Journal of Discrete Algorithms 45 (2017) 26–34

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

On the complexity of multi-parameterized cluster editing ☆

Faisal N. Abu-Khzam

Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form 15 May 2017
Accepted 11 July 2017
Available online 19 July 2017

Keywords:
Cluster editing
Fixed-parameter tractability
Multi-parameterization
Kernelization

ABSTRACT

The Cluster Editing problem seeks a transformation of a given undirected graph into a disjoint union of cliques via a minimum number of edge additions or deletions. A multi-parameterized version of the problem is studied, featuring a number of constraints that bound the amounts of both edge-additions and deletions per single vertex, as well as the size of a clique-cluster. We show that the problem remains $\mathcal{NP}$-hard even when only one edge can be deleted and at most two edges can be added per vertex. However, the new formulation allows us to solve Cluster Editing (exactly) in polynomial time when the number of edge-edit operations per vertex is smaller than half the minimum cluster size. In other words, Cluster Editing can be solved efficiently when the number of false positives/negatives per single data element is expected to be small compared to the minimum cluster size. As a byproduct, we obtain a kernelization algorithm that delivers linear-size kernels when the two edge-edit bounds are small constants.

1. Introduction

Given a simple undirected 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 $\mathcal{NP}$-Complete [22,26], but it is fixed-parameter tractable with respect to the parameter k [7,18].[1] The problem received considerable attention recently as can be seen from a long sequence of continuous algorithmic improvements (see [4–6,8, 9,18,19]). The current asymptotically fastest fixed-parameter algorithm runs in $O^*(1.618^k)$ time [4]. Moreover, a kernel of order $2k$ was obtained recently in [9]. This means that an arbitrary Cluster Editing instance can be reduced in polynomial time into an equivalent instance where the number of vertices is at most $2k$. The number of edges in the reduced instance can be quadratic in k.

Cluster Editing can be viewed as a model for accurate unsupervised "correlation clustering." In such context, edges to be deleted from or added to a given instance are considered false positives or false negatives, respectively. Such errors could be small in some practical applications, and they tend to be even smaller per input object, or vertex. In fact, a single data element that is causing too many false positives/negatives might be considered as outlier.

We consider a parameterized version of Cluster Editing where both the number of edges that can be deleted and the number of edges that can be added, per vertex, are bounded by input constraints. We refer to these two bounds by *error parameters*. Similar parameterizations appeared in [20] and [21]. In [20], two parameters p and q were used to bound

☆ A preliminary version of a portion of this paper appeared in the 7th Annual International Conference on Combinatorial Optimization and Applications (COCOA 2013) [1].

E-mail address: faisal.abukhzam@lau.edu.lb.

[1] We assume familiarity with the notions of fixed-parameter tractability and kernelization algorithms [10,16,17,25].

http://dx.doi.org/10.1016/j.jda.2017.07.003

(respectively) (i) the number of edges that can be added between elements of the same cluster and (ii) the number of edges that can be deleted between a cluster and the rest of the graph. Further work on this problem formulation appeared in [24]. In [21], the total number of edge-edit operations, per vertex, and the number of clusters in the target solution are used as additional parameters. Further work on this version appeared recently in [12]. We shall see that setting separate bounds on the two error parameters could affect the complexity of the problem.

We introduce another constraint that bounds, from below, the minimum acceptable cluster size and present a polynomial time algorithm that solves Cluster Editing exactly whenever the sum of error parameters $(a + d)$ is small compared to the minimum cluster size. This condition could be of particular interest in applications where the cluster size is expected to be large or when error parameters per data element are not expected to be high. In this respect, the message conveyed by our work bears the same theme as another, rather experimental, study of various clustering methods conducted in [13], where it was suggested that Clustering is not as hard as claimed by corresponding $\mathcal{NP}$-hardness proofs.

We shall also study the complexity of the multi-parameterized version of Cluster Editing when the error-parameters are small constants. In particular, we show that Cluster Editing remains $\mathcal{NP}$-hard even when at most one edge can be deleted and at most two edges can be added per vertex. Moreover, we show in this case that a simple reduction procedure yields a problem kernel whose total size is linear in the parameter k. Previously known kernelization algorithms cannot be applied to the considered multi-parameterized version and they deliver kernels whose order (number of vertices only) is linear in k.

The paper is organized as follows: Section 2 presents some preliminaries; in Section 3 we study the complexity of Cluster Editing when parameterized by the error-parameters; Section 4 is devoted to a general reduction procedure; the consequent complexity results are presented in Sections 5; and Section 6 concludes with a summary.

2. Preliminaries

We adopt common graph theoretic terminologies[2] such as neighborhood, vertex degree and adjacency. 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 $G[S]$. The neighborhood of a vertex $v \in V$ is denoted by $N(v)$ and, for $S \subset V$, the neighborhood of S in G is $N(S) = \cup_{v \in S} N(v)$. A clique in a graph is a subgraph induced by a set of pair-wise adjacent vertices. An edge-editing operation is either a deletion or an addition of an edge. We shall use the term *cluster graph* to denote a transitive undirected 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-editing operations. In this paper, we consider a multi-parameterized version of this problem that assumes a set of constraints, or parameters (independent of the input). The (a, d, s, k)-Cluster Editing is formally defined as follows.

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

We shall further use some special terminology to better present our results. 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*. Three vertices that induce a path of length two are 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 is permanent. To *join* a vertex v to a 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 the three cases corresponding to deleting one of the two edges in the path or inserting the missing edge [18]. In each of the three cases, the algorithm proceeds recursively. As such, the said algorithm runs in $O(3^k n^2)$ time (3 cases per conflict triple). The same idea has been used in almost all subsequent algorithms, which added more sophisticated branching rules.

We first study the (a, d)-Cluster Editing problem, which corresponds to the case where k and s are not parameters or their values are set to infinity and one respectively. This version is similar to the one introduced in [21] where a bound c is placed on the total number of edge-edit operations per single vertex. The corresponding problem is called c-Cluster Editing. When $c \geq 4$, c-Cluster Editing is $\mathcal{NP}$-hard (shown also in [21]). This does not imply, however, that (a, d)-Cluster Editing is $\mathcal{NP}$-hard when $a + d = 4$. To see this, note that $(a, 0)$-Cluster Editing is solvable in polynomial time for any $a \geq 0$: any

[2] We refer the reader to the book of Diestel [15] for common graph theoretic terminologies.

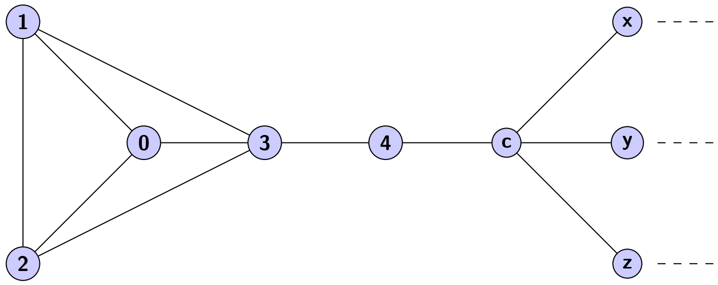

Fig. 1. Clause gadget. The node corresponding to clause $c = (x \vee y \vee z)$ is forced to lose one of the edges connecting it to a variable gadget.

solution must add all the needed edges to get rid of all conflict-triples, if this is possible with at most a edge additions per vertex.

Observe that if (a, d)-Cluster Editing is $\mathcal{NP}$-hard then so is (a', d')-Cluster Editing 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 shall prove that $(2, 1)$-Cluster Editing is $\mathcal{NP}$-hard, which yields the $\mathcal{NP}$-hardness of $(a, 1)$-Cluster Editing for all $a > 1$. It was shown in [21] that $(0, 2)$-Cluster Editing is $\mathcal{NP}$-hard. We observe in Section 5 that $(0, 1)$-Cluster Editing is solvable in polynomial time, and we conjecture that $(1, 1)$-Cluster Editing is solvable in polynomial time.

A kernelization algorithm with respect to an input parameter k is a polynomial time reduction procedure that yields an equivalent problem instance whose size is bounded by a function of k. Known kernelization algorithms for Cluster Editing have so far obtained kernels whose order (i.e., number of vertices) is bounded by a linear function of k [8,9,19]. The most recent order-bound is $2k$ [9].

We introduce the minimum acceptable cluster size, s, as another constraint/parameter. This is especially useful when the input graph is preprocessed so it is not expected to contain outlier vertices. Observe that the size of a cluster is expected to be relatively large in some correlation clustering applications, such as social networks [23]. We shall present (in Section 4) a simple reduction procedure that leads to solving Cluster Editing in polynomial time when $s > 2(a + d)$. When $s \leq 2(a + d)$, the same reduction procedure delivers kernels whose number of edges is bounded by $\frac{5k}{4} max(a, 2d)(a + 3d)$. In other words, when the constraints a and d are small constants, the kernel size is linear in k.

3. The (a, d)-cluster editing problem

We consider the case where the Cluster Editing problem is parameterized by the addition and deletion capacities a and d, only. In other words, the main objective is to check whether it is possible to obtain a cluster graph by performing at most a additions and d deletions per vertex. We denote the corresponding problem by (a, d)-Cluster Editing.

It was shown in [21] that Cluster Editing is $\mathcal{NP}$-hard when the total number of edge-edit operations per vertex is four or more. However, as observed earlier, the result (and proof) of [21] cannot be used in studying the complexity of each of the separate cases (for a and d) when $a + d = 4$. We prove in this section that $(2, 1)$-Cluster Editing is $\mathcal{NP}$-hard by reduction from the 4-Bounded-Positive-one-in-three-SAT problem, which is formally defined as follows:

4-Bounded positive one-in-three-SAT
Given: A 3-CNF formula ϕ in which each variable appears only positively and in at most four clauses.
Question: Is there a truth assignment that satisfies ϕ so that only one variable per clause is set to true?

4-Bounded-positive-one-in-three-SAT was shown to be $\mathcal{NP}$-hard in [14]. The reduction to $(2, 1)$-Cluster Editing proceeds by constructing a graph with three types of vertices: *variable*, *clause* and *auxiliary*. Each clause $c = (x \vee y \vee z)$ is represented by a clause gadget as shown in Fig. 1.

Observe that edge $\{3, 4\}$ must be deleted since the K_4 formed by $\{0, 1, 2, 3\}$ is permanent. It follows that edge $\{c, 4\}$ cannot be deleted, so exactly one of the three edges cx, cy and cz must be deleted by any feasible solution. The deletion of the edge cx means that the variable x is set to true, otherwise it is false.

Since each variable is a member of at most four clauses, the gadget corresponding to a variable x contains, among other vertices, four vertices labeled x. Each such vertex is connected to exactly one clause gadget as shown in Fig. 1 above. If x is set to false, each vertex labeled x has to remain in the clause gadget in the final cluster graph, as shown in Fig. 2. On the other hand, if x is set to true, each vertex labeled x must loose the edge connecting it to the corresponding clause gadget.

In the variable gadget, shown in Fig. 3, the four edges connecting the vertices labeled x to the cycle are subject to the same edit operation: either all four are deleted or all four are kept as permanent. To see this, note that it is not possible to delete exactly one edge of the cycle $\{1, 2, 3, 4\}$. We either have to turn this cycle to a K_4 by two edge additions and by removing each of the edges incident on vertices labeled x, or two opposite edges are deleted as shown in Fig. 4.

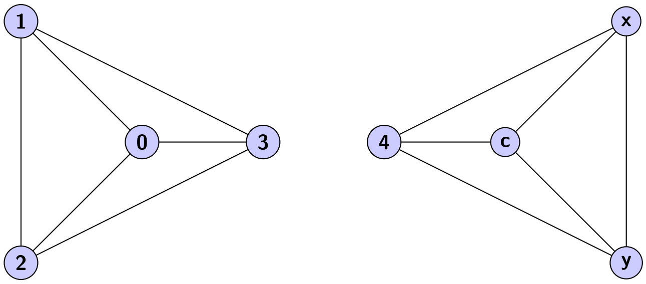

Fig. 2. Clause gadget resolved after setting z to true while x and y are set to false.

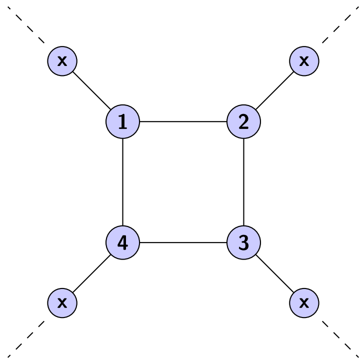

Fig. 3. Variable gadget. Variable x may belong to up to four clauses. Each node labeled x is connected to its corresponding clause gadget.

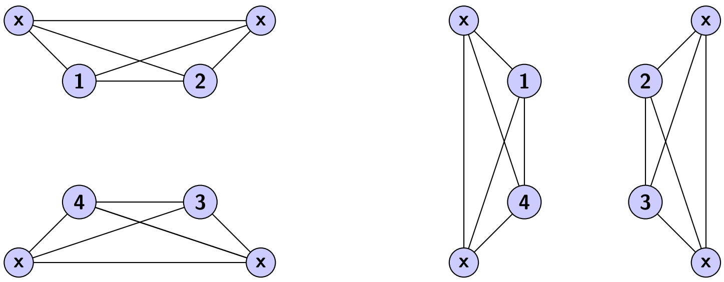

Fig. 4. Variable gadget: the two cases where exactly two edges of the C_4 are deleted.

If a variable x belongs to i different clause(s) where $i < 4$, then $4 - i$ vertice(s) labeled x in the corresponding gadget will be pendant (not connected to a clause gadget). If there is a satisfying truth assignment of a given formula of $4BP$-one-in-three-SAT, then every variable that is set to true will have its variable gadget turned into two clusters as shown in Fig. 4, and every variable x that is set to false will have its corresponding four vertices disconnected from the C_4, which is turned into a K_4 in its gadget. In the latter case, each of the vertices labeled x will join a (corresponding) clause cluster as shown in Fig. 2.

Conversely, if there is a solution of the $(2, 1)$-Cluster Editing instance, then exactly one variable-vertex neighbor of a clause-vertex is deleted. Due to the above construction, the corresponding variable can be set to true and a satisfying assignment is obtained for the $4BP$-one-in-three-SAT instance. Note again that the variable gadget ensures that all of its four variable-vertices behave the same: either all the four edges between vertices labeled x and the cycle are deleted or none of them is deleted. So once a variable is set to true in one clause it must be true in all other clauses where it appears. We have thus proved the following.

Lemma 1. *The* $(2, 1)$*-Cluster Editing problem is* $\mathcal{NP}$*-Hard.*

The membership of $(a, 1)$-Cluster Editing in $\mathcal{NP}$ is obvious. With the above Lemma, and our definition of (a, d)-Cluster Editing, we obtain the following theorem.

Theorem 1. *For* $a > 1$, *the* $(a, 1)$*-Cluster Editing problem is* $\mathcal{NP}$*-Complete.*

4. 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.

In what follows, we assume an instance (G, a, d, s, k) of (a, d, s, k)-Cluster Editing is given. In other words, the problem is parameterized by the add and delete capacities, as in the previous section, along with the lower-bound on cluster size s and the total number of edge-edit operations k. Any added edge, in what follows, is automatically set to permanent.

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. Some of the rules are folklore. We shall prove the soundness of new, non-obvious, reduction rules only.

4.1. Base-case reductions

Reduction rule 1. The reduction algorithm terminates and reports a no-instance, whenever any of $k, \alpha(v)$, or $\delta(v)$ is negative for some vertex $v \in V(G)$.

Reduction rule 2. For any vertex v, if $\delta(v) = 0$ (or becomes zero), then set every edge incident to v to permanent.

Reduction rule 3. If $\alpha(u) = 0$, then for each $v \in V(G) \setminus N(u)$, set the non-edge uv to forbidden.

4.2. Reductions based on conflict-triples

The following two reduction rules seem to have first appeared in [18]. Of course, they are slightly adjusted here to work with our multi-parameterized version of the problem.

Reduction rule 4. If uv and uw are permanent edges and vw is not permanent then:

– if vw is forbidden then halt and report a no-instance.
– if vw is a non-edge that is not forbidden, then add vw and decrement each of k, $\alpha(v)$ and $\alpha(w)$ by one.
– if vw is an existing edge, then set it to permanent.

At this stage, if $\delta(v) = 0$ for some vertex v, then applying Rule 2 and Rule 4 leads to cliquifying $N(v)$ unless there is a forbidden non-edge between two neighbors of v. We also note that adding edges between two vertices may yield negative parameters, which triggers Rule 1 and causes the algorithm to terminate with a negative answer.

Reduction rule 5. If uv is a permanent edge and uw is a forbidden edge, then set vw as forbidden. If vw exists, delete it and decrement each of k, $\delta(v)$ and $\delta(w)$ by one.

Soundness: After applying Rule 4, vw cannot be a permanent edge.

4.3. Reductions based on common neighbors

Reduction rule 6. If two non-adjacent vertices u and v have more than $\delta(u) + \delta(v)$ common neighbors, and uv is not forbidden, then add edge uv and decrement each of k, $\alpha(u)$ and $\alpha(v)$ by one. If uv is forbidden then halt and report a no-instance.

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 its edge-deletion capacity, which is not possible.

Reduction rule 7. If two adjacent vertices, u and v, have at least $\delta(u) + \delta(v) - 1$ (or $2d - 1$) common neighbors then set uv as permanent edge.

Soundness: If the two vertices are in different clusters of a solution graph, then deleting edge uv reduces both $\delta(u)$ and $\delta(v)$ by one (so each would be at most $d - 1$). Since they have to lose their $\delta(u) + \delta(v) - 1$ common neighbors, then at least one of them has to lose more edges than its edge-deletion capacity, which is not possible.

Reduction rule 8. If two vertices u and v are such that $|N(u) \setminus N(v)| > \alpha(v) + \delta(u)$ (or just $a + d$) then set edge uv as forbidden. If u and v are adjacent, and uv is not permanent, then delete uv and decrement each of k, $\delta(u)$ and $\delta(v)$ by one. If uv is permanent, then halt and report a no-instance.

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

4.4. Reductions based on cluster-size

Reduction rule 9. If $s > 2$ and two non-adjacent vertices u and v have less than $s - \alpha(u) - \alpha(v)$ common neighbors (or $s - 2a$ such neighbors) then set edge uv as forbidden.

Soundness: If $\alpha(u) = 0$ or $\alpha(v) = 0$, then uv is already forbidden by Rule 3. 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 $\alpha(u) + \alpha(v) - 2$ ($\alpha(u) - 1$ for u and $\alpha(v) - 1$ for v). The total number of common neighbors after adding all possible edges remains less than $s - 2$.

Reduction rule 10. If $s > 2$ and two adjacent vertices u and v have less than $s - \alpha(u) - \alpha(v) - 2$ common neighbors, then delete edge uv and decrement each of k, $\delta(u)$ and $\delta(v)$ by one. If uv is a permanent edge, then halt and report a no-instance.

Soundness: The argument is similar to the previous case, except that the maximum number of common neighbors that can be added is bounded above by $\alpha(u) + \alpha(v)$, thus the total number of common neighbors remains less than $s - 2$.

4.5. Permanent and isolated cliques

An isolated clique C is a clique whose vertices have no neighbors in $V(G) \setminus C$, i.e., $N(C) \setminus C = \emptyset$. An isolated clique is said to be *small* if its size is less than s. In general, deleting isolated cliques is a sound reduction rule for the single-parameter Cluster Editing problem. In multi-parameterized versions, we either add a parameter that bounds the number of small isolated cliques, including outliers, or (to adhere to our problem formulation) we must keep a number of such cliques. To see this, note the example of a single isolated vertex v and an isolated clique C with less than $\alpha(v)$ vertices. In this case, v can be joined to C to avoid having a small cluster that can potentially yield a no answer.

If a clique contains more than $2d$ vertices, then it is permanent due to Rule 7. Moreover, no vertex can be joined to an isolated permanent clique with more than a vertices. As a consequence, we obtain the following reduction rule.

Reduction rule 11. If an isolated clique C has more than $max(a, 2d)$ vertices then:
if $|C| \geq s$, then delete C; otherwise halt and report a no-instance.

The presence of permanent cliques can trigger problem reductions that are not obtained by exhaustive applications of the above reduction rules. Note that, in a reduced instance, the subgraph formed by permanent edges is a disjoint union of permanent cliques. This latter observation is due to the exhaustive applications of previous reduction rules (especially Rule 4). It follows that finding all the maximal permanent cliques takes polynomial time (a disjoint-set data structure may also be used to keep track of all the permanent cliques.)

Reduction rule 12. If a vertex v has more than $\delta(v)$ neighbors in a permanent clique C, then join v to C and decrement k and $\alpha(v)$ by $|C \setminus N(v)|$. Moreover, for each $u \in C \setminus N(v)$, decrement $\alpha(u)$ by one.

Soundness: Obviously, it is not possible to delete some, but not all, edges connecting v to C. Moreover, if $u \in C \setminus N(v)$ then uv cannot be forbidden, otherwise all edges connecting v to elements of C would have to be deleted by the application of Rule 5, which is impossible since v has more than $\delta(v)$ neighbors in C. Note here that a no-instance would automatically result if any of the parameters becomes negative (by Rule 1).

Reduction rule 13. Let C be a permanent clique of size $> a$. If a vertex v has less than $|C| - \alpha(v)$ neighbors in C, then detach v from C and decrement k and $\delta(v)$ by $|C \cap N(v)|$. Moreover, for each $u \in C \cap N(v)$, decrement $\delta(u)$ by one.

Soundness: If there is a permanent edge between v and an element of C then v has to be joined to C due to (possibly successive application of) Rule 4, which is impossible. Therefore any edge between v and C must be deleted.

At this stage, if there is an isolated clique C in the so-far reduced instance, then $|C| \leq max(a, 2d)$. If C is small (i.e., $1 \leq |C| < s$) then each vertex of C must be affected by at least one edge-editing operation. Consequently:

Reduction rule 14. If the total number of vertices in small isolated cliques is $> 2k$ then (halt and) report a no-instance.

On the other hand, if an isolated clique C is not small then it must satisfy $s \le |C| \le max(a, 2d)$. As observed above, we need to keep a few of these cliques. Since this is needed only when $s \ge 2$, we can safely keep at most $k/2$ such cliques.

Reduction rule 15. Let C be a non-small isolated clique of maximum size. If $s > 1$ and there are more than $k/2$ such non-small isolated cliques, then delete C.

Soundness: Since C has at least s vertices, it can be considered a cluster in the solution graph. If a vertex must be joined to an isolated clique (to avoid having a small isolated clique), then the smallest of the other $k/2$ non-small isolated cliques can be used for this purpose. Moreover, as observed above, at most $k/2$ such non-small isolated cliques can be extended into larger cliques without consuming more than k edge additions.

5. The complexity of multi-parameterized cluster editing

An instance of (a, d, s, k)-Cluster Editing is said to be reduced if the above reduction rules have been exhaustively applied to the input graph G.

Our second main theorem, proved below, addresses the optimization version of (a, d, s, k)-Cluster Editing, which seeks a minimum number of edge-edit operations. So k is not a parameter in this case, but we keep the other constraints.

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

Proof. Let u and v be non-adjacent vertices such that uv is not forbidden. By Rules 6 and 9: $s - 2a \le |N(u) \cap N(v)| \le 2d$. This is impossible since $s - 2a > 2d$, so uv must be forbidden and any two non-adjacent vertices must belong to different clusters. If u and v are adjacent vertices such that uv is not permanent, then by Rules 7 and 10: $s - 2a - 2 \le |N(u) \cap N(v)| < 2d - 1$. Again, this is impossible (it implies $s < 2(a + d) + 1$), so edges between adjacent vertices must be permanent. It follows that in a reduced instance any edge is either permanent or deleted. □

In a typical clustering application, 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 large compared to such error, Theorem 2 asserts that Cluster Editing is solvable (exactly) in polynomial time.

When $s \le 2(a + d)$, the Minimum Cluster Editing problem remains $\mathcal{NP}$-hard even if a and d are small constants and the size of a cluster is not important (i.e., $s = 1$). In this case, the reduction procedure may still help to obtain faster parameterized algorithms. In fact, we shall prove that applying the above reduction rules yields equivalent instances whose (total) size is bounded by a linear function of the main parameter k. The following key lemma follows from the reduction procedure.

Lemma 2. *Let (G, a, d, s, k) be a reduced yes-instance of 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 6, 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 8. By Rules 7 and 8, every edge incident on v is either deleted or becomes permanent. Applying Rules 4 and 5 exhaustively leads to cliquifying and isolating $N[v]$, which then results in deleting $N[v]$ due to Rule 11. □

5.1. The case where a and d are small fixed constants

It was shown in [21] that Cluster Editing is $\mathcal{NP}$-hard when $a = 0$ and $d = 2$. This implies the $\mathcal{NP}$-hardness of $(a, 2)$-Cluster Editing for $a \ge 0$. As noted earlier, this case corresponds to $s = 1$ and $k = \infty$ (or just $|V(G)|^2$).

When $d = 1$, our reduction procedure results in a triangle-free instance or a no answer. To see this observe that every clique of size three or more becomes permanent ($2d + 1 = 3$). Moreover, a vertex with more than one neighbor in a clique of size three must be in the clique, by Rule 12, while an edge joining a vertex to only one member of a triangle must be deleted, by Rule 13. It follows that cliques of size three or more become isolated and deleted. We observe the following.

Proposition 1. *The $(0, 1)$-Cluster Editing problem is solvable in polynomial time.*

Proof. To see this, note that a vertex of degree three must be part of a clique of size at least three, since at most one of its incident edges can be deleted. Unless we have a no-instance, such vertex cannot exist in a reduced instance, being triangle-free by the above discussion. It follows that a reduced yes-instance must have a maximum degree of two. In this case, the problem is easily solved in linear-time. □

At this stage, the only remaining open problem is whether (1, 1)-Cluster Editing is solvable in polynomial time. We believe it is the case, especially since every reduced instance is triangle-free, as observed above. Moreover, the reader would easily observe that every such instance is of maximum degree three.

5.2. Kernelization

We now give a bound on the number of vertices in a reduced instance.

Theorem 3. *There is a polynomial-time reduction algorithm that takes an arbitrary instance of multi-parameterized Cluster Editing and either determines that no solution exists or produces an equivalent instance whose order is bounded by* $\frac{5k}{2} max(a, 2d)$.

Proof. Let (G, a, d, k) be a reduced instance of Cluster Editing. Let A be the set of vertices of G that are incident to edges that must be deleted or to non-edges that must be added to obtain some minimum solution, if any. In other words, A is the set of vertices *affected* by edge-editing operations. If (G, a, d, k) is a yes-instance, then $|A| \leq 2k$. Let $B = N(A)$ and $C = V(G) \setminus (A \cup B)$.

The connected components of B are cliques, being non-affected by edge-editing. For the same reason, every vertex $x \in A$ satisfies: $G[B \cap N(x)]$ is a clique. Let $B_x = N(x) \cap B$ for some $x \in A$ and let $N(B_x) = \{u \in A : B_x \subset N(u)\}$. If $|B_x \cup \{x\}| > max(a, 2d)$ then our reduction procedure sets $B_x \cup \{x\}$ to permanent and automatically joins every element of $N(B_x)$ to a larger isolated clique containing B_x, which is then deleted by Rule 11. Therefore $|A \cup B| \leq 2k \cdot max(a, 2d)$.

Observe that isolated cliques of size $< s$ must be totally contained in A, so their elements are part of the $2k \cdot max(a, 2d)$ vertices of $A \cup B$. In fact, if there are more than $2k$ such elements, the application of Rule 14 would have resulted in a no answer. It follows by Rule 15 that C consists of at most $k/2$ (non-small) isolated cliques, each of size $max(a, 2d)$. □

The following corollary follows easily from Theorem 3 and Lemma 2.

Corollary 1. *There is a polynomial-time reduction algorithm that takes an arbitrary instance of Multi-parameterized Cluster Editing and either determines that no solution exists or produces an equivalent instance whose size is bounded by* $\frac{5k}{4} max(a, 2d)(a + 3d)$.

6. Concluding remarks

We considered the (a, d)-Cluster Editing problem, a constrained version of Cluster Editing where at most a edges can be added and at most d edges can be deleted per single vertex. We proved that (a, d)-Cluster Editing is $\mathcal{NP}$-hard, in general, for any $a \geq 2$ and $d \geq 1$. We also observed that (0, 1)-Cluster Editing is solvable in polynomial time while the (0, 2) case is $\mathcal{NP}$-hard [21]. It remains open whether (1, 1)-Cluster Editing can be solved in polynomial time.

We presented a reduction procedure that solves the Cluster Editing problem in polynomial time when the smallest acceptable cluster size exceeds twice the total allowable edge operations per vertex. It is worth noting that edge-editing operations per single data element are expected to be small compared to the cluster size, especially if the input is free of outliers. When the bounds on the two edge-edit operations per vertex are small constants, and the cluster size is unconstrained, our reduction algorithm gives a kernel whose size is linear in the main parameter k. Previously known kernelization algorithms are not applicable to the multi-parameterized version and achieve a linear bound on the number of vertices only.

The reduction algorithm presented in this paper has been implemented along with the simple branching on conflict triples described in Section 2. Experiments show a promising performance especially on graphs obtained from clinical research data [3]. The multi-parameterized algorithm finished consistently in seconds, reporting significant clusters.

Finally we note that different Cluster Editing solutions to the same problem instance may differ in terms of the practical significance of obtained clusters. A possible approach would be to combine enumeration and editing, as reported in [11], by enumerating all possible Cluster Editing solutions. Furthermore, in some biology applications, a data element may be an active member of different clusters. In such cases, the enumeration of all maximal cliques was used as a possible alternative [2]. To permit a data element to belong to more than one cluster, we suggest allowing *vertex-division* (AKA. vertex *cleaving*) as another edit operation whereby a vertex is replaced by two different vertices. The number of allowed divisions per vertex can be added as (yet) another parameter. This latter formulation is under consideration for future work.

Acknowledgements

The author is grateful to the anonymous referees for their valuable and significant comments and suggestions that resulted in an improved presentation of this paper.

The work on this project has been partially supported by the Lebanese American University under grant SRDC-t2013-45.

References

[1] F.N. Abu-Khzam, The multi-parameterized cluster editing problem, in: P. Widmayer, Y. Xu, B. Zhu (Eds.), Combinatorial Optimization and Applications, 7th International Conference Proceedings, COCOA 2013, Chengdu, China, December 12–14, 2013, in: Lecture Notes in Computer Science, vol. 8287, Springer, 2013, pp. 284–294.

[2] F.N. Abu-Khzam, N.E. Baldwin, M.A. Langston, N.F. Samatova, On the relative efficiency of maximal clique enumeration algorithms, with applications to high-throughput computational biology, in: International Conference on Research Trends in Science and Technology, 2005.
[3] F.N. Abu-Khzam, L. Lin, P. Shaw, H. Smith-Vaughan, R. Marsh, Effective use of multi-parameterized correlation clustering in mining nasopharyngeal carriage and disease data from young children. Manuscript, 2017.
[4] S. Böcker, A golden ratio parameterized algorithm for cluster editing, J. Discret. Algorithms 16 (2012) 79–89.
[5] S. Böcker, S. Briesemeister, Q.B.A. Bui, A. Truss, Going weighted: parameterized algorithms for cluster editing, Theor. Comput. Sci. 410 (52) (2009) 5467–5480.
[6] S. Böcker, S. Briesemeister, G.W. Klau, Exact algorithms for cluster editing: evaluation and experiments, Algorithmica 60 (2) (2011) 316–334.
[7] L. Cai, Fixed-parameter tractability of graph modification problems for hereditary properties, Inf. Process. Lett. 58 (4) (May 1996) 171–176.
[8] Y. Cao, J. Chen, Cluster editing: kernelization based on edge cuts, Algorithmica 64 (1) (2012) 152–169.
[9] J. Chen, J. Meng, A 2k kernel for the cluster editing problem, J. Comput. Syst. Sci. 78 (1) (2012) 211–220.
[10] M. Cygan, F.V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh, Parameterized Algorithms, Springer, 2015.
[11] P. Damaschke, Fixed-parameter enumerability of cluster editing and related problems, Theory Comput. Syst. 46 (2) (2010) 261–283.
[12] P. Damaschke, Sufficient conditions for edit-optimal clusters, Inf. Process. Lett. 116 (4) (2016) 267–272.
[13] A. Daniely, N. Linial, M. Saks, Clustering is difficult only when it does not matter, arXiv:1205.4891, 2012.
[14] R. Denman, S. Foster, Using clausal graphs to determine the computational complexity of k-bounded positive one-in-three {SAT}, Discrete Appl. Math. 157 (7) (2009) 1655–1659.
[15] R. Diestel, Graph Theory, 4th edition, Graduate Texts in Mathematics, vol. 173, Springer, 2012.
[16] R.G. Downey, M.R. Fellows, Fundamentals of Parameterized Complexity, Texts in Computer Science, Springer, 2013.
[17] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, 2006.
[18] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Graph-modeled data clustering: exact algorithms for clique generation, Theory Comput. Syst. 38 (4) (2005) 373–392.
[19] J. Guo, A more effective linear kernelization for cluster editing, Theor. Comput. Sci. 410 (8–10) (2009) 718–726.
[20] P. Heggernes, D. Lokshtanov, J. Nederlof, C. Paul, J.A. Telle, Generalized graph clustering: recognizing (p,q)-cluster graphs, in: Dimitrios M. Thilikos (Ed.), Graph Theoretic Concepts in Computer Science, 36th International Workshop Revised Papers, WG 2010, Zarós, Crete, Greece, June 28–30, 2010, in: Lecture Notes in Computer Science, vol. 6410, 2010, pp. 171–183.
[21] C. Komusiewicz, J. Uhlmann, Cluster editing with locally bounded modifications, Discrete Appl. Math. 160 (15) (2012) 2259–2270.
[22] M. Křivánek, J. Morávek, Np-hard problems in hierarchical-tree clustering, Acta Inform. 23 (3) (June 1986) 311–323.
[23] J. Leskovec, K.J. Lang, A. Dasgupta, M.W. Mahoney, Community structure in large networks: natural cluster sizes and the absence of large well-defined clusters, Internet Math. 6 (1) (2009) 29–123.
[24] D. Lokshtanov, D. Marx, Clustering with local restrictions, Inf. Comput. 222 (2013) 278–292.
[25] R. Niedermeier, An Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[26] R. Shamir, R. Sharan, D. Tsur, Cluster graph modification problems, Discrete Appl. Math. 144 (1–2) (2004) 173–182.