Cluster Editing with Locally Bounded Modifications[☆]

Christian Komusiewicz[*], Johannes Uhlmann

Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany

Abstract

Given an undirected graph $G = (V, E)$ and a nonnegative integer k, the NP-hard Cluster Editing problem asks whether G can be transformed into a disjoint union of cliques by modifying at most k edges. In this work, we study how "local degree bounds" influence the complexity of Cluster Editing and of the related Cluster Deletion problem which allows only edge deletions. We show that even for graphs with constant maximum degree Cluster Editing and Cluster Deletion are NP-hard and that this implies NP-hardness even if every vertex is incident with only a constant number of edge modifications. We further show that under some complexity-theoretic assumptions both Cluster Editing and Cluster Deletion cannot be solved within a running time that is subexponential in k, $|V|$, or $|E|$. Finally, we present a problem kernelization for the combined parameter "number d of clusters and maximum number t of modifications incident with a vertex" thus showing that Cluster Editing and Cluster Deletion become easier in case the number of clusters is upper-bounded.

[☆]An extended abstract containing some of the results from this work as well as further fixed-parameter tractability results for Cluster Editing and Cluster Deletion appeared under the title "Alternative Parameterizations for Cluster Editing" in the proceedings of the *37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2011)* [18]. The results of this work are also contained in the first author's dissertation [17].

[*]To whom correspondence should be addressed.

Email addresses: christian.komusiewicz@tu-berlin.de (Christian Komusiewicz), johannes.uhlmann@campus.tu-berlin.de (Johannes Uhlmann)

URL: http://www.user.tu-berlin.de/ckomus/ (Christian Komusiewicz), http://theinf1.informatik.uni-jena.de/~uhlmann/ (Johannes Uhlmann)

Keywords: graph modification problems, parameterized algorithmics, exponential-time hypothesis, data reduction

1. Introduction

The NP-hard Cluster Editing problem is among the best-studied parameterized problems. It is usually defined as follows:

> Cluster Editing
> **Input:** An undirected graph $G = (V, E)$ and an integer $k \geq 0$.
> **Question:** Can G be transformed by up to k edge modifications into a cluster graph?

Herein, an *edge modification* is either the deletion or insertion of an edge and a *cluster graph* is a graph where every connected component is a clique. The cliques of a cluster graph are referred to as *clusters*. The NP-hard Cluster Deletion problem, which is also studied in this work, is defined analogously except that only edge deletions are allowed.

One way of attacking the NP-hardness of Cluster Editing are fixed-parameter algorithms that run in time $f(k) \cdot \text{poly}(n)$ time where k is a problem-specific parameter and n is the input size. Fixed-parameter algorithms are thus fast in case k is small. So far, the proposed fixed-parameter algorithms for Cluster Editing almost exclusively employ the parameter number k of edge modifications [5, 3, 6, 8, 13, 14]. The focus on this parameterization is contrasted by the observation that k is often not really small for real-world instances. For example in a protein similarity data set that has been frequently used for evaluating Cluster Editing algorithms, the instances with $n \geq 30$, n being the number of vertices, have an average number k of edge modifications that is between $2n$ and $4n$ [5]. Hence, it would be interesting to show fixed-parameter tractability for parameters that are *stronger* than the parameter number k of edge modifications, that is, parameters that are always at most as large as k and that can be arbitrarily small compared to k.

In this work, we consider a parameter that is naturally a *stronger* parameter than the number k of edge modifications. We call this parameter *local modification bound* t. In the following, we refer to a set of edge deletions and insertions as *edge modification set*.

Definition 1. *Let $G = (V, E)$ be an undirected graph, and let S be an edge modification set for G. We say that S is* locally t-bounded *if for every vertex $v \in V$ it holds that*

$$|\{e \in S \mid v \in e\}| \leq t.$$

Informally, this means that a locally t-bounded edge modification set performs at most t edge modifications on each vertex of the input graph. Another intuitive way of looking at locally t-bounded edge modification sets is to visualize the graph that has vertex set V and edge set S. If S is locally t-bounded, then this graph has maximum degree t.

The local modification bound t relates to the overall number k of edge modifications in the following way: First, any edge modification set S is clearly locally $|S|$-bounded. Second, the local modification bound t can be arbitrarily small compared to the overall number of edge modifications. Hence, the local modification bound t is indeed a stronger parameter than the overall number of edge modifications. We expect that in most practically relevant instances the local modification bound t is much smaller than the overall number of edge modifications. As we observe in Section 2, the local modification bound is upper-bounded by the maximum degree Δ of the input graph which is the second parameter that we consider. Unfortunately, as we show in this work, it turns out that Cluster Editing and Cluster Deletion are NP-hard already for constant Δ and also for constant t.

A further way to counter the fact that k is usually not that small would be to present subexponential-time fixed-parameter algorithms for the parameter k; so far, all presented fixed-parameter algorithms for Cluster Editing have running time $2^{\Omega(k)} \cdot \text{poly}(|V|)$. We show, however, that under the so-called exponential-time hypothesis, Cluster Editing and Cluster Deletion cannot be solved within time that is subexponential in the number k of edge modifications or in the size of the input graph. Furthermore, this result holds even if Δ is a constant. To contrast these hardness results, we show that parameterizing by the combined parameter "upper bound d on the number of clusters and local modification bound t" yields fixed-parameter tractability.

Related Work. The NP-hardness of Cluster Editing has been shown several times [19, 22, 2]. The currently fastest fixed-parameter algorithm for parameter k has running time $O(1.62^k + |E|)$ [3], and the currently smallest problem kernel contains at most $2k$ vertices [8]. Other parameterizations

have played a marginal role so far. To the best of our knowledge, the only other parameter that has been considered is the "cluster vertex deletion number" which is the number of vertices one needs to delete in order to obtain a cluster graph. Cluster Editing and Cluster Deletion are both fixed-parameter tractable with respect to the cluster vertex deletion number of the input graph [18, 23]. However, the running times of the algorithms for this parameter seem to be impractical so far.

A variant of Cluster Editing in which the number of clusters is fixed (instead of upper-bounded as we consider in Section 4) has been previously studied: For every $d \geq 2$ it is NP-hard to decide whether the input graph can be transformed by at most k edge modifications into a graph with *exactly* d clusters [22]. Guo [14] showed that this variant of Cluster Editing admits a problem kernel consisting of at most $(d+2) \cdot k + d$ vertices. Finally, Fomin et al. [11] presented a randomized algorithm that solves Cluster Editing with exactly d clusters in $2^{O(\sqrt{dk})} + \text{poly}(|V|)$ time. This algorithm can also be used to solve Cluster Editing with at most d clusters in the same running time.

While not as extensively studied as Cluster Editing, some results have been obtained for Cluster Deletion as well: Cluster Deletion is NP-hard in general and when one demands that the cluster graph has exactly $d \geq 3$ clusters but polynomial-time solvable when one demands that the cluster graph has exactly two clusters [22]. Cluster Deletion can be solved in $O(1.415^k + |V|^3)$ time by a search tree algorithm [4].

Our Results. Table 1 summarizes our findings which are as follows. We present a reduction from 3-SAT to Cluster Editing which yields several hardness results.[1] First, we can infer that Cluster Editing is NP-hard even on input graphs with maximum degree six. Second, we can infer that Cluster Editing is NP-hard even when every solution is locally 4-bounded. Hence, the local modification bound itself is not a suitable parameter for Cluster Editing. Finally, the reduction from 3-SAT shows that Cluster Editing does not admit an algorithm with running time $2^{o(k)} \cdot \text{poly}(|V|)$ time unless the exponential-time hypothesis fails. Our result on the nonexistence of such a subexponential-time algorithm for the parameter k negatively answers a recent conjecture by Cao and Chen [7].

[1] Previous NP-hardness results were obtained for example by reductions from 3-Dimensional Matching [19] or Exact Cover by 3-Sets [22].

Table 1: Summary of our results for Cluster Editing and Cluster Deletion and the parameters maximum degree Δ, local modification bound t, number k of edge modifications, and the combined parameter (d, t) where d is an upper bound on the number of clusters in the cluster graph. The results for parameter k hold unless the exponential-time hypothesis fails.

Parameter	Cluster Editing	Cluster Deletion
Δ	NP-hard for $\Delta \geq 6$	NP-hard for $\Delta \geq 4$, $\in P$ for $\Delta \leq 3$
t	NP-hard for $t \geq 4$	NP-hard for $t \geq 2$
k	No $2^{o(k)} \cdot \text{poly}(\lvert V\rvert)$ algorithm	No $2^{o(k)} \cdot \text{poly}(\lvert V\rvert)$ algorithm
(d, t)	$4dt$-vertex kernel	$2dt$-vertex kernel

Independently to our work, Fomin et al. [11] also showed that Cluster Editing does not admit a subexponential-time algorithm for parameter k.

For Cluster Deletion, we can show hardness for even more restricted cases by observing close connections to Partition Into Triangles. We show that Cluster Deletion is NP-hard even when the input graph has maximum degree four, and that it is NP-hard even when every solution is locally 2-bounded. Again, we also observe that our results imply that Cluster Deletion does not admit an algorithm with running time $2^{o(k)} \cdot \text{poly}(|V|)$ unless the exponential-time hypothesis fails. We also show that Cluster Deletion is polynomial-time solvable on graphs with maximum degree three, thus achieving a dichotomy with respect to the maximum degree of the input graph.

We complement the negative results for Cluster Editing and Cluster Deletion by showing that both problems are fixed-parameter tractable with respect to the combined parameter (d, t), where d is an upper bound on the number of clusters in the cluster graph and t is the local modification bound. More precisely, we consider a constrained version of both problems that might be of independent interest. Our algorithms for these problems are based on simple data reduction rules that produce in $O(|V|^3)$ time a problem kernel consisting of at most $4dt$ vertices (in the case of Cluster Editing) and $2dt$ vertices (in the case of Cluster Deletion).

Preliminaries. We only consider simple undirected graphs $G = (V, E)$. Unless stated otherwise, we use $n := |V|$ to denote the number of vertices of a graph and $m := |E|$ to denote the number of edges. For a vertex v, we

denote with $N(v) := \{u \in V \mid \{u, v\} \in E\}$ the *neighborhood* of v. The *closed neighborhood* is defined as $N[v] := N(v) \cup \{v\}$. We call an edge modification set of size at most k that produces a cluster graph a *solution*.

We briefly recall the relevant notions of parameterized algorithmics; for an introduction to the field refer to the monographs of Downey and Fellows [9], Flum and Grohe [10], and Niedermeier [21]. Parameterized problems consist of two dimensions. One dimension is the input instance I (as in classical complexity theory), and the other one is a parameter k. A parameterized problem L is *fixed-parameter tractable* if there is an algorithm that decides in $f(k) \cdot \text{poly}(|I|)$ time whether $(I, k) \in L$, where f is a computable function depending only on k. A parameterized problem L admits a *problem kernel* if there is a polynomial-time transformation of any instance (I, k) to an instance (I', k') such that $(I, k) \in L \Leftrightarrow (I', k') \in L$, $|I'| \leq g(k)$, and $k' \leq k$. The function $g(k)$, which depends only on k, is called the *size* of the problem kernel. A data reduction algorithm that yields a problem kernel is called *kernelization*. Kernelizations are often represented by a set of *data reduction rules*. A *data reduction rule* is a reduction from an instance of (I, k) of a parameterized problem L to an instance (I', k') of L. We say that a data reduction rule is *correct* if $(I, k) \in L$ if and only if $(I', k') \in L$. We say that a data reduction rule has been *exhaustively applied* if any further application of this rule does not modify the instance. An instance is called *reduced* with respect to a set of data reduction rules if each data reduction rule in the set has been exhaustively applied.

The exponential-time hypothesis states that x-SAT, $x \geq 3$, cannot be solved within a running time of $2^{o(n)}$ or $2^{o(m)}$, where n is the number of variables and m is the number of clauses in the input x-CNF formula. This approach for showing super-polynomial lower bounds for running times goes back to work of Impagliazzo et al. [15]; some aspects and applications of the exponential-time hypothesis are discussed in surveys by Lokshtanov et al. [20] and Woeginger [26]. In this context, algorithms with running time $2^{o(p)}$ for some parameter p are called *subexponential-time* algorithms.

2. Constant Maximum Degree and Constant Local Modification Bound

We show that Cluster Editing is NP-hard even when restricted to graphs with maximum degree six. To the best of our knowledge the previous NP-hardness proofs require an unbounded degree [19, 2, 22]. As an imme-

diate consequence of our NP-hardness proof, Cluster Editing is NP-hard even for a constant local modification bound. The following structural lemma will be used in our proof of NP-hardness.

Lemma 1. *Let $G = (V, E)$ be an undirected graph. There is a minimum-cardinality solution S producing a cluster graph G' such that for all vertices $u, v \in V$ with $|N(u) \cap N(v)| \leq 1$ and $\{u, v\} \notin E$ it holds that u and v are in different clusters of G'.*

Proof. Assume that there is a minimum-cardinality solution S that yields a cluster graph G' such that there is a pair of vertices $u, v \in V$ with $|N(u) \cap N(v)| \leq 1$ and $\{u, v\} \notin E$ that are in the same cluster K of G'. We show that one can construct from S a solution S' with $|S'| \leq |S|$ that yields a cluster graph G'' in which either u or v is a singleton cluster.

Let $X := N(u) \cap N(v)$ be the common neighborhood of u and v in G, let $K_v := K \cap N(v) \setminus X$, and let $K_u := K \cap N(u) \setminus X$. Note that $|X| \leq 1$. Without loss of generality, assume that $|K_v| \geq |K_u|$. Then, u is in G adjacent to at most $\lfloor (|K|-1)/2 \rfloor$ vertices in K since $|K_u| \leq \lfloor (|K|-3)/2 \rfloor$ and since u has in G at most one further neighbor in K (because $|X| \leq 1$). Therefore, cutting u from K yields a solution S' with $|S'| \leq |S|$ since this operation "undoes" at least $\lceil (|K| - 1)/2 \rceil$ edge insertions and causes at most $\lfloor (|K| - 1)/2 \rfloor$ additional edge deletions.

Exhaustively applying the modification above for each such pair of vertices results in a minimum-cardinality solution with the desired property. Since each application of this modification produces at least one singleton cluster, there can be at most n iterations of this procedure. Hence, a solution with the desired property does indeed exist. □

In the Cluster Editing instances produced by the reduction, any two nonadjacent vertices have at most one vertex in common. Hence, the lemma above implies that in every one of these instances there is an optimal solution that only deletes edges.

For the NP-hardness proof we present a reduction from 3-SAT, which has as input a boolean formula ϕ in conjunctive normal form with at most three literals per clause (3-CNF) and asks whether there is an assignment to the variables of ϕ that fulfills all clauses of ϕ.[2] For simplicity, we assume that

[2] A similar reduction was previously used to show NP-hardness of the Transitivity Editing problem which is defined on directed graphs [25].

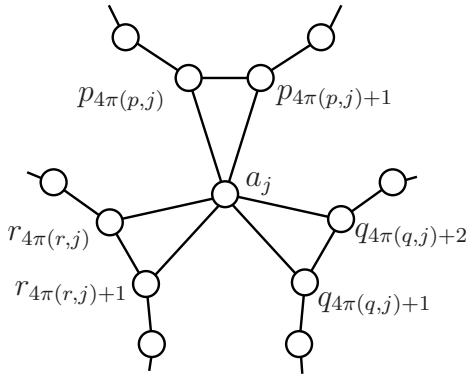

Figure 1: Illustration of the clause gadget for a clause $C_j = (x_p \vee \overline{x_q} \vee x_r)$. Note that for each variable either the "+0/+1"-vertices (if it is nonnegated) or the "+1/+2"-vertices (if it is negated) are adjacent to a_j, but never the "+3"-vertex.

every clause contains *exactly* three literals; this can be easily achieved by adding a further variable x to each clause with two variables, and forcing x to be false by a constant number of further clauses and variables.

The basic idea of the reduction is as follows. For each variable x_i of a given 3-CNF formula ϕ, we construct a *variable cycle* of length $4m_i$, where m_i denotes the number of clauses that contain x_i. It is easy to verify that only deleting every second edge yields a minimum-cardinality edge modification set for transforming an even-length cycle into a cluster graph. The corresponding two possibilities are used to represent the two choices for the value of x_i. Moreover, for each clause C_j containing the variables x_p, x_q, and x_r, we connect the three corresponding variable cycles by a clause gadget. In doing so, the goal is to ensure that if the solutions for the variable gadgets correspond to an assignment that satisfies C_j, then one needs only four edge modifications for the clause gadget and otherwise one needs at least five edge modifications. Let m be the number of clauses in ϕ and observe that, since ϕ is a 3-CNF formula, the overall number of vertices in the variable cycles is $12m$. Our construction guarantees that there is a satisfying assignment for ϕ if and only if the constructed graph can be transformed into a cluster graph by exactly $6m + 4m = 10m$ edge modifications, where $6m$ modifications are used for the variable cycles and $4m$ modifications are used for the clause gadgets. The details follow.

Given a 3-CNF formula ϕ consisting of the clauses $C_0, \ldots, C_{m-1}$ over

the variables $\{x_0, \ldots, x_{n-1}\}$, construct a Cluster Editing-instance $(G = (V, E), k)$ as follows.

For each variable x_i, $0 \leq i < n$, G contains a *variable cycle* that consists of the vertices $V_i^v := \{i_0, \ldots, i_{4m_i-1}\}$ and the edges $E_i^v := \{\{i_k, i_{k+1}\} \mid 0 \leq k < 4m_i\}$ (for ease of presentation let $i_{4m_i} = i_0$). An edge $\{i_x, i_{x+1}\}$ is *even* if x is even, and *odd* otherwise. So far, the constructed graph consists of a disjoint union of cycles and has $12m$ vertices and edges. Next, we add a *clause gadget* to G for each clause of ϕ.

In the construction of the clause gadgets, we need for each clause C in the variable cycles of C's variables a fixed set of vertices that are "reserved" for C. To this end, suppose that for each variable x_i an arbitrary but fixed ordering of the clauses that contain x_i is given, and let $\pi(i, j) \in \{0, \ldots, 4m_i - 1\}$ denote the position of a clause C_j that contains x_i in this ordering. We now give the details of the construction of the clause gadgets. Let C_j be a clause containing the variables x_p, x_q, and x_r (either negated or nonnegated). We construct a clause gadget connecting the variable cycles of x_p, x_q, and x_r. First, let a_j be a new vertex that appears only in the clause gadget for clause C_j. Let E_j^c denote the edge set of the clause gadget and let E_j^c contain for each $i \in \{p, q, r\}$ the edges $\{a_j, i_{4\pi(i,j)}\}$ and $\{a_j, i_{4\pi(i,j)+1}\}$ if x_i occurs nonnegated in C_j or the edges $\{a_j, i_{4\pi(i,j)+1}\}$ and $\{a_j, i_{4\pi(i,j)+2}\}$, otherwise. See Figure 1 for an illustration. Then, the construction of $G = (V, E)$ is completed by setting $V := \bigcup_{i=0}^{n-1} V_i^v \cup \bigcup_{j=0}^{m-1} \{a_j\}$ and $E := \bigcup_{i=0}^{n-1} E_i^v \cup \bigcup_{j=0}^{m-1} E_j^c$.

Theorem 1. Cluster Editing *is NP-hard even when restricted to graphs with maximum vertex degree six.*

Proof. Let ϕ be a 3-SAT formula and let G be constructed from ϕ as described above. We show the correctness of the reduction by showing the following claim.

> ϕ is satisfiable $\Leftrightarrow$ G can be transformed into a cluster graph by at most $k := 10m$ edge modifications

In the following, we use the characterization of cluster graphs as the graphs that do not contain an induced P_3, that is, an induced path on three vertices. This well-known characterization of cluster graphs has been used repeatedly in the literature.

$\Rightarrow$: Given a satisfying assignment β for ϕ we can transform G into a cluster graph as follows. For each variable x_i delete the odd edges of the

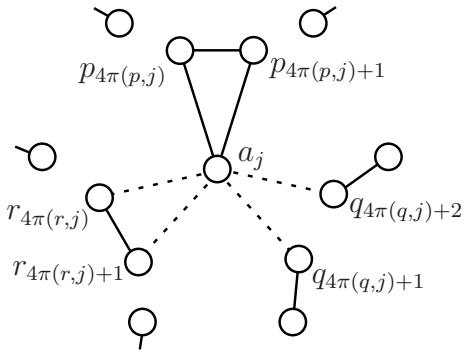

Figure 2: If all odd edges in the variable cycle of x_p are deleted (observe that x_p occurs nonnegated in C_j, since a_j is adjacent to $p_{4\pi(p,j)}$ and $p_{4\pi(p,j)+1}$), then all induced P_3s that contain a_j can be destroyed by four additional edge deletions (marked by dotted lines).

variable cycle of x_i if $\beta(x_i) = \text{true}$ and the even edges otherwise. Moreover, for each clause C_j proceed as follows. Assume that C_j contains the variables x_p, x_q, and x_r. Without loss of generality assume that the literal that corresponds to x_p is true. All induced P_3s that contain a_j can be destroyed by the deletion of the four edges with one endpoint being a_j and the other endpoints from $V_q^v \cup V_r^v$ (see Figure 2). For the variable cycles, we perform altogether $\sum_{0 \leq i < n} 4m_i/2 = 6m$ edge modifications, and for each clause gadget four edges are deleted. Hence, $10m$ edge modifications are performed overall. By construction, every induced P_3 contains either three vertices of the same variable cycle or at least one of the a_j's. Hence, all induced P_3s are destroyed and the resulting graph is a cluster graph.

$\Leftarrow$: Let S denote an optimal solution for G with $|S| \leq k := 10m$. To show that ϕ is satisfiable, we need some observations about the structure of G and S.

First, we show that $10m$ is a lower bound on any solution for G, that is, $|S| \geq 10m$ and thus $|S| = 10m$. By the construction of G, for every nonadjacent pair of vertices u, v in G, it holds that $|N(u) \cap N(v)| \leq 1$. Therefore, we can assume, by Lemma 1, that S performs only edge deletions (since no nonadjacent vertices end up in the same cluster). Furthermore, note that for each variable x_i the variable cycle contains $4m_i/2$ edge-disjoint induced P_3s with all three vertices on the cycle and that deleting either all even or all odd edges are the only two optimal ways to destroy these induced P_3s.

Hence, G contains $6m$ edge-disjoint induced P_3s such that all three vertices of the induced P_3 are in the same variable cycle. Clearly, at least $6m$ edge deletions are needed for these induced P_3s. For each clause C_j, $0 \leq j < m$, at least four edge deletions are needed to destroy all induced P_3s that contain a_j as a middle vertex. Observe that these four edge deletions are all incident with a_j, that is, they are from the clause gadget E_j^c, and thus they do not contain edges from variable cycles. Hence, every solution has size at least $10m$ and thus $|S| = 10m$.

Now, since at least $6m$ edges are deleted in the variable cycles, this means that for each clause C_j *exactly* four edges incident with a_j are deleted by S. Consequently, for each variable cycle either all even or all odd edges are deleted.

Consider the assignment β for ϕ that, for each x_i, $0 \leq i < n$, sets $\beta(x_i) :=$ true if all odd edges of V_i^v are deleted and sets $\beta(x_i) :=$ false if all even edges of V_i^v are deleted. We show that β is a satisfying assignment. Consider an arbitrary clause C_j containing the variables x_p, x_q, and x_r. Since in the final cluster graph a_j is not a middle vertex of a P_3, it can have edges to at most one variable, say x_p, of C_j. Furthermore, since exactly four edge deletions are incident with a_j, *both* edges that are incident with the vertices of the variable cycle of x_p are not deleted by S. Without loss of generality, assume that x_p appears nonnegated in C_j. Then the two vertices of V_p^v that are adjacent to a_j are $p_{4\pi(p,j)}$ and $p_{4\pi(p,j)+1}$. Since S is a solution, the edge $\{p_{4\pi(p,j)}, p_{4\pi(p,j)+1}\}$ is not deleted by S. Hence, all odd edges of V_p^v are deleted, and therefore the assignment β fulfills clause C_j. □

We can use the presented reduction to obtain further hardness results for Cluster Editing. Obviously, since the constructed graph has maximum degree six, every optimal solution is locally 6-bounded. This is due to the fact that if a vertex v is incident with more than 6 edge modifications, then one can obtain a better solution by undoing these edge modifications and deleting all edges that are incident with v in G.

This observation can be strengthened even further by observing that, by the construction of G, we either need more than $10m$ edge modifications or that the maximum number of edge modifications per vertex is four. The latter can be seen as follows. As described in the proof of Theorem 1, if there is a solution of size at most $10m$, then there is also a solution that only performs edge deletions and that has the following further properties. It performs $6m$ edge deletions in the variable cycles, and on each vertex in the

variable cycle at most one of the deleted edges is incident. Note that each of the vertices in the variable cycle has at most one neighbor in a clause gadget. Hence, for each vertex of the variable cycle at most two edge deletions are performed on incident edges. Furthermore, for each clause gadget exactly four edge deletions are performed. Hence, we can assume that there is a solution that is locally 4-bounded.

Corollary 1. Cluster Editing *is NP-hard even when the input is restricted such that every yes-instance has a solution that is locally* 4*-bounded.*

Our final hardness result for Cluster Editing can be drawn from the observation that the solution size is ten times the number of clauses in the 3-CNF formula. By our reduction, a subexponential-time algorithm for Cluster Editing parameterized by k would imply an algorithm for solving 3-SAT that has running time subexponential in the number m of clauses. The same can be observed for the number $|V|$ of vertices and the number $|E|$ of edges in the Cluster Editing instance. Hence, we arrive at the following.

Theorem 2. Cluster Editing *cannot be solved in* $2^{o(k)} \cdot \mathrm{poly}(|V|)$ *time, in* $O(2^{o(|V|)})$ *time, or in* $O(2^{o(|E|)})$ *unless the exponential-time hypothesis fails. This holds even when the input graph has maximum degree six.*

For Cluster Deletion, we can obtain hardness for even more restricted input graphs by observing close connections to Partition Into Triangles on graphs with maximum degree four. As recently shown by van Rooij et al. [24], Partition Into Triangles is NP-hard even when the input graph $G = (V, E)$ is 4-regular. Moreover, NP-hardness persists even when for each vertex $v \in V$ the graph $G[N[v]]$ is isomorphic to one of the three graphs shown in Figure 3 [24]. In the following, we refer to such graphs as 4*-regular neighborhood-restricted graphs*. The variant of Partition Into Triangles that we use in our reduction is formalized as follows:

> Restricted Partition Into Triangles (RPIT)
> **Input:** An undirected 4-regular neighborhood restricted graph $G = (V, E)$.
> **Question:** Can V be partitioned into $|V|/3$ sets such that each set of the partition induces a triangle, that is, a complete graph on three vertices, in G?

The following easy observation is useful for establishing the connection to Cluster Deletion.

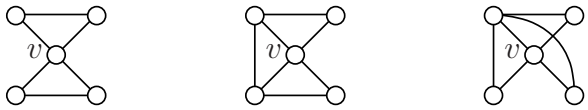

Figure 3: The three different neighborhoods in a 4-regular neighborhood restricted graph. None of these graphs contains a clique of order four.

Observation 1. *Let $G = (V, E)$ be a 4-regular neighborhood-restricted graph. Then G does not contain any clique of order four or more.*

The observation says that if we use a 4-regular neighborhood-restricted graph as input graph for Cluster Deletion, then the largest clusters in the resulting cluster graph are triangles. In the next lemma, we show that the case in which every cluster is a triangle is optimal. Let $n := |V|$ in what follows.

Lemma 2. *Let $G = (V, E)$ be an instance of* RPIT*. Then, G is a yes-instance of* RPIT $\Leftrightarrow$ $(G, k := n)$ *is a yes-instance of* Cluster Deletion*.*

Proof. We show both directions separately.

$\Rightarrow$: Let G be a yes-instance of RPIT, and let $G_1, \ldots, G_{n/3}$ denote a set of triangles into which the input graph can be partitioned. Note that each G_i contains three edges and three vertices. Since G is 4-regular, it has $2n$ edges. Hence, there are exactly n edges that are not contained in any G_i. Deleting these edges from G yields a cluster graph, since each component is a triangle.

$\Leftarrow$: Let $S \subseteq E$ be an edge set of size at most $k := n$ such that deleting S from G yields a cluster graph G'. By Observation 1, every cluster contains at most three vertices. Each cluster on three vertices has exactly three edges and clusters with one or two vertices have less edges than vertices. Consequently, G' has at most n edges. Since $|S| \leq n$ and $|E| = 2 \cdot n$, G' has exactly n edges. Hence, every cluster is a triangle. Consequently, the clusters are a set of vertex-disjoint triangles, and I is thus a yes-instance of RPIT. □

The above lemma directly implies a polynomial-time reduction from RPIT to Cluster Deletion on 4-regular neighborhood-restricted graphs: all that needs to be done is to set $k := n$. Our main result that can be obtained by using this reduction is as follows.

Theorem 3. Cluster Deletion *is NP-hard even on 4-regular graphs.*

Note that since the input graph of the Cluster Deletion instance is 4-regular, and since every cluster must be a triangle every solution is locally 2-bounded.

Corollary 2. Cluster Deletion *is NP-hard even when the input is restricted such that every solution is locally* 2*-bounded.*

Finally, we can also obtain lower bounds for the running time of Cluster Deletion with respect to parameter k. RPIT does not admit a subexponential-time algorithm [24]. Since we can reduce RPIT to Cluster Deletion instances on the same graph with $k = n$, we arrive at the following.

Theorem 4. Cluster Deletion *cannot be solved in* $2^{o(k)} \cdot \text{poly}(|V|)$ *time, in* $O(2^{o(|V|)})$ *time, or in* $O(2^{o(|E|)})$ *unless the exponential-time hypothesis fails. This holds even when the input graph is 4-regular.*

3. Cluster Deletion on graphs with maximum degree three

In the following, we present a polynomial-time algorithm for Cluster Deletion in case the input graph has maximum degree three. Hence, we obtain the following dichotomy: Cluster Deletion is polynomial-time solvable on graphs with maximum degree three, and NP-hard, otherwise. The main idea of the presented algorithm is as follows. The algorithm starts by exhaustively applying two data reduction rules. One rule deals with all isolated cliques in the input graph and, as we show, hence with all clusters of size four in the cluster graph. The other rule deals with a certain type of triangles. We then show that after these reduction rules have been exhaustively applied, we can reduce our instance to a weighted version of Cluster Deletion whose input graph is triangle-free. Finally, we show that this instance can be solved by computing a maximum-weight matching.

Next, we present the two reduction rules in detail. The aim of the first reduction rule is to deal with all clusters of size four in the final cluster graph. Suppose that the cluster graph contains such a cluster. Then, since the input graph G has maximum degree three, this cluster must be a connected component of G and thus an isolated clique of G. Hence, we can remove all vertices that are part of these clusters in $O(n)$ time with the following trivial reduction rule.

Reduction Rule 1. *Remove from G all connected components that are cliques.*

Clearly, Reduction Rule 1 is correct and can be exhaustively applied in $O(n)$ time. We now present the second data reduction rule.

Reduction Rule 2. *If G contains three vertices u, v, and w such that*

- *$\{u, v, w\}$ induces a triangle in G, and*
- *there is no vertex $x \in V \setminus \{u, v, w\}$ that has at least two neighbors in $\{u, v, w\}$,*

then delete all edges between $\{u, v, w\}$ and $V \setminus \{u, v, w\}$, decrease k by the number of performed edge deletions, and remove $\{u, v, w\}$ from G.

Lemma 3. *Reduction Rule 2 is correct and can be exhaustively performed in $O(n)$ time.*

Proof. We first prove the correctness of the rule, and then bound its running time. To show the correctness of the rule, we show that there is an optimal solution that yields a cluster graph in which $\{u, v, w\}$ is a cluster. Let $S \subseteq E$ be an optimal solution, let $G' := (V, E \setminus S)$ be the resulting cluster graph, and assume that $\{u, v, w\}$ does not form a cluster of G'. Then, either three or two edges between u, v, and w are deleted (if only one edge is deleted then u, v, and w induce a P_3). In the first case, we can obtain a solution S' by undoing all three edge deletions between u, v, and w and instead deleting the at most three edges between $\{u, v, w\}$ and $V \setminus \{u, v, w\}$. Clearly $|S'| \leq |S|$. In the second case, suppose that $\{u, v\}$ is not deleted by S. Then, $\{u, v\}$ is a cluster of G'. We can obtain a solution S' from S by undoing the deletion of $\{u, w\}$ and $\{v, w\}$ and instead deleting at most one edge between w and $V \setminus \{u, v, w\}$. Since $|S'| < |S|$, S is not an optimal solution, a contradiction.

The running time can be seen as follows. First, we can label in $O(n)$ time the edges of all triangles to which Reduction Rule 2 applies by checking for each vertex $v \in V$ whether $N[v]$ contains a triangle that fulfills the condition of the rule. Then, we can delete in $O(n)$ time all unlabeled edges that have a common endpoint with a labeled edge, since these are precisely the "outgoing" edges of a triangle that fulfills the condition of the rule. After the deletion of these edges, the rule has been exhaustively applied since the application of the rule does not create "new" triangles to which

the rule can be applied. This can be seen as follows. Observe that the endpoints of an edge e that is deleted by Reduction Rule 2 do not have any common neighbors, since one of e's endpoints is in a triangle in which no two vertices have a common neighbor outside the triangle and G has maximum degree three. Now suppose that the deletion of an edge e produces a triangle $T = \{u, v, w\}$ to which Reduction Rule 2 applies. Clearly, e must be incident with one vertex from T. Hence, assume without loss of generality that $e = \{u, x\}$. Since the triangle T did not fulfill the condition of the rule before the deletion of $\{u, x\}$, the vertex x must have another neighbor in T, say w. This contradicts the observation that the endpoints of a deleted edge do not have any common neighbors. Hence, Reduction Rule 2 can be exhaustively applied in one pass which can be performed in $O(n)$ time. □

A graph with maximum degree three to which neither Reduction Rule 1 nor Reduction Rule 2 applies has the following property: for each triangle $\{u, v, w\}$ there is at least one other vertex x that has two neighbors, say u and v, in the triangle. In other words, every triangle has two vertices u and v that have two common neighbors. Since the graph has maximum degree three and since they are adjacent, it holds that $N[u] = N[v]$. Note that since the graph does not contain cliques of size four after Reduction Rule 1 has been applied, there is also no further vertex y that is adjacent to two vertices in $\{u, v, w\}$. Altogether this leads to the following observation.

Observation 2. *Let G be a graph with maximum degree three that is reduced with respect to Reduction Rules 1 and 2. Then, every triangle contains exactly two degree-three vertices u and v with $N[u] = N[v]$.*

The above observation can be used in the following way: the vertices u and v are part of exactly two triangles, and they can be in at most one of those triangles in a cluster graph. Furthermore, the two vertices that are neighbors of u and v are part of exactly one triangle since they have at most one further neighbor. Hence, all triangles come in isolated pairs of which at most one is a cluster of the cluster graph. We will show that in this case two vertices in the intersection of two triangles end up in the same cluster. We can therefore "get rid" of these triangles by reducing the problem to a weighted version of Cluster Deletion by merging the two vertices. The resulting instance of this weighted version is triangle-free which makes it possible to compute an optimal solution by computing a maximum-weight matching.

Lemma 4. *Let (G, k) be an instance of* Cluster Deletion *such that G has maximum degree three and (G, k) is reduced with respect to Reduction Rules 1 and 2. Then, (G, k) can be solved in $O(n^{1.5} \cdot \log^2 n)$ time.*

Proof. Let (G, k) be as described in the lemma. We describe a polynomial-time algorithm for (G, k) that consists of two main steps. First, we reduce (G, k) to a triangle-free instance of the following edge-weighted version of Cluster Deletion:

> Weighted Cluster Deletion
> **Input:** An undirected graph $G = (V, E)$, an edge-weight function $\omega : E \to \mathbb{N} \setminus \{0\}$, and an integer $k \geq 0$.
> **Question:** Is there an edge set $S \subseteq E$ such that deleting S from G results in a cluster graph and $\sum_{e \in S} \omega(e) \leq k$?

Afterwards, we show that triangle-free instances of Weighted Cluster Deletion can be solved in polynomial time by computing a maximum-weight matching.

The reduction from Cluster Deletion to Weighted Cluster Deletion works as follows. First, we set $\omega(e) = 1$ for each $e \in E$ and thus obtain an instance of Weighted Cluster Deletion. Clearly, this instance is equivalent to the original instance. Then, we further apply the following reduction rule to reduce this instance of Weighted Cluster Deletion into a triangle-free instance of Weighted Cluster Deletion.[3] As long as G contains a triangle, do the following. Let u and v denote the degree-three vertices of the triangle with $N[u] = N[v]$ (by Observation 2 there is exactly one such pair of vertices). Furthermore, let w and x denote the other two neighbors of u and v. Then, remove u from G and set $\omega(v, w) := 2$ and $\omega(v, x) := 2$. Note that after u is removed from G, v has degree two and is not contained in any triangle in G.

The correctness of the reduction rule described above can be seen as follows. Since $N[u] = N[v]$ and by Observation 2, u and v are a so-called critical clique, that is, a maximal vertex set in which all vertices have the same closed neighborhood. Furthermore, all edges incident with u and v have

[3]The presented reduction rule is similar to previous approaches for Cluster Editing that replace an unweighted instance by a weighted instance that works on the so-called critical clique graph [5]. For the sake of completeness we include a short proof of correctness.

weight one since u and v are still part of a triangle. Every optimal solution puts u and v into the same cluster which can be seen as follows. Suppose that there is an optimal solution S that puts u and v into different clusters. Since S is optimal, there must be a vertex w such that one of u and v, say u is in a cluster with w: otherwise, undoing the deletion of $\{u, v\}$ yields a better clustering. Then, by undoing the deletions of $\{u, v\}$ and $\{v, w\}$ and deleting at most one other edge instead, we obtain a better solution. As a consequence, if $\{u, w\}$ is deleted by an optimal solution, then also $\{v, w\}$ is deleted by this solution. Hence, every optimal solution before the removal of u one-to-one corresponds to an optimal solution after the removal of u (and the subsequent increase of the edge weights).

After all triangles have been replaced by edges of weight two, we have a triangle-free instance of Weighted Cluster Deletion. We now show that this instance can be solved in polynomial time. The basis of this algorithm is the following claim:

> Let $G = (V, E)$ be a triangle-free graph, let $S \subseteq E$ be an edge set, and let $M := E \setminus S$. Then, $(V, E \setminus S)$ is a cluster graph $\Leftrightarrow$ M is a matching.

This claim can be seen as follows. Since G is triangle-free, any cluster graph that can be obtained by edge deletions has clusters of size at most two. Hence, the edges of this cluster graph are a matching. The converse is also true, since any two edges of a matching do not have an endpoint in common. Therefore, the graph that contains these edges and all vertices of the input graph is a cluster graph. Furthermore, since

$$\sum_{e \in S} \omega(e) = \sum_{e \in E} \omega(e) - \sum_{e \in M} \omega(e)$$

for $S \subseteq E$ and $M := E \setminus S$, minimizing the sum of the weights of the deleted edges is the same as maximizing the weight of the matching. Hence, we can compute an optimal solution for the triangle-free Weighted Cluster Deletion instance by computing a maximum-weight matching M of G. This computation can be performed in $O(\sqrt{n}m \cdot \log^2 n)$ time [12]. The overall running time is therefore $O(n^{1.5} \cdot \log^2 n)$ since the procedure of replacing the triangles can be performed in $O(n)$ time and $m \leq 2n$. □

Altogether, we arrive at the following.

Theorem 5. Cluster Deletion *can be solved in* $O(n^{1.5} \cdot \log^2 n)$ *time when the input graph has maximum degree three.*

Proof. Given an instance of maximum degree three, we first exhaustively apply Reduction Rule 2 in $O(n)$ time. Then, we exhaustively apply Reduction Rule 1, also in $O(n)$ time. Note that the application of Reduction Rule 1 does not produce any triangle to which Reduction Rule 2 applies. Hence, the instance is reduced with respect to both reduction rules. Consequently, Lemma 4 can be applied; the overall running time follows. □

4. Parameterization by "Number of Clusters and Local Modification Bound"

In the hardness results of Section 2, the number of clusters in the final cluster graph is unbounded. A natural question thus is: how does the number of clusters affect the computational complexity for instances that have a fixed local modification bound t? We answer this question by showing that a constrained version of Cluster Editing is fixed-parameter tractable with respect to the combined parameter "number d of clusters in the target graph and local modification bound t". We choose the following formulation to incorporate d and t into the problem:

> (d,t)-Constrained-Cluster Editing:
> **Input:** An undirected graph $G = (V, E)$, a function $\tau : V \to \{0, \ldots, t\}$, and nonnegative integers d and k.
> **Question:** Can G be transformed into a cluster graph G' by applying at most k edge modifications such that G' has at most d clusters and each vertex $v \in V$ is incident with at most $\tau(v)$ modified edges?

We use τ during our algorithm to keep track of the number of modifications that each vertex has been incident with. We can initially set $\tau(v) := t$ for each $v \in V$ and directly obtain the constraints posed by the local modification bound t. We refer to the corresponding problem in which only edge deletions are allowed as (d,t)-Constrained-Cluster Deletion. Clearly, Cluster Editing is the same as (n,n)-Constrained Cluster Editing where $\tau(v) = n$ for each $v \in V$. To show the fixed-parameter tractability of (d,t)-Constrained-Cluster Editing and (d,t)-Constrained-Cluster Deletion with respect to the combined parameter (d,t), we

present a set of polynomial-time data reduction rules. Before doing so, we discuss several aspects of the problem formulation and parameterization.

Concerning the problem formulation, in many application scenarios a reasonable upper bound for the number of clusters d is given in advance. Furthermore, the local modification bound t yields another measure of closeness of the cluster graph to the input graph. In comparison to Cluster Editing, (d,t)-Constrained-Cluster Editing thus allows to further constrain the solution by adjusting the values of d and t. In certain application scenarios this may help to obtain better clusterings. In this sense, (d,t)-Constrained-Cluster Editing directly corresponds to a multi-criteria optimization problem where there is a trade-off between finding solutions that have small values of d, t, or k.

Concerning the parameterization, one can observe that for some instances k is not bounded by a function in d and t. Consider for example a graph $G = (V, E)$ that consists of two cliques K_1 and K_2, each of order $|V|/2$. Furthermore, let each $v \in K_1$ have exactly one neighbor in K_2 and vice versa. An optimal solution for this graph is to delete all $|V|/2$ edges between K_1 and K_2. Hence, the parameter k is very large for this instance, whereas $d = 2$ and $t = 1$. In general, we can always assume $t \leq k$. The general relation between d and k is a bit more tricky. For example, in case G is connected, we can assume $d \leq k + 1$ since applying k edge modifications to G produces at most $k + 1$ connected components. Furthermore, in case G does not contain isolated cliques, we can assume $d \leq 2k$, since at least one edge modification is incident with each clique in the final cluster graph. In most application scenarios, the connected components of the input graph are processed independently from each other. Hence, we usually have $d \leq k + 1$ for real-world instances. In summary, the parameters d and t can be arbitrarily small compared to k, are bounded from above by a linear function of k when G does not contain isolated cliques, and are usually smaller than k for real-world instances.

We now show that (d,t)-Constrained-Cluster Editing is fixed-parameter tractable with respect to (d,t). More precisely, we present four data reduction rules for (d,t)-Constrained-Cluster Editing that produce a problem kernel consisting of at most $4dt$ vertices. The first two rules identify edge modifications that have to be performed by every solution, since otherwise there would be vertices to which more than t edge modifications are incident.

Reduction Rule 3. *If G contains two adjacent vertices $u, v \in V$ such that $|N(u) \setminus N[v]| > 2t$, then remove $\{u, v\}$ from E and set $\tau(v) \leftarrow \tau(v) - 1$, $\tau(u) \leftarrow \tau(u) - 1$, and $k \leftarrow k - 1$.*

Reduction Rule 4. *If G contains two nonadjacent vertices $u, v \in V$ such that $|N(u) \cap N(v)| > 2t$, then add $\{u, v\}$ to E and set $\tau(v) \leftarrow \tau(v) - 1$, $\tau(u) \leftarrow \tau(u) - 1$, and $k \leftarrow k - 1$.*

Lemma 5. *Reduction Rules 3 and 4 are correct and can be exhaustively performed in $O(n^3)$ time.*

Proof. Let $(G = (V, E), d, t, k)$ be an input instance of (d, t)-Constrained-Cluster Editing. We show the correctness of each rule and then bound the running time of exhaustively applying both rules.

Let u and v be as described in Reduction Rule 3. We show that every locally t-bounded solution deletes the edge $\{u, v\}$. Suppose that there is a locally t-bounded solution S that does not delete $\{u, v\}$, let G' be the cluster graph that results from applying S to G, and let K be the cluster of G' such that $u, v \in K$. Clearly, $|K \cap N(u) \setminus N[v]| \leq t$ since at most t inserted edges are incident with v. Then, however, more than t deleted edges are incident with u. This contradicts that S is a solution.

Let u and v be as described in Reduction Rule 4. We show that every solution adds the edge $\{u, v\}$. Suppose that there is some solution S that does not add $\{u, v\}$, let G' be the cluster graph that results from applying S to G, and let K be the cluster of G' such that $u \in K$ and $v \notin K$. Since at most t deleted edges are incident with u, we have $|N(u) \cap N(v) \cap K| > t$. Then, however more than t deleted edges are incident with v. This contradicts that S is a solution.

To achieve a running time of $O(n^3)$ we proceed as follows. First, we initialize for each pair of vertices $u, v \in V$ three counters, one counter that counts $|N(u) \cap N(v)|$, one counting $|N(u) \setminus N[v]|$, and one counting $|N(v) \setminus N[u]|$. For each such pair, this is doable in $O(n)$ time when an adjacency matrix has been constructed in advance. Hence, the overall time for initializing the counters for all possible vertex pairs is $O(n^3)$. All counters that warrant an application of either Reduction Rule 3 or Reduction Rule 4 are stored in a list. We call these counters *active*. Next, we apply the reduction rules. Overall, since $k \leq n^2$ the rules can be applied at most n^2 times. As long as the list of active counters is nonempty, we perform the appropriate rule for the first active counter of the list. It remains to update

all counters according to the edge modification applied by the rule. Suppose Reduction Rule 4 applies to u and v, that is, $\{u, v\}$ is added. Then, we have to update the counters for each pair containing v or u. For v, this can be done in $O(n)$ time, by checking for each $w \neq v$, whether u must be added to $N(v) \cap N(w)$ or added to $N(v) \setminus N[w]$ or removed from $N(w) \setminus N[v]$ (for each counter this can be done in $O(1)$ time by using the constructed adjacency matrix). For each updated counter, we also check in $O(1)$ time whether it needs to be added to/removed from the list of active counters. The case that Reduction Rule 3 applies to u and v can be shown analogously. Overall, we need $O(n^3)$ time to initialize the counters and $O(n^3)$ time for the exhaustive application of the rules. □

The following reduction rule simply checks whether the instance contains vertices to which already more than t modifications have been applied. Clearly, in this case the instance is a no-instance.

Reduction Rule 5. *If there is a vertex $v \in V$ with $\tau(v) < 0$, then output "no".*

The final reduction rule identifies isolated cliques that cannot be merged or split, and whose removal thus does not destroy solutions of (d, t)-Constrained-Cluster Editing.

Reduction Rule 6. *If there is an isolated clique K in G such that $|K| > 2t$, then remove K from G and set $d := d - 1$.*

Lemma 6. *Reduction Rule 6 is correct and can be exhaustively performed in $O(m)$ time.*

Proof. The running time of the rule is obvious; for the correctness we show that K is a cluster of any cluster graph that can be obtained by a locally t-bounded solution.

Since $|K| > 2t$, there is at least one vertex that is adjacent to at least t vertices of K in any cluster graph that can be obtained by a locally t-bounded solution. Hence, there is a cluster K' of size at least $t + 1$ that contains only vertices from K. Since every vertex from K that is not part of K' is incident with at least $t + 1$ edge deletions, we have $K \subseteq K'$. Furthermore, we have $K' = K$ since adding a vertex $v \in V \setminus K$ to K causes at least $2k$ edge insertions that are incident with v. □

We now show that applying Reduction Rules 3–6 yields a problem kernel.

Theorem 6. (d,t)-Constrained-Cluster Editing *admits a* $4dt$*-vertex problem kernel which can be found in* $O(n^3)$ *time. It is thus fixed-parameter tractable with respect to the parameter* (d,t).

Proof. We first show the problem kernel size and then bound the running time of the kernelization.

Let $(G = (V,E), d, t, k)$ be an input instance of (d,t)-Constrained-Cluster Editing and let G be reduced with respect to Reduction Rules 3–6. We show the following:

$$(G,d,t,k) \text{ is a yes-instance} \Rightarrow G \text{ has at most } 4dt \text{ vertices.}$$

Let S be a solution of the input instance and let G' be the cluster graph that results from applying S to G. We show that every cluster K_i of G' has at most $4t$ vertices. Assume toward a contradiction that there is some K_i in G' with $|K_i| > 4t$. Since G is reduced with respect to Reduction Rule 6, there must be either an edge $\{u,v\}$ in G such that $u \in K_i$ and $v \in V \setminus K_i$ or a pair of vertices $u,v \in K_i$ such that $\{u,v\}$ is not an edge in G.

Case 1: $u \in K_i$**,** $v \in V \setminus K_i$ **and** $\{u,v\} \in E$**.** Since at most $t-1$ edge insertions are incident with u, it has in G at least $3t+1$ neighbors in K_i. Furthermore, since at most t edge deletions are incident with v, it has in G at most t neighbors in K_i. Hence, there are at least $2t+1$ vertices in K_i that are neighbors of u but not neighbors of v. Therefore, Reduction Rule 3 applies in G, a contradiction to the fact that G is reduced with respect to this rule.

Case 2: $u,v \in K_i$ **and** $\{u,v\} \notin E$**.** Both u and v are in G adjacent to at least $|K_i| - (t-1)$ vertices of $K_i \setminus \{u,v\}$. Since $|K_i| > 4t$ they thus have in G at least $2t+1$ common neighbors. Therefore, Reduction Rule 4 applies in G, a contradiction to the fact that G is reduced with respect to this rule.

We have shown that $|K_i| \leq 4t$ for each cluster K_i of G'. Since G' has at most d clusters, the overall bound on the number of vertices follows.

It remains to bound the running time of of obtaining an instance that is reduced with respect to Reduction Rules 3–6. By Lemma 5, the exhaustive application of Reduction Rules 3 and 4 runs in $O(n^3)$ time. After these two rules have been exhaustively applied, Reduction Rules 5 and 6 can be exhaustively applied in $O(m)$ time. Finally, observe that applying Reduction Rules 5 and 6 does not lead to an instance to which Reduction Rules 3 and 4 can be applied again. □

The data reduction rules can be adapted to the case that only edge deletions are allowed. Indeed, we can show a $2dt$-vertex problem kernel for (d,t)-Constrained-Cluster Deletion by replacing $2t$ by t in Reduction Rule 3 (note that Reduction Rule 4 is not suitable for Cluster Deletion since it adds an edge). More precisely, we have the following two reduction rules specifically for Cluster Deletion.

Reduction Rule 7. *If G contains two adjacent vertices $u, v \in V$ such that $|N(u) \setminus N[v]| > t$, then remove $\{u, v\}$ from E and set $\tau(v) := \tau(v) - 1$, $\tau(u) := \tau(u) - 1$, and $k := k - 1$.*

Lemma 7. *Reduction Rule 7 is correct and can be exhaustively applied in $O(n^3)$ time.*

Proof. The running time was already shown in the proof of Lemma 5. Hence, we only show the correctness of the rule.

Every locally t-bounded solution deletes at most t edges incident with u. Hence, in the cluster graph that results from applying such a solution, u has at least one neighbor $w \notin N[v]$. Hence, the solution must also delete $\{u, v\}$. Otherwise the graph is not a cluster graph. □

The second rule deals with isolated clusters in G.

Reduction Rule 8. *If there is an isolated clique K in G, then remove K from G and set $d := d - 1$.*

The correctness of the rule follows from the observation that this isolated clique produces at least one cluster. Finally, we also apply Reduction Rule 5 in order to find vertices to which too many edge modifications have been applied. Altogether, the exhaustive application of these rules yields a $2dt$-vertex problem kernel, as we show in the following.

Theorem 7. (d,t)-Constrained-Cluster Deletion *admits a $2dt$-vertex problem kernel which can be found in $O(n^3)$ time. It is thus fixed-parameter tractable with respect to the parameter (d,t).*

Proof. The proof works in complete analogy to the proof of Theorem 6, the only difference is that we can show that every cluster of the cluster graph has at most $2t$ vertices instead of $4t$ vertices.

Let G be a graph that is reduced with respect to Rules 7, 8, and 5. We show that each cluster of every cluster graph that can be obtained by a locally

t-bounded solution has size at most $2t$. Assume toward a contradiction that there is such a cluster graph that contains a cluster K that has more than $2t$ vertices. Since G is reduced with respect to Reduction Rule 8, there must be a pair of vertices $u \in K$ and $v \in V \setminus K$ such that $\{u, v\}$ is an edge in G. Since the solution is locally t-bounded, v has in G at most t neighbors in K. Hence, u has in G more than t neighbors that are not neighbors of v. Therefore, Reduction Rule 7 applies, a contradiction to the assumption that G is reduced. □

5. Concluding Remarks

The presented hardness and tractability results provide a more detailed view on the computational complexity of Cluster Editing and Cluster Deletion. Several open questions and research tasks concerning Cluster Editing arise immediately from these results.

For instance concerning the NP-hardness of Cluster Editing for graphs with bounded degree, achieving a complexity-dichotomy, as we now have for Cluster Deletion, would be desirable. We conjecture that Cluster Editing on graphs with maximum degree three is solvable in polynomial time. For graphs with maximum degree four, we have no conjecture at the moment. For graphs with maximum degree five, the NP-hardness appears to follow from a recent result by Fomin et al. [11].

Concerning the parameter "local modification bound t" several questions arise. For example, is Cluster Editing polynomial-time solvable when the solution is locally 1-bounded? Another question is whether there are other graph modification problems for which this parameter yields fixed-parameter tractability? A good candidate seems to be the Feedback Arc Set in Tournaments problem, which appears to be "easier" than Cluster Editing.[4] Concerning the combined parameter "number d of clusters and local modification bound t", developing a search tree algorithm would complement our problem kernelization results. Moreover, experimental studies should be performed to analyze what typical values of d and t are in real-world instances, and to determine whether adding our data reduction rules provides a speed-up for some instances. Finally, further suitable parameterizations of Cluster Editing should be explored. These could be structural

[4]For example, Feedback Arc Set in Tournaments can be solved in time that is subexponential in the size of the solution [1, 16].

graph parameters but also parameters that are related to the solution such as for example the parameter "number of edge deletions performed by the solution". This parameter could be considerably smaller than the parameter number of edge modifications.

References

[1] N. Alon, D. Lokshtanov, and S. Saurabh. Fast FAST. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming (ICALP '09), Part 1*, volume 5555 of *Lecture Notes in Computer Science*, pages 49–58. Springer, 2009.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1–3):89–113, 2004.

[3] S. Böcker. A golden ratio parameterized algorithm for cluster editing. *Journal of Discrete Algorithms*, 2012. To appear, electronically available.

[4] S. Böcker and P. Damaschke. Even faster parameterized cluster deletion and cluster editing. *Information Processing Letters*, 111(14):717–721, 2011.

[5] S. Böcker, S. Briesemeister, Q. B. A. Bui, and A. Truß. Going weighted: Parameterized algorithms for cluster editing. *Theoretical Computer Science*, 410(52):5467–5480, 2009.

[6] S. Böcker, S. Briesemeister, and G. W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2011.

[7] Y. Cao and J. Chen. Cluster editing: Kernelization based on edge cuts. In *Proceedings of the 5th International Symposium on Parameterized and Exact Computation (IPEC '10)*, volume 6478 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2010.

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

[9] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[10] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[11] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger. Subexponential fixed-parameter tractability of cluster editing. *CoRR*, abs/1112.4419, 2011.

[12] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for general graph-matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[13] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Graph-modeled data clustering: Exact algorithms for clique generation. *Theory of Computing Systems*, 38(4):373–392, 2005.

[14] J. Guo. A more effective linear kernelization for cluster editing. *Theoretical Computer Science*, 410(8–10):718–726, 2009.

[15] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63 (4):512–530, 2001.

[16] M. Karpinski and W. Schudy. Faster algorithms for feedback arc set tournament, Kemeny rank aggregation and betweenness tournament. In *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC '10), Part 1*, volume 6506 of *Lecture Notes in Computer Science*, pages 3–14, 2010.

[17] C. Komusiewicz. *Parameterized Algorithmics for Network Analysis: Clustering & Querying*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2011.

[18] C. Komusiewicz and J. Uhlmann. Alternative parameterizations for cluster editing. In *Proceedings of the 37th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '11)*, volume 6543 of *Lecture Notes in Computer Science*, pages 344–355. Springer, 2011.

[19] M. Křivánek and J. Morávek. NP-hard problems in hierarchical-tree clustering. *Acta Informatica*, 23(3):311–323, 1986.

[20] D. Lokshtanov, D. Marx, and S. Saurabh. Lower bounds based on the exponential time hypothesis. *Bulletin of the EATCS*, 105:41–72, 2011.

[21] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[22] R. Shamir, R. Sharan, and D. Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1–2):173–182, 2004.

[23] J. Uhlmann. *Multivariate Algorithmics in Biological Data Analysis*. PhD thesis, Technische Universität Berlin, Berlin, Germany, 2011.

[24] J. M. M. van Rooij, M. E. van Kooten Niekerk, and H. L. Bodlaender. Partition into triangles on bounded degree graphs. In *Proceedings of the 37th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM '11)*, volume 6543 of *Lecture Notes in Computer Science*, pages 558–569. Springer, 2011.

[25] M. Weller, C. Komusiewicz, R. Niedermeier, and J. Uhlmann. On making directed graphs transitive. *Journal of Computer and System Sciences*, 78(2):559–574, 2012.

[26] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Combinatorial Optimization*, volume 2570 of *Lecture Notes in Computer Science*, pages 185–208. Springer, 2003.