcal{C} \setminus \{K^*, K_v \cup \{v\}\}$, v has at most k neighbors in C. Let K'_v be the cluster of $\mathcal{C}'$ containing all vertices of K_v. Since K_v has size more than $3k$, K'_v contains at most k non-neighbors of v and at least $3k$ neighbors of v. Hence, not moving v from K'_v to C yields a better clustering. Since $\mathcal{C}'$ is the best clustering with $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$, this is not possible.

Concluding, there is a nonempty vertex set S of size at most k such that $\mathcal{C}' = \{K^* \cup S\} \cup \{K_v \mid v \in S\} \cup \{\{v\} \cup K_v \mid v \in V \setminus S\}$. More precisely, the vertices of S are exactly the vertices that are moved to obtain $\mathcal{C}'$ from $\mathcal{C}$.

It remains to show that S has size k and that $E_G(S) = E_{G'}(S)$ contains at least $\binom{k}{2} - d$ edges. Due to Claim 4.3 and since $\mathcal{C}'$ is improving over $\mathcal{C}$, $2 \cdot |E_G(S)| - \binom{|S|}{2} - |S| \cdot \frac{k-3}{2} \geq 1$.

▷ Claim 4.4. S has size k.

Proof. If $|S| < k-1$, then $2|E_G(S)| \leq 2 \cdot \binom{|S|}{2} < |S| \cdot \frac{k-3}{2} + \binom{|S|}{2} + 1$. Since $2|E_G(S)| \geq |S| \cdot \frac{k-3}{2} + \binom{|S|}{2} + 1$, we conclude $|S| \geq k-1$.

Assume towards a contradiction that S has size exactly $k-1$. By assumption, $|E_G(S)| < \binom{k-1}{2} - \frac{k-1}{4} = (k-1) \cdot (\frac{k-2}{2} - \frac{1}{4}) = (k-1) \cdot \frac{2k-5}{4}$. Hence, $(k-1) \cdot \frac{2k-5}{2} > 2 \cdot |E_S(G)| \geq |S| \cdot \frac{k-3}{2} + \binom{|S|}{2} + 1 = (k-1) \cdot \frac{k-3}{2} + \binom{(k-1)}{2} + 1 = (k-1) \cdot \frac{2k-5}{2} + 1$, a contradiction.

Consequently, S contains exactly k vertices since to obtain $\mathcal{C}'$ from $\mathcal{C}$, each vertex of S moved and $\text{move}(\mathcal{C}, \mathcal{C}') \leq k$. ◁

It remains to show that $|E_G(S)| \geq \binom{k}{2} - d$. By Claim 4.3, $2|E_G(S)| \geq |S| \cdot \frac{k-3}{2} + \binom{|S|}{2} + 1 = \frac{2k^2-4k}{2} + 1 = k^2 - k - (k-1)$. Thus $|E_G(S)| \geq \frac{k^2-k}{2} - \frac{k-1}{2} = \binom{k}{2} - d$. Hence, I is a yes-instance of Densest-k-Subgraph.

Parameter bounds. Recall that the size $\ell := \max_{C \in \mathcal{C}} |C|$ of the largest cluster in $\mathcal{C}$ is $\mathcal{O}(k)$.

Due to Theorem 4.1, Densest-k-Subgraph cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f, unless the ETH fails. This implies that LS-Cluster Deletion cannot be solved in $f(k+\ell) \cdot |V'|^{o(k+\ell)}$ time for any computable function f, unless the ETH fails, since $|V'| \in n^{\mathcal{O}(1)}$.

Next, we analyze the degeneracy degen' of G'. Let degen denote the degeneracy of G. Since each vertex of K_v for some vertex $v \in V$ has only neighbors in $K_v \cup \{v\}$, each such vertex

has degree $\mathcal{O}(k)$ in G'. Furthermore, each vertex of V has only $|K_v| + |K^*| \in \mathcal{O}(k)$ additional neighbors in G'. Hence, the degeneracy of G' is $\mathcal{O}(k + \text{degen})$. Consequently, LS-Cluster Editing is W[1]-hard when parameterized by $k + \ell + \text{degen}'$, since due to Theorem 4.1, Densest-k-Subgraph is W[1]-hard when parameterized by $k + \text{degen}$. ◀

Next, we show that even when the initial clustering consists only of two clusters, LS-Cluster Editing remains W[1]-hard when parameterized by k.

▶ **Theorem 4.5** (⋆). *Even when the initial clustering consists of only two clusters, LS-Cluster Editing is* W[1]*-hard when parameterized by k and cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f, unless the ETH fails.*

Hence, LS-Cluster Editing is W[1]-hard when parameterized by the arguably most natural parameter combinations $k + \max_{C \in \mathcal{C}} |C|$ and $k + |\mathcal{C}|$.

Finally, we present our hardness results for LS-Cluster Deletion. As we show, these hardness results also hold for the permissive version of LS-Cluster Deletion.

▶ **Theorem 4.6.** *LS-Cluster Deletion*

- *is* W[1]*-hard when parameterized by $k + \ell$, where $\ell := \max_{C \in \mathcal{C}} |C|$ denotes the size of the largest cluster in $\mathcal{C}$,*
- *cannot be solved in $f(k + \ell) \cdot n^{o(k+\ell)}$ time for any computable function f, unless the ETH fails, and*
- *does not admit a polynomial kernel when parameterized by $k + \text{vc}$, unless* NP $\subseteq$ coNP/poly*, where* vc *denotes the vertex cover number of G.*

All of this holds even if there is an optimal clustering $\mathcal{C}^*$ with $E(\mathcal{C}^*) \subseteq E$ in the k-move-neighborhood of $\mathcal{C}$.

Proof. We present a polynomial time reduction from Multicolored Clique, which is W[1]-hard when parameterized by the size of the sought clique [14].

Multicolored Clique
Input: A graph $G = (V, E)$ and an integer k such that G is k-partite.
Question: Is there a clique of size k in G?

Let $I := (G = (V, E), k)$ be an instance of Multicolored Clique and let V_k be the largest part of the k-partition $(V_1, \ldots, V_k)$ of G. We define an instance $I' := (G' := (V', E'), k', \mathcal{C})$ of LS-Cluster Deletion as follows: Initialize G' as a copy of G. Then for each $i \in [1, k-1]$, do the following:

- add a vertex x_i to G' and
- for each vertex $v \in V_i$, add a vertex set K_v of size $z := 2k$ to G' and turn $K_v \cup \{v\}$ and $K_v \cup \{x_i\}$ into cliques in G'.

Let $X := \{x_i \mid 1 \leq i \leq k-1\}$. Finally, we add two additional adjacent vertices a and b to G' and make $X \cup \{a\}$ a clique in G'. This completes the construction of G'. We complete the construction of I' by setting $k' := 2k - 1$ and

$$\mathcal{C} := \{X \cup \{a\}, \{b\}\} \cup \bigcup_{v \in V \setminus V_k} \{K_v \cup \{v\}\} \cup \bigcup_{v \in V_k} \{\{v\}\}.$$

Note that by construction $E(\mathcal{C}) \subseteq E'$ and $|E(\mathcal{C})| = |V \setminus V_k| \cdot \binom{z+1}{2} + \binom{k}{2}$.

Next, we show that there is a clique of size k in G if and only if there is a clustering $\mathcal{C}'$ for G' that improves over $\mathcal{C}$. We further show that each clustering $\mathcal{C}'$ for G' that improves over $\mathcal{C}$ is a k'-move-neighbor of $\mathcal{C}$.

($\Rightarrow$) Let S be a clique of size k in G. Since $(V_1, \ldots, V_k)$ is a k-partition of G, for each $i \in [1,k]$, S contains exactly one vertex of V_i. For each $i \in [1,k]$, let v_i be that unique vertex of $V_i \cap S$. We set

$$\mathcal{C}' := \{S, \{a,b\}\} \cup \bigcup_{i\in[1,k-1]} \{K_{v_i} \cup \{x_i\}\} \cup \bigcup_{v\in V\setminus(V_k\cup S)} \{K_v \cup \{v\}\} \cup \bigcup_{v\in V_k\setminus\{v_k\}} \{\{v\}\}.$$

Note that by construction, $E(\mathcal{C}') \subseteq E'$ and by definition of $\mathcal{C}'$, $|E(\mathcal{C}')| = \binom{k}{2} + |X| \cdot \binom{z+1}{2} + |V \setminus (V_k \cup S)| \cdot \binom{z+1}{2} + 1 = |V \setminus V_k| \cdot \binom{z+1}{2} + \binom{k}{2} + 1 = |E(\mathcal{C})| + 1$. Hence, $\mathcal{C}'$ is improving over $\mathcal{C}$. Moreover, by construction of $\mathcal{C}'$, $\mathcal{C}'$ and $\mathcal{C}$ are k'-move-neighbors.

($\Leftarrow$) Let $\mathcal{C}'$ be a best clustering for G' with $E(\mathcal{C}') \subseteq E'$ and suppose that $\mathcal{C}'$ improves over $\mathcal{C}$. Before we show that there is a clique of size k in G, we prove some properties of the clustering $\mathcal{C}'$.

First, we show that for each vertex $v \in V \setminus V_k$, there is a cluster C'_v in $\mathcal{C}'$ with $K_v \subseteq C'_v$. Suppose that there are at least two clusters C'_v and C''_v in $\mathcal{C}'$ with $K_v \cap C'_v \neq \emptyset$ and $K_v \cap C''_v \neq \emptyset$ and suppose that $|C'_v| \geq |C''_v|$. Let v' be an arbitrary vertex of $C''_v \cap K_v$. Recall that by construction of G', v' has the same closed neighborhood as any other vertex of K_v. Since $E(\mathcal{C}') \subseteq E'$, each vertex of C'_v is part of the closed neighborhood of v'. Hence, the clustering $\mathcal{C}'' := (\mathcal{C} \setminus \{C'_v, C''_v\}) \cup \{C'_v \cup \{v'\}, C''_v \setminus \{v'\}\}$ fulfills $E(\mathcal{C}'') \subseteq E'$ and improves over $\mathcal{C}'$. Since $\mathcal{C}'$ is a best clustering for G' with $E(\mathcal{C}') \subseteq E'$, no such two clusters exist and thus for each vertex $v \in V \setminus V_k$, there is a cluster C'_v in $\mathcal{C}'$ with $K_v \subseteq C'_v$.

Next, we show that for each $i \in [1, k-1]$ and each vertex $v \in V_i$, the cluster C'_v is either $K_v \cup \{v\}$ or $K_v \cup \{x_i\}$. Suppose that this is not the case and let $i \in [1, k-1]$ and let v be a vertex of V_i such that $C'_v \notin \{K_v \cup \{v\}, K_v \cup \{x_i\}\}$. Note that this implies that $C'_v = K_v$. Furthermore, let C'' be the cluster of $\mathcal{C}'$ that contains v. Since $E(\mathcal{C}') \subseteq E'$ and v has only neighbors in $V \cup K_v$, the cluster C'' is a clique in G. Moreover, since G is k-partite, this implies that C'' has size at most k. Hence, the clustering $\mathcal{C}'' := (\mathcal{C} \setminus \{C'_v, C''\}) \cup \{K_v \cup \{v\}, C'' \setminus \{v\}\}$ fulfills $E(\mathcal{C}'') \subseteq E'$ and improves over $\mathcal{C}'$. Since $\mathcal{C}'$ is a best clustering for G' with $E(\mathcal{C}') \subseteq E'$, this implies that for each $i \in [1, k-1]$ and each vertex $v \in V_i$, the cluster C'_v is either $K_v \cup \{v\}$ or $K_v \cup \{x_i\}$.

Note that this implies that for each $i \in [1, k-1]$, there is at most one vertex $v_i \in V_i$ for which $C'_{v_i} \neq K_v \cup \{v_i\}$. Intuitively, if such a vertex v_i moves out of its cluster from $\mathcal{C}$, then the vertex x_i has to move into the original cluster of v_i.

Let $S' \subseteq V'$ be a minimal set of vertices that have to move to obtain $\mathcal{C}'$ from $\mathcal{C}$. Moreover, let $S := S' \cap (V \setminus V_k)$. By the above, for each $i \in [1, k-1]$, S contains at most one vertex of V_i. Recall that each vertex of V_k has neighbors only in $V \setminus V_k$ and can thus only be in a cluster of $\mathcal{C}'$ with a (potentially empty) subset of vertices of S. Hence, S' contains no vertex of V_k. In the following, we show that there is a vertex $v_k \in V_k$ such that $S \cup \{v_k\}$ is a clique of size k in G.

To this end, we analyze the number of edges in the clusters $\mathcal{C}_S \subseteq \mathcal{C}'$ that contain vertices of $S \cup V_k$ and have size at least two and the number of edges in the clusters $\mathcal{C}_X \subseteq \mathcal{C}'$ that have size at least two and contain only vertices of $X \cup \{a,b\}$. By the above, each cluster C of $\mathcal{C}_S$ contains only vertices of $S \cup V_k$ and C contains at most one vertex of V_k, since V_k is an independent set in G'. Note that this implies that each cluster of $\mathcal{C}_S$ contains at least one vertex of S. Furthermore, note that $E(\mathcal{C}') = \bigcup_{v\in V\setminus V_k} \binom{C'_v}{2} \cup E(\mathcal{C}_S) \cup E(\mathcal{C}_X)$. Since for each vertex $v \in V \setminus V_k$, C'_v has size $z+1$, $|E(\mathcal{C}')| = |V \setminus V_k| \cdot \binom{z+1}{2} + |E(\mathcal{C}_S)| + |E(\mathcal{C}_X)|$. Moreover, since $\mathcal{C}'$ is improving over $\mathcal{C}$, $|E(\mathcal{C}')| - |E(\mathcal{C})| = |E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| - \binom{k}{2} \geq 1$. In the following, we show that $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| \leq \binom{k}{2} + 1$ and that $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| \leq \binom{k}{2}$ if there is no vertex $v_k \in V_k$ such that $S \cup \{v_k\}$ is a clique of size k in G. To this end, we

analyze the size of $E(\mathcal{C}_S)$ and the size of $E(\mathcal{C}_X)$ separately.

First, we show that if $\mathcal{C}_S$ has size at least two, then $|E(\mathcal{C}_S)| < \binom{|S|+1}{2}$. This follows inductively by the fact that each cluster of $\mathcal{C}_S$ has size at least two and contains at most one vertex of V_k, and for each $i \geq 2$ and each $j \geq 2$, $\binom{i}{2} + \binom{j}{2} < \binom{i+j-1}{2}$. Hence, $E(\mathcal{C}_S)$ has size at least $\binom{|S|+1}{2}$ if and only if there is a vertex $v_k \in V_k$ such that $\mathcal{C}_S = \{S \cup \{v_k\}\}$. Moreover, this implies that $|E(\mathcal{C}_S)| \leq \binom{|S|+1}{2}$.

Second, we analyze the size of $E(\mathcal{C}_X)$. Since for each $i \in [1, k-1]$, if there is a vertex $v_i \in V_i \cap S$, then the vertex x_i moves to the cluster containing all vertices of K_{v_i}, that is, x_i is not contained in any cluster of $\mathcal{C}_X$. Hence, at most $k-1-|S|$ vertices of X are contained in clusters of $\mathcal{C}_X$. Since b is adjacent only to a, this implies that $|E(\mathcal{C}_X)| \leq \binom{k-|S|}{2} + 1$.

Finally, we show that for some vertex $v_k \in V_k$, $S \cup \{v_k\}$ is a clique of size k in G. Assume that $|S| < k-1$. Hence, by the above $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| \leq \binom{|S|+1}{2} + \binom{k-|S|}{2} + 1 < \binom{k}{2} + 1$ and thus $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| \leq \binom{k}{2}$. Since $|E(\mathcal{C}')| > |E(\mathcal{C})|$, this is not possible.

Consequently, $|S| = k-1$. Hence, by the above $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| \leq \binom{|S|+1}{2} + \binom{k-|S|}{2} + 1 = \binom{k}{2} + 1$. Hence, $|E(\mathcal{C}')| \leq |E(\mathcal{C})| + 1$. Since $\mathcal{C}'$ improves over $\mathcal{C}$, $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| = \binom{k}{2} + 1$. As shown before, $|E(\mathcal{C}_S)| + |E(\mathcal{C}_X)| = \binom{k}{2} + 1$ only holds if $\mathcal{C}_S$ consists of a single cluster $C' := S \cup \{v_k\}$ for some vertex $v_k \in V_k$. Hence, there is a clique of size k in G and I is a yes-instance of Multicolored Clique.

Moreover, note that this implies that $\mathcal{C}'$ is a k'-move-neighbor of $\mathcal{C}$, since only the $k-1$ vertices of S, the $k-1$ vertices of X, and either a or b changed their cluster.

Parameter bounds. Let $\ell := \max_{C \in \mathcal{C}} |C|$. Recall that since $z = 2k$, $\ell = 2k+1$. Since Multicolored Clique is W[1]-hard when parameterized by k and cannot be solved in $f(k) \cdot n^{o(k)}$ time for any computable function f, unless the ETH fails [14], this implies that LS-Cluster Deletion is W[1]-hard when parameterized by $k' + \ell$ and cannot be solved in $f(k'+\ell) \cdot |V'|^{o(k'+\ell)}$ time for any computable function f, unless the ETH fails, since $|V'| \in n^{\mathcal{O}(k)}$. Moreover, note that $V' \setminus V_k$ is a vertex cover of G' of size $|V \setminus V_k| \cdot (2k+1) + k + 1$. Since Multicolored Clique does not admit a polynomial kernel when parameterized by $k + |V \setminus V_k|$, unless NP $\subseteq$ coNP/poly [24], this implies that LS-Cluster Deletion does not admit a polynomial kernel when parameterized by $k'+\mathrm{vc}(G')$ unless NP $\subseteq$ coNP/poly. ◀

Note that due to the last restriction, the permissive version of LS-Cluster Deletion shares the same W[1]-hardness and the same ETH-based lower bound, meaning that also for permissive LS-Cluster Deletion, the algorithms in Section 3 are essentially optimal.

5 Conclusion

We analyzed the parameterized complexity for LS-Cluster Editing and LS-Cluster Deletion, leaving some open questions for future work: First, what is the complexity of LS-Cluster Deletion with respect to the combined parameter number $|\mathcal{C}|$ of clusters and k? Second, can we show lower bounds for the permissive variant of LS-Cluster Editing? Finally, can some of the algorithmic ideas of our work be used to improve the local-search based heuristics for Cluster Deletion or Cluster Editing?

References

1 Manuel Aprile, Matthew Drescher, Samuel Fiorini, and Tony Huynh. A tight approximation algorithm for the cluster vertex deletion problem. *Math. Program.*, 197(2):1069–1091, 2023. doi:10.1007/s10107-021-01744-w.

2 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004. doi:10.1023/B:MACH.0000033116.57574.95.

3 Valentin Bartier, Gabriel Bathie, Nicolas Bousquet, Marc Heinrich, Théo Pierron, and Ulysse Prieto. PACE solver description: μsolver - heuristic track. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 33:1–33:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.33.

4 Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3-4):281–297, 1999. doi:10.1089/106652799318274.

5 Daniel Berend and Tamir Tassa. Improved bounds on bell numbers and on moments of sums of random variables. *Probability and Mathematical Statistics*, 30(2):185–205, 2010.

6 René van Bevern, Vincent Froese, and Christian Komusiewicz. Parameterizing edge modification problems above lower bounds. *Theory of Computing Systems*, 62(3):739–770, 2018. doi:10.1007/s00224-016-9746-5.

7 Thomas Bläsius, Philipp Fischbeck, Lars Gottesbüren, Michael Hamann, Tobias Heuer, Jonas Spinner, Christopher Weyand, and Marcus Wilhelm. PACE solver description: Kapoce: A heuristic cluster editing algorithm. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 31:1–31:4. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.31.

8 Sebastian Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 16:79–89, 2012. doi:10.1016/j.jda.2012.04.005.

9 Édouard Bonnet, Yoichi Iwata, Bart M. P. Jansen, and Lukasz Kowalik. Fine-grained complexity of k-OPT in bounded-degree graphs for solving TSP. In *Proceedings of the 27th Annual European Symposium on Algorithms (ESA '19)*, volume 144 of *LIPIcs*, pages 23:1–23:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

10 Yixin Cao and Jianer Chen. Cluster Editing: Kernelization based on edge cuts. *Algorithmica*, 64(1):152–169, 2012.

11 Yixin Cao and Yuping Ke. Improved Kernels for Edge Modification Problems. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:14, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.IPEC.2021.13.

12 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In *Proceedings of the 47th Annual ACM Symposium on Theory of Computing (STOC '15)*, pages 219–228. ACM, 2015. doi:10.1145/2746539.2746604.

13 Jianer Chen and Jie Meng. A $2k$ kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.

14 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

15 Martin Dörnfelder, Jiong Guo, Christian Komusiewicz, and Mathias Weller. On the parameterized complexity of consensus clustering. *Theoretical Computer Science*, 542:71–82, 2014. doi:10.1016/j.tcs.2014.05.002.

16 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

17 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Frances A. Rosamond, Saket Saurabh, and Yngve Villanger. Local search: Is brute-force avoidable? *Journal of Computer and System Sciences*, 78(3):707–719, 2012.

18 Michael R. Fellows, Michael A. Langston, Frances A. Rosamond, and Peter Shaw. Efficient parameterized preprocessing for Cluster Editing. In *Proceedings of the 16th International*

Symposium on Fundamentals of Computation Theory (FCT '07), volume 4639 of *LNCS*, pages 312–321. Springer, 2007. doi:10.1007/978-3-540-74240-1_27.

19 Jaroslav Garvardt, Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Parameterized local search for max c-cut. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, pages 5586–5594. ijcai.org, 2023. doi:10.24963/ijcai.2023/620.

20 Serge Gaspers, Joachim Gudmundsson, Mitchell Jones, Julián Mestre, and Stefan Rümmele. Turbocharging treewidth heuristics. *Algorithmica*, 81(2):439–475, 2019. doi:10.1007/s00453-018-0499-1.

21 Serge Gaspers, Eun Jung Kim, Sebastian Ordyniak, Saket Saurabh, and Stefan Szeider. Don't be strict in local search! In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI '12)*. AAAI Press, 2012.

22 Martin Josef Geiger. PACE solver description: A simplified threshold accepting approach for the cluster editing problem. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 34:1–34:2. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.34.

23 Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

24 Niels Grüttemeier and Christian Komusiewicz. On the relation of strong triadic closure and cluster deletion. *Algorithmica*, 82(4):853–880, 2020. doi:10.1007/s00453-019-00617-1.

25 Niels Grüttemeier, Christian Komusiewicz, and Nils Morawietz. Efficient Bayesian network structure learning via parameterized local search on topological orderings. In *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 12328–12335. AAAI Press, 2021. Full version available at https://doi.org/10.48550/arXiv.2204.02902. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17463.

26 Jiong Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8-10):718–726, 2009. doi:10.1016/j.tcs.2008.10.021.

27 Jiong Guo, Sepp Hartung, Rolf Niedermeier, and Ondrej Suchý. The parameterized complexity of local search for TSP, more refined. *Algorithmica*, 67(1):89–110, 2013.

28 Jiong Guo, Danny Hermelin, and Christian Komusiewicz. Local search for string problems: Brute-force is essentially optimal. *Theoretical Computer Science*, 525:30–41, 2014.

29 Sepp Hartung and Rolf Niedermeier. Incremental list coloring of graphs, parameterized by conservation. *Theoretical Computer Science*, 494:86–98, 2013.

30 Giuseppe F. Italiano, Athanasios L. Konstantinidis, and Charis Papadopoulos. Structural parameterization of cluster deletion. In Chun-Cheng Lin, Bertrand M. T. Lin, and Giuseppe Liotta, editors, *Proceedings of the 17th International Conference and Workshops on Algorithms and Computation (WALCOM 2023)*, volume 13973 of *Lecture Notes in Computer Science*, pages 371–383. Springer, 2023. doi:10.1007/978-3-031-27051-2_31.

31 Maximilian Katzmann and Christian Komusiewicz. Systematic exploration of larger local search neighborhoods for the minimum vertex cover problem. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI '17)*, pages 846–852. AAAI Press, 2017.

32 Leon Kellerhals, Tomohiro Koana, André Nichterlein, and Philipp Zschoche. The PACE 2021 parameterized algorithms and computational experiments challenge: Cluster editing. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 26:1–26:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.26.

33 Christian Komusiewicz, Simone Linz, Nils Morawietz, and Jannik Schestag. On the complexity of parameterized local search for the maximum parsimony problem. In *Proceedings of the 34th Annual Symposium on Combinatorial Pattern Matching (CPM 2023)*, volume 259 of *LIPIcs*, pages 18:1–18:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.CPM.2023.18.

34 Christian Komusiewicz and Nils Morawietz. Parameterized local search for vertex cover: When only the search radius is crucial. In *Proceedings of the 17th International Symposium on Parameterized and Exact Computation (IPEC 2022)*, volume 249 of *LIPIcs*, pages 20:1–20:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.IPEC.2022.20.

35 Christian Komusiewicz and Frank Sommer. Enumerating connected induced subgraphs: Improved delay and experimental comparison. *Discrete Applied Mathematics*, 303:262–282, 2021.

36 Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012. doi:10.1016/j.dam.2012.05.019.

37 Shaohua Li, Marcin Pilipczuk, and Manuel Sorge. Cluster editing parameterized above modification-disjoint P_3-packings. In *Proceedings of the 38th International Symposium on Theoretical Aspects of Computer Science (STACS '21)*, volume 187 of *LIPIcs*, pages 49:1–49:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.STACS.2021.49.

38 Junjie Luo, Hendrik Molter, André Nichterlein, and Rolf Niedermeier. Parameterized dynamic cluster editing. *Algorithmica*, 83(1):1–44, 2021. doi:10.1007/s00453-020-00746-y.

39 Dániel Marx. Searching the k-change neighborhood for TSP is W[1]-hard. *Operations Research Letters*, 36(1):31–36, 2008.

40 Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. doi:10.1016/j.cosrev.2007.05.001.

41 Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004. doi:10.1007/3-540-36379-3_33.

42 Sylwester Swat. PACE solver description: Clues - a heuristic solver for the cluster editing problem. In *Proceedings of the 16th International Symposium on Parameterized and Exact Computation (IPEC 2021)*, volume 214 of *LIPIcs*, pages 32:1–32:3. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPIcs.IPEC.2021.32.

43 Stefan Szeider. The parameterized complexity of k-flip local search for SAT and MAX SAT. *Discrete Optimization*, 8(1):139–145, 2011.

44 Dekel Tsur. Faster parameterized algorithm for cluster vertex deletion. *Theory Comput. Syst.*, 65(2):323–343, 2021. doi:10.1007/s00224-020-10005-w.

45 Dekel Tsur. Cluster deletion revisited. *Information Processing Letters*, 173:106171, 2022. doi:10.1016/j.ipl.2021.106171.

46 Esther Ulitzsch, Qiwei He, Vincent Ulitzsch, Hendrik Molter, André Nichterlein, Rolf Niedermeier, and Steffi Pohl. Combining clickstream analyses and graph-modeled data clustering for identifying common response processes. *Psychometrika*, 86(1):190–214, 2021. doi:10.1007/s11336-020-09743-0.