The Multi-parameterized Cluster Editing Problem

Faisal N. Abu-Khzam

Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
faisal.abukhzam@lau.edu.lb

Abstract. The Cluster Editing problem seeks a transformation of a given undirected graph into a transitive graph via a minimum number of edge-edit operations. Existing algorithms often exhibit slow performance and could deliver clusters of no practical significance, such as singletons. A constrained version of Cluster Editing is introduced, featuring more input parameters that set a lower bound on the size of a clique-cluster as well as upper bounds on the amount of both edge-additions and deletions per vertex. The new formulation allows us to solve Cluster Editing (exactly) in polynomial time when edge-edit operations per vertex is smaller than half the minimum cluster size. Moreover, we address the case where the new edge addition and deletion bounds (per vertex) are small constants. We show that Cluster Editing has a linear-size kernel in this case.

1 Introduction

Given an undirected loopless graph $G = (V, E)$ and an integer $k > 0$, the Cluster Editing Problem asks whether k or less edge additions or deletions can transform G into a graph whose connected components are cliques. Cluster Editing is NP-Complete [13, 16], but can be solved in polynomial-time when the parameter k is fixed [4, 10]. In other words, the problem is fixed-parameter tractable when parameterized by the total number of edge-edit operations[1].

The Cluster Editing problem received considerable attention recently. This can be seen from a long sequence of continuous algorithmic improvements (see [1–3, 5, 6, 10, 11]). The current asymptotically fastest fixed-parameter algorithm runs in $O^*(1.618^k)$ [2]. Moreover, a kernel of order $2k$ was achieved recently in [6]. In other words, an arbitrary Cluster Edit instance can be reduced in polynomial-time into an equivalent instance where the number of vertices is at most $2k$.

In application domains, Cluster Editing is also known under the name "Correlation Clustering" where it can be viewed as a model for accurate unsupervised clustering. In such context, edges to be deleted/added from a given instance are

[1] We assume familiarity with the notion of fixed-parameter tractability and kernelization algorithms (see [7, 8, 15]).

considered false positives/negatives. Such errors could be small in some practical applications, and they tend to be even smaller per input object, or vertex, unless the said object is an outlier. Moreover, in some applications clusters are not expected to be too small or differ significantly in size.

Motivated by the above, we consider a parameterized version of Cluster Editing, dubbed Constrained Cluster Editing, which assumes a cluster is bounded below by some application-specific threshold and both the number of edges that can be deleted and the number of edges that can be added, per vertex, are also bounded. We refer to these two bounds by *error parameters*. We introduced these parameters in [9] where an experimental study was conducted on their effect on the running time of branching algorithms. Similar work appeared recently in [12] where the total number of edge-edit operations, per vertex, and the number of clusters in the target solution are used as additional parameters. We note here that setting separate bounds on the two parameters could affect the complexity as well as algorithmic solutions of the problem. Moreover, according to computational biologists [14], the expected frequency of false positives and false negatives differ in many applications.

We present a polynomial time algorithm that solves the optimization version of Cluster Edit whenever the sum of the two error parameters is smaller than half the expected minimum cluster size in the target solution. This is a rather common case whenever the clusters are of significant minimum size and error parameters are not expected to be high, being often due to noise.

We shall discuss the complexity of Constrained Cluster Editing when the error-parameters are small constants. In particular, we show that Minimum Cluster Editing is solvable in polynomial-time when at most one edge can be deleted and at most one edge can be added per vertex. Moreover, in the case where error parameters are small constants, we show that our simple reduction procedure leads to a problem kernel whose size is linear in the parameter k.

The paper is organized as follows: section 2 presents some preliminaries; section 3 is devoted to the main reduction procedure; our main complexity result is presented in section 4; in section 5 we study the optimization version of Cluster Editing when the error parameters are small constants, and section 6 concludes with a summary.

2 Preliminaries

We adopt common graph theoretic terminologies, such as neighborhood, degrees, adjacency, path, etc. The term non-edge is used to designate a pair of non-adjacent vertices. Given a graph $G = (V, E)$, and a set $S \subset V$, the subgraph induced by S is denoted by $G[S]$. A clique in a graph is a subgraph induced by a set of pair-wise adjacent vertices. An edge-edit operation is either a deletion or an addition of an edge. We shall use the term *cluster graph* to denote a transitive graph, which consists of a disjoint union of cliques, as connected components.

For a given graph G and parameter k, the Parameterized Cluster Editing problem asks whether G can be transformed into a cluster graph via k or less edge-edit operations. In this paper, we consider a parameterized version of this problem, called *Constrained Cluster Editing*, which can be formally defined as follows.

Input: A graph G, parameters k, s, a, d, and two functions $a : V(G) \rightarrow \{0, 1, \cdots, a\}$, $d : V(G) \rightarrow \{0, 1, \cdots, d\}$.
Question: Can G be transformed into a disjoint union of cliques by at most k edge-edit operations such that each clique is of size s or more and, for each vertex $v \in V(G)$, the number of added (deleted) edges incident on v is at most $a(v)$ ($d(v)$ respectively)?

We shall use some special terminology to help us present our algorithm. The expression *solution graph* may be used instead of cluster graph, when dealing with a specific input instance. Edges that are not allowed to be in the cluster graph are called *forbidden* edges, while edges that are (decided to be) in the solution graph are *permanent*. An induced path of length two is called a *conflict triple*, which is so named because it can never be part of a solution graph. To *cliquify* a set S of vertices is to transform $G[S]$ into a clique by adding edges.

A clique is permanent if each of its edges are permanent. To *join* a vertex v to clique C is to add all edges between v and vertices of C that are not in $N(v)$. This operation makes sense only when C is permanent or when turning C to a permanent clique. If v already contains C in its neighborhood, then *joining* v to C is equivalent to making $C \cup \{v\}$ a permanent clique. To *detach* v from C is the opposite operation (of deleting all edges between v and the vertices of C).

The first, and simplest, algorithm for Cluster Editing finds a conflict triple in the input graph and "resolves" it by exploring two cases: either delete one of the two edges in the path or insert the missing edge, In both cases, the algorithm proceeds recursively. As such, the said algorithm runs in $O(3^k)$ (3 cases per conflict triple). The same idea has been used in almost all subsequent algorithms, which added more sophisticated "branching rules."

A kernelization algorithm, with respect to an input parameter k, is a polynomial-time reduction procedure that yields an equivalent problem instance whose size, or order, is bounded by a function of the input parameter. Known kernelization algorithms for Cluster Editing have so far obtained kernels whose order (number of vertices only) is bounded by a linear function of k [5, 6, 11]. The most recent order-bound is $2k$ [6]. We shall present a simple reduction procedure that delivers kernels whose number of edges is bounded by $2(a+3d)^2k$. The same reduction procedure is used to obtain our main result, which states that Minimum Cluster Editing is solved exactly in polynomial time when $s > 2(a+d)$.

When $s = 1$, the bound on a cluster size is not important. The corresponding version of the problem can be denoted by (a, d)-Cluster Editing. This version is different from the one introduced in [12] where a bound c is placed on the total number of edge-edit operations per vertex. We refer to this problem as c-Cluster Editing. When $c > 3$, c-Cluster Editing is NP-hard (shown also in [12]).

Note that 4-Cluster Editing is not equivalent to (a, d)-Cluster Editing where $a + d = 4$. According to our definition, $a + d = 4$ means that that (a, d) is one of the five elements of $\{(i, j) : i + j = 4\}$. Therefore, the NP-hardness of c-Cluster Editing for $c \geq 4$ does not imply that (a, d)-Cluster Editing is NP-hard when $a + d = 4$. Note that, for any a, $(a, 0)$-Cluster Editing is solvable in polynomial time: any solution must add all edges to get rid of all conflict-triples, if possible.

We note here that if (a, d)-Cluster Editing is NP-hard then so is (a', d')-Cluster Edit for all $a' \geq a$ and $d' \geq d$. This follows immediately from the definition since every instance of the first is an instance of the second. We conjecture that $(a, 1)$-Cluster Editing is NP-hard for $a > 1$, and we prove in section 5 that $(1, 1)$-Cluster Editing is solvable in polynomial-time when the cluster size is at least four. We pose as open problem whether $(2, 1)$-Cluster Editing is NP-hard.

Finally, as an additional practical objective, our reduction procedure assumes that no edge can be added between two different connected components. This can be motivated by the common practice in social networks to (try to) connect people who have mutual friends/connections. This constraint may be made stronger by requiring that two vertices with disjoint neighborhoods must belong to different clusters. The latter constraint is not studied in this paper.

3 A Reduction Procedure

In general, a problem-reduction procedure is based on reduction rules, each of the form $\langle$*condition*, *action*$\rangle$, where *action* is an operation that can be performed to obtain an equivalent instance of the problem whenever *condition* holds. If a reduction is not possible, or the *action* violates a problem-specific constraint, then we have a no instance. Moreover, a reduction rule is sound if its action results in an equivalent instance.

Let (G, k) be an instance of (a, d, s, k)-Cluster Editing. In what follows, a set is cliquifiable if it can be made a clique by adding all pairs of missing edges without violating any of the constraints.

The main reduction rules are given below. They are assumed to be applied successively in such a way that a rule is not applied, or re-applied, until all the previous rules have been applied exhaustively. We shall prove the soundness of non-obvious reduction rules only.

3.1 Base-Case Reductions

Reduction Rule 1. *The reduction algorithm terminates and reports a no instance, whenever any of $k, a(v)$, or $d(v)$ becomes negative for some vertex $v \in V(G)$.*

Reduction Rule 2. *For any vertex v, if $d(v) = 0$ (or becomes zero), then $N(v)$ is cliquified.*

Note that applying Rule 2 may yield negative parameters, which triggers Rule 1 and causes the algorithm to terminate with a negative answer.

Reduction Rule 3. *If $a(u) = 0$, then set every non-edge of u to forbidden.*

Reduction Rule 4. *If $a(u) = d(u) = 0$, then delete $N[u]$. This results in deleting all edges connecting $N(u)$ to $V(G) \setminus N(u)$ and decrementing all the corresponding parameters.*

Soundness: In this case, $N[u]$ is cliquified by Rule 2. If the algorithm does not terminate, and since no new neighbors can be added to u, $N[u]$ is a cluster.

3.2 Reductions Based on Conflict-Triples

Reduction Rule 5. *If uv and uw are permanent edges and vw is a non-edge, then add vw and decrement each of k, $a(v)$ and $a(w)$ by one. If vw is a non-permanent edge, then set vw as permanent.*

Reduction Rule 6. *If uv is a permanent edge and uw is a forbidden edge, then set vw as forbidden. If vw exists, delete it and decrement k, $d(v)$ and $d(w)$ by one.*

3.3 Reductions Based on Common Neighbors

Reduction Rule 7. *If two non-adjacent vertices u and v have more than $d(u)+d(v)$ common neighbors (or just $> 2d$ common neighbors), then add edge uv and decrement each of k, $a(u)$ and $a(v)$ by one.*

Soundness: If u and v are not in the same clique in the solution graph, then at least one of them has to lose more than d edges, which is not possible.

Reduction Rule 8. *If two adjacent vertices, u and v, have at least $2d$ common neighbors then set uv as permanent edge.*

Soundness: Same argument as in Rule 7 above.

Reduction Rule 9. *If two vertices u and v are such that $|N(u) \setminus N(v)| > a + d$ then set edge uv as forbidden. If u and v are adjacent, then delete uv and decrement each of k, $d(u)$ and $d(v)$ by one.*

Soundness: For u and v to be in the same cluster, at most d neighbors may be deleted from $N(u)$ and at most a neighbors can be added to $N(v)$.

3.4 Reductions Based on Cluster-Size

Reduction Rule 10. *If $s > 1$ and there is a vertex v satisfying: $0 \leq degree(v) < s - a(v) - 1$, then return No.*

Soundness: Obviously, v needs more than $a(v)$ edges to be a member of a cluster in a solution graph.

Reduction Rule 11. *If $s > 1$ and there is a vertex v satisfying: $0 \leq degree(v) - d(v) < s - a(v) - 1$, then set $d(v) = degree(v) - s + a + 1$.*

Soundness: If $d(v)$ edges incident on v are deleted, we get a no-instance by Rule 10.

Reduction Rule 12. *If $s > 2$ and two non-adjacent vertices u and v have less than $s - 2a$ common neighbors, then set edge uv as forbidden.*

Soundness: For u and v to belong to the same cluster, they must have at least $s - 2$ common neighbors. After adding uv, the maximum number of common neighbors we can add is $2a - 2$ ($a - 1$ edges between u and $N(v)$ and vice versa). The total number of common neighbors after adding all possible edges remains less than $s - 2 (= s - 2a + 2a - 2)$.

Reduction Rule 13. *If $s > 2$ and two adjacent vertices u and v have $< s - 2a - 2$ common neighbors, then delete edge uv.*

Soundness: The argument is similar to the previous case, except that each vertex must add at least a neighbors of the other to obtain $s - 2$ common neighbors.

3.5 Permanent and Isolated Cliques

Deleting all isolated cliques is a sound reduction rule for the general Cluster Editing problem. In our formulation we do not allow a cluster to be of size $< s$. Therefore, and since we do not allow the addition of edges between different connected components, we shall assume that an isolated clique of size $< s$ yields a no instance.

Reduction Rule 14. *If a clique C is such that $N(C) \setminus C = \emptyset$ and $|C| < s$ then we have a no-instance.*

Reduction Rule 15. *If a clique C is such that $N(C) \setminus C = \emptyset$, and $|C| \geq s$, then delete C.*

Soundness: This follows from our assumption that no edges are to be added between different connected components.

Finally, the presence of permanent cliques can yield problem reductions that are not obtained by exhaustive applications of the above reduction rules. Note that a permanent edge is a special case of a permanent clique.

Reduction Rule 16. *If a vertex v has more than d neighbors in a permanent clique C, then v is joined to C.*

Reduction Rule 17. *Let C be a permanent clique of size $> a$. If a vertex v has less than $|C| - a$ neighbors in C, then v is detached from C.*

4 Complexity of Constrained Cluster Editing

An instance (G, a, d, s, k) of Constrained Cluster Editing is said to be reduced if the above reduction rules have been exhaustively applied to the input graph G. The following key Lemma follows from the reduction procedure.

Lemma 1. *Let (G, a, d, s, k) be a reduced yes-instance of Constrained Cluster Editing. Then every vertex of G has at most $a + 3d$ neighbors.*

Proof. Assume there is a vertex v such that $|N(v)| > a + 3d$. By Rule 7, any vertex u is either a neighbor of v or has at most $2d$ common neighbors with v. In the latter case, v has more than $a + d$ vertices that are not common with u. Edge uv would then be forbidden by Rule 9. By Rules 8 and 9, every edge incident on v is either deleted or becomes permanent. Applying Rules 5 and 6 (exhaustively) leads to cliquifying and isolating $N[v]$, which then results in deleting $N[v]$.

We now consider the optimization version of Cluster Editing, which seeks a minimum number of edge-edit operations. So k is not a parameter in this case, but we keep the three constraints a, d and s.

Theorem 1. *When $s > 2(a + d)$, and $ad > 0$, the Minimum Cluster Editing problem is solvable in polynomial time.*

Proof. By Rules 7 and 12, any two vertices u, v of a reduced instance that are at distance two from each other satisfy: $s - 2a \leq |N(u) \cap N(v)| \leq 2d$. When $s - 2a > 2d$, and provided the reduction procedure does not detect a no instance, the reduction rules will automatically lead to a P_2-free graph since no two non-adjacent vertices can have common neighbors.

In practice, the total number of errors per data element is expected to be small and should be much smaller than a cluster size. In the seemingly common case where the cluster size is greater than two times such error, our theorem asserts that optimum clustering is solvable in polynomial time.

When $s \leq 2(a + d)$, the Minimum Constrained Cluster Editing problem remains NP-hard. In this case, the reduction procedure may still help to obtain faster parameterized algorithms. Moreover, when the error-parameters a and d are small constants and the size of a cluster is not important (i.e., $s = 1$), applying the above reduction rules yields equivalent instances whose size is bounded by a linear function of the main parameter k.

Theorem 2. *There is a polynomial-time reduction algorithm that takes an arbitrary instance of Constrained Cluster Editing and either determines that no solution exists or produces an equivalent instance whose order is bounded by $2k(a + 3d + 1)$.*

Proof. Let (G, a, d, k) be a reduced instance of Cluster Editing. By Lemma 1, each remaining vertex in a reduced instance has degree $\leq a + 3d$. Let A be the set of vertices of G that are incident to an edge that must be deleted or a non-edge that must be added to obtain a minimum solution. If (G, a, d, k) is a yes instance, then $|A| \leq 2k$. Let $B = N(A)$ and $C = V(G) \setminus (A \cup B)$. Observe that any member of a conflict triple is either in A or in B. It follows that C must be empty since any isolated clique is deleted in a reduced instance. The proof follows from the fact that $|A| \leq 2k$ and every vertex of A has at most $a + 3d$ neighbors in B.

The following Corollary follows easily from Theorem 2 and Lemma 1.

Corollary 1. *There is a polynomial-time reduction algorithm that takes an arbitrary instance of Constrained Cluster Editing and either determines that no solution exists or produces an equivalent instance whose size is bounded by* $2k(a+3d)^2$.

5 The (a, d, s)-Cluster Editing Problem

In [9], a Cluster Editing algorithm was modified so that it solves $(a, d, 1, k)$-Cluster Editing. Experiments show that using the add and delete parameters improves the running time on yes instances, despite the fact that a few combinations of values of a and d were tried on each instance before finding a solution. It was observed that solutions were always found with small values of a and d. Motivated by these experiments, we discuss the complexity of the optimization version of Cluster Editing when a, d and s are small constants. We shall refer to this optimization version of the problem as (a, d, s)-Cluster Editing.

It was shown in [12] that Cluster Editing is NP-hard when the total number of edge-edit operations per vertex is four or more. Unfortunately, the result (and proof) of [12] cannot be used in studying the complexity of each of the separate cases where $a + d = 4$ ($(a, d) \in \{(0, 4), (1, 3), (2, 2), (3, 1)\}$). It was also shown in [12] that Cluster Editing is NP-hard when $a = 0$ and $d = 2$. This implies the NP-hardness of $(a, 2, s)$-Cluster Editing for $a \geq 0$ and $s \geq 1$. Motivated by these results, and by the need for non-trivial clusters, we show that our reduction procedure solves $(1, 1, s)$-Cluster Editing in polynomial time. We shall address the case $s \geq 4$ in the rest of this paper [2].

Lemma 2. *A reduced instance of* $(1, 1, s)$*-Cluster Editing is triangle free.*

Proof. If the reduced instance has a triangle $T = \{u, v, w\}$ that is not contained in (or does not form) a clique-cluster, then we distinguish two cases:

(i) Some vertex x of G has exactly two neighbors, say v and w, in T. This triggers Rule 8, which forces edge vw to become permanent. Then Rule 16 applies to both u and x, forcing $\{u, v, w, x\}$ to become a clique in the solution graph (if possible).

(ii) Some vertex of G has exactly one neighbor in T, which triggers Rule 17 and deletes the edge between the said vertex and T.

[2] It can be shown that the problem is solvable in polynomial-time for any $s > 0$, but the proof is too long to include in this paper.

Lemma 3. *In a reduced yes instance of* $(1,1,s)$*-Cluster Editing, the maximum degree is bounded above by three.*

Proof. By Lemma 1 every vertex is of degree ≤ 4. Let v be a vertex of degree 4. Since at most one incident edge can be deleted, v must belong to a cluster that contains at least three of its neighbors. Since G is triangle-free, two edges would be needed to connect each such neighbor to the other two, which is impossible.

If $s > 4$, and by Theorem 1, the problem is completely solved by the reduction procedure since $s > 4 = 2(a+d)$. When $s = 4$, any yes-instance cannot contain a vertex of degree one. Moreover, edges incident on a degree-two vertex cannot be deleted. Therefore neighbors of a degree-two vertex must be in the same cluster. This is guaranteed by Rule 11, which sets $d(v)$ to sero for any degree-two vertex. This results in introducing triangles and triggering other rules (as described above) that yield automatic isolation (and removal) of clusters whenever possible. The remaining graph must be 3-regular.

The following reduction rule is specific to reduced (triangle-free) instances of $(1,1,s)$-Cluster Editing.

Reduction Rule 18. *If* u *and* v *are adjacent degree-three vertices of a reduced* $(1,1,s)$*-Cluster Editing instance, and none of the neighbors of* u *is adjacent to a neighbor of* v *(i.e.,* $N(N(u)) \cap N(v) = \emptyset$*), then delete edge* uv*.*

Soundness: If uv is permanent, and since $d = 1$, at least one neighbor of each is in the same cluster of u and v. This requires adding two edges to a neighbor of each, which is impossible.

Lemma 4. *In a reduced yes instance of* $(1,1,4)$*-Cluster Editing, if two vertices have a common neighbor then they must belong to a cycle of length four.*

Proof. Every vertex of the reduced instance is of degree-three. If two adjacent vertices are not part of a C_4, then Rule 18 applies.

Since G is triangle-free and 3-regular, exactly one edge must be deleted per vertex. Moreover, every edge of G is part of a cycle of length four. It follows that edges that are not deleted by an optimum solution, if one exists, form a disjoint union of cycles of length four each (the remaining subgraph, before performing any edge addition, is 2-regular). To find an optimum solution, we transform the problem into an instance of *Independent Edge Cover*, which seeks a minimum set of edges S such that every vertex is incident on exactly one element of S. In fact, we just treat a reduced instance as an instance of Edge Cover since each and every vertex has a d value of one. We now obtain the following.

Theorem 3. $(1,1,4)$*-Cluster Editing is solvable in polynomial-time.*

Proof. This follows from the above reduction to Independent Edge Cover, which can be solved in polynomial-time [17].

When $s = 3$, the reduction procedure would also result in a 3-regular graph and the reduction to Independent Edge Cover can also be used. To see this note that deleting a single edge of a connected component results in subsequent reductions that turn it into a cluster graph, unless a no instance is detected. It follows that $(1, 1, 3)$-Cluster Editing is solvable in polynomial-time. We note that the case $s \geq 1$ is also solvable in polynomial-time, via a reduction to Weighted Edge Cover.

6 Conclusion

We studied a multi-parameterized version of the Cluster Editing problem and showed that a simple reduction procedure solves Minimum Cluster Editing in polynomial-time when the smallest acceptable cluster size exceeds twice the allowable edge operations per vertex. Such operations are viewed as errors or false positives/negatives in the input.

On the other hand, when the bound on the edge-edit operations per vertex is constant, we showed that a linear-size kernel is obtained. Previously known kernels have a linear bound on the number of vertices only, and they are not easily applicable to the constrained version.

We observed that (a, d, s, k)-Cluster Editing is NP-hard when $d \geq 2$, and conjectured that $(a, 1, s, k)$-Cluster Editing is NP-hard, in general, for $a \geq 2$. To prove this claim, it would be enough to show the NP-hardness for the case $a = 2$ and $d = 1$.

References

1. Böcker, S., Briesemeister, S., Bui, Q.B.A., Truss, A.: Going weighted: Parameterized algorithms for cluster editing. Theor. Comput. Sci. 410(52), 5467–5480 (2009)
2. Böcker, S.: A golden ratio parameterized algorithm for cluster editing. In: Iliopoulos, C.S., Smyth, W.F. (eds.) IWOCA 2011. LNCS, vol. 7056, pp. 85–95. Springer, Heidelberg (2011)
3. Böcker, S., Briesemeister, S., Klau, G.W.: Exact algorithms for cluster editing: Evaluation and experiments. Algorithmica 60(2), 316–334 (2011)
4. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Inf. Process. Lett. 58(4), 171–176 (1996)
5. Cao, Y., Chen, J.: Cluster editing: Kernelization based on edge cuts. Algorithmica 64(1), 152–169 (2012)
6. Chen, J., Meng, J.: A $2k$ kernel for the cluster editing problem. In: Thai, M.T., Sahni, S. (eds.) COCOON 2010. LNCS, vol. 6196, pp. 459–468. Springer, Heidelberg (2010)
7. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer (1999)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
9. Ghrayeb, A.: Improved search-tree algorithms for the cluster edit problem. MS thesis, Lebanese American University (2011)
10. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. Theor. Comp. Sys. 38(4), 373–392 (2005)

11. Guo, J.: A more effective linear kernelization for cluster editing. Theor. Comput. Sci. 410(8-10), 718–726 (2009)
12. Komusiewicz, C., Uhlmann, J.: Cluster editing with locally bounded modifications. Discrete Applied Mathematics 160(15), 2259–2270 (2012)
13. Kriváanek, M., Morávek, J.: *NP* -hard problems in hierarchical-tree clustering. Acta Inf. 23(3), 311–323 (1986)
14. Langston, M.A.: Private communication (2012)
15. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford University Press (2006)
16. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics 144(1-2), 173–182 (2004)
17. van Rooij, J.M.M., Bodlaender, H.L.: Exact algorithms for edge domination. Algorithmica 64(4), 535–563 (2012)