Cluster Editing with Vertex Splitting

Faisal N. Abu-Khzam[1,2], Judith Egan[1], Serge Gaspers[3,4], Alexis Shaw[5], and Peter Shaw[6]

[1] Charles Darwin University, Darwin, Australia
[2] Lebanese American University, Beirut, Lebanon
[3] The University of New South Wales, Sydney, Australia
[4] Data61, CSIRO, Sydney, Australia
[5] Centre for Quantum Computation and Communication Technology, Centre for Quantum Software and Information, Faculty of Engineering and Information Technology, University of Technology Sydney, Australia
[6] Massey University, Manawatu, New Zealand

Abstract. In the Cluster Editing problem, a given graph is to be transformed into a disjoint union of cliques via a minimum number of edge editing operations. In this paper we introduce a new variant of Cluster Editing whereby a vertex can be divided, or split, into two or more vertices thus allowing a single vertex to belong to multiple clusters. This new problem, Cluster Editing with Vertex Splitting, has applications in finding correlation clusters in discrete data, including graphs obtained from Biological Network analysis. We initiate the study of this new problem and show that it is fixed-parameter tractable when parameterized by the total number of vertex splitting and edge editing operations. In particular we obtain a $4k(k+1)$ vertex kernel for the problem.

1 Introduction

Given a graph G and a non-negative integer k, the Cluster Editing problem asks whether G can be turned into a disjoint union of cliques by a sequence of at most k edge-editing operations. The problem is known to be $\mathcal{NP}$-Complete since the work of Křivánek and Morávek in [20], and does not seem to have any reasonable polynomial-time approximation unless the number of clusters is at most two [24].

We assume that the reader is familar with fixed-parameter tractability and kernelization [9,12,16,22]. The Cluster Editing problem is fixed-parameter tractable when parameterized by k, the total number of edge editing operations [6, 17]. Over the last decade, Cluster Editing has been well studied from both theoretical and practical perspectives (see, for example, [4, 7, 8, 10, 11, 13, 14, 18, 19]).

In general, clustering results in a partition of the input graph, thus it forces each and every data element to be in one and only one cluster. This can be a limitation when a data element plays roles in multiple clusters. This situation, (i.e. the existence of *hubs*), is recorded in work on gene regulatory networks [1], where enumeration of maximal cliques was considered a viable alternative to clustering.

Moreover, the existence of hubs can effectively hide clique-like structures and also greatly increase the computational time required to obtain optimum correlation clustering solutions [23, 25]. Improved solutions (for correlation clustering) can be obtained using the Multi-parameterized Cluster-Editing problem [2] which further restricts the number of false positive and/or false negative correlations (add and delete edge-edits incident to a vertex) that can be ascribed to a given variable. However, the need to identify variables that lie in the intersection of multiple clusters could further complicate this multi-parameterized model.

The Cluster Editing with Vertex Splitting problem (CEVS) is introduced in this paper in an attempt to allow for overlapping clusters in graphs that are assumed to be noisy in the sense that edges are assumed to have been perturbed after the clusters overlap. CEVS can be viewed as an extended version of the Cluster Editing problem.

In addition to introducing CEVS, we investigate its parameterized complexity and obtain a polynomial kernel for the problem. In doing so we employ the notion of a critical clique, as introduced in [21], and applied to the Cluster Editing problem in [18]. Our proof technique is based on a novel *c*lean edit sequence approach which could be of interest by itself.

This paper is structured as follows. Section 2 overviews some background material. Section 3 introduces the edit sequence approach while section 4 is devoted to critical cliques. In section 5 we obtain a quadratic kernel. We conclude in section 6 with a summary and future directions.

2 Preliminaries

We assume familiarity with basic graph theoretic terminology. All graphs in this work are simple, unweighted and undirected. The vertex and edge sets of a graph G are denoted by $V(G)$ and $E(G)$ respectively. For a subset V' of $V(G)$, we denote by $G[V']$ the subgraph of G that is induced by V'.

A *kernelization* or *kernel* for a parameterized problem P is a polynomial time function that maps an instance (I, k) to an instance (I', k') of P such that:

- (I, k) is a YES instance for P if and only if (I', k') is a YES instance;
- $|I'| < f(k)$ for some computable function f;
- $k' < g(k)$ for some computable function g.

A *proper* kernelization is a kernelization such that $g(k) < k$ [3]. The function $f(k)$ is also called the size of the kernel. A problem has a kernel if and only if it is FPT [12], however not every FPT problem has a kernel of polynomial size [5].

A *k-partition* of a set S is a collection of pairwise disjoint sets $S_1, S_2, \ldots S_k$ such that $S = \bigcup_{i=1}^{k} S_i$. A *k-covering* of a set S is a collection of sets $S_1, S_2, \ldots S_k$ such that $S = \bigcup_{i=1}^{k} S_i$. A *cluster graph* is a graph in which the vertex set of each connected component induces a clique.

Problem Definition. The Cluster Editing with Vertex Splitting Problem (henceforth CEVS) is defined as follows. Given a graph $G = (V, E)$ and an integer k, can a cluster graph G' be obtained from G by a k-edit-sequence $e_1 \dots e_k$ of the following operations:

1. do nothing,
2. add an edge to E,
3. delete an edge from E, and
4. an *inclusive vertex split*, that is for some $v \in V$ partition the vertices in $N(v)$ into two sets U_1, U_2 such that $U_1 \cup U_2 = N(v)$, then remove v from the graph and add two new vertices v_1 and v_2 with $N(v_1) = U_1$ and $N(v_2) = U_2$.

A vertex $v \in V(G)$ is said to *correspond* to a vertex $v' \in V(G')$, constructed from G by an edit-sequence S if v' is a leaf on the division-tree T for v defined as follows:

(i) v is the root of the tree, and

(ii) if an edit sequence operation splits a vertex u which lies on the tree then the two vertices that result from the split are children of u.

As noted earlier, Cluster Editing corresponds to the special case where no vertex splitting is permitted. So it would appear that Cluster Editing with Vertex Splitting is $\mathcal{NP}$-Hard due to the $\mathcal{NP}$-hardness of the Cluster Editing problem. Moreover, suppressing vertex splitting is not an option in the definition of CEVS. The $\mathcal{NP}$-hardness of the problem is not obvious, so we pose it as an open problem at this stage.

Our main focus in this paper is on the parameterized complexity of CEVS. We shall present a quadratic-size kernel for the problem, which proves it to be fixed-parameter tractable.

A similar problem has been defined and studied in [15] where a vertex is allowed to be part of at most s clusters. In this case s is either treated as constant or as a different parameter, which makes the $\mathcal{NP}$-hardness proof easy since the case $s = 1$ corresponds to the Cluster Editing problem. In our work we model the overlap via another editing operation so we do not set a separate parameter for the number of splittings per vertex. We are able to design a kernelization algorithm that achieves a quadratic-size kernel (while the best kernel bound achieved in [15] is cubic for the special case where $s = 2$).

3 The Edit-Sequence Approach

Defining CEVS in terms of edit-sequences is based on looking for the closest graph which is a cluster graph, where distance is defined by the shortest edit-sequence. An edit-sequence may however include a lot of redundancy (for example, swap two edge additions). In this section we show how to eliminate redundancy, first by showing that we can consider a specific form of edit sequence, and then showing that we can efficiently compute the shortest edit-sequence between two graphs. This will provide some much needed structure to the problem and provide a base for subsequent proofs.

3.1 Restricted Re-ordering of the edit-sequence

Two edit sequences $S = e_1 \dots e_k$ and $S' = e'_1 \dots e'_k$ are said to be *equivalent* if:

- G_S and $G_{S'}$, the graphs obtained from G by S and S' respectively are isomorphic to each other with isomorphism $f : V(G_S) \to V(G_{S'})$, and
- if $u_S \in V(G_S)$ and $u_{S'} = f(u_S)$ then the division tree which u_S is contained in and the division tree which $u_{S'}$ is contained in share a common root. In other words, u_S and $u_{S'}$ correspond to the same vertex of the original graph.

Lemma 1. *For any edit-sequence $S = e_1 \dots e_i e_{i+1} \dots e_k$ where e_i is an edge deletion and e_{i+1} is an edge addition, there is an equivalent edit-sequence $S' = e_1 \dots e'_i e'_{i+1} \dots e_k$ of the same length where either e'_i is an edge addition and e'_{i+1} is an edge deletion, or both e'_i and e'_{i+1} are do-nothing operations.*

Proof. We begin by noting that we only have to consider the edits e_i and e_{i+1} as we can think of the edit-sequence being a sequence of functions composed with each other, thus if e_i deletes edge uv and e_{i+1} adds edge wx then the graph immediately after applying the two operations in the opposite order will be the same in all cases except that where $uv = wx$ whereby the net effect is that nothing happens, as required.

Lemma 2. *For any edit-sequence $S = e_1 \dots e_i e_{i+1} \dots e_k$ where e_i is a vertex splitting and e_{i+1} is an edge deletion there is an equivalent edit-sequence $S' = e_1 \dots e'_i e'_{i+1} \dots e_k$ where either e'_i is an edge deletion and e_{i+1} is a vertex splitting or e'_i is a do-nothing operation and e'_{i+1} is a vertex splitting.*

Proof. If the edge deleted by e_{i+1} is not incident to one of the resulting vertices of the splitting e_i then swapping the two operations produces the required edit-sequence E'. Otherwise let e_i split vertex v and e_{i+1} delete edge uv_i. Then if e_i has associated covering U_1, U_2 of $N(v)$ and without loss of generality $u \in U_1$ then if $u \notin U_2$ then the edit-sequence with e'_i being a deletion operation deleting uv and e'_i being the vertex splitting and $U'_i = U_i \setminus \{u\}$ and $U'_1 = U_2$ is equivalent to E. Otherwise, $u \in U_1 \cap U_2$. Without loss of generality, suppose uv_2 is deleted by e_{i+1}. Then the sequence where e'_i is a do-nothing operation and where e'_{i+1} is a vertex splitting on v with covering U'_1, U'_2 with $U'_1 = U_1$ and $U'_2 = U_2 \setminus \{u\}$ is equivalent.

Lemma 3. *For any edit-sequence $S = e_1 \dots e_i e_{i+1} \dots e_k$ where e_i is a vertex splitting and e_{i+1} is an edge addition there exists an equivalent sequence $S' = e_1 \dots e'_i e'_{i+1} \dots e_k$ where either e'_i is an edge addition and e'_{i+1} is a vertex splitting, or e'_i is a do-nothing operation and e'_{i+1} is a vertex splitting.*

Proof. If the edge added by e_{i+1} is not incident to one of the resulting vertices of the splitting e_i then simply swap the two operations to produce the required edit-sequence E'. Otherwise, without loss of generality, let e_i divide vertex v on covering U_1, U_2 and e_{i+1} add vertex wv_1. Then let e'_i be the operation that adds the edge wv, if wv does not exist at that point, otherwise e'_i is a do-nothing operation and let e'_{i+1} split vertex v on covering $U'_1 = U_1 \cup \{w\}, U'_2 = U_2$. The resulting edit-sequence is equivalent to E.

Lemma 4. *For any edit-sequence $S = e_1 \dots e_k$ where e_i is a do-nothing operation, the edit-sequence $S' = e_1 \dots e_{i-1} e_{i+1} \dots e_k$ is equivalent to it and has strictly smaller length.*

3.2 Edit sequences in Add-Delete-Split form

From the above lemmas we can deduce the following theorem.

Theorem 1. *For every edit-sequence $S = e_1 \dots e_k$ there is an edit-sequence $S' = e'_1 \dots e'_{k'}$ with equal or lesser length such that*

1. *if e'_i is an edge addition and e'_j is an edge deletion or a vertex splitting, then $i < j$,*
2. *if e'_i is an edge deletion and e'_j is a vertex splitting, then $i < j$, and*
3. *S' contains no do-nothing operations*

We refer to an edit-sequence satisfying the statement of Theorem 1 as an edit-sequence in the add-delete-split form. We will now consider only these edit-sequences, as for any equivalence class of edit-sequences, there is a minimal member of that equivalence class which is in add-delete-split form. In fact, the equivalence class of an add-delete-split edit-sequence is the intersection of an equivalence class of edit-sequences and the set of edit-sequences in add-delete-split form. A minimal member of any such equivalence class is an edit-sequence in add-delete-split form.

Uniqueness of the pre-splitting edge modification graph corresponding to any add-delete-split edit-sequence equivalence class. It is now necessary to prove that in any equivalence class the graph obtained after the addition and deletion of edges and before splitting vertices is fixed. By doing so we provide a significant amount of structure to the problem, and do away with the direct use of edit-sequences altogether when searching for a solution.

The approach we adopt is to work on time-reversed edit-sequences, taking the final graph of the edit-sequence and the relation between the vertices in the initial graph and the final graph, and proving that we always arrive at the same graph. In preparation for this we define the *split relation*, $f : V \to 2^{V'}$ for a given solution S to CEVS for a graph G and edit-distance k as a function.

The split relation for such a solution S, graph G and distance k is a function $f : V \to 2^{V'}$ defined such that when $G' = (V', E')$ is derived from G by S the following properties hold on f

1. For a vertex $v \in V$: $v' \in f(v)$ if and only if v corresponds to v' under S,
2. For any $u, v \in V$ that $f(u) \cap f(v) = \emptyset$, and
3. For any $u \in V$ that $f(u) \neq \emptyset$.

A simple consequence of this definition is that two edit-sequences are equivalent if and only if the resulting graphs are isomorphic and the split relation is equivalent under that isomorphism.

In order to talk about time-reversed vertex-splitting sequences we define a *merge graph* as being a graph $G' = (V', E')$ derived from another graph $G = (V, E)$ by a sequence of vertex merge operations, that is there is a relation $f : V' \to 2^V$ called the *merge relation* which partitions the vertex set V on members of V' such that $u'v' \in E'$ if and only if $uv \in E$ for some $u \in f(u')$ and some $v \in f(v')$. A *vertex merge* operation constructs a merge graph with a merge relation such that there is only one v' such that $|f(v')| \neq 1$ and for that value $f(v') = u, v$; we call this the *merger* of u and v. A *k-merge-graph* G' of G is a merge graph for which there is a sequence of exactly k vertex merges $G^1 \dots G^k = G'$ such that for all $i = 1 \dots k$ and defining $G^0 = G$, we have G^i is derived from G^{i-1} by a vertex merge operation.

Lemma 5. *For any graph $G = (V, E)$ and merge-graph $G' = (V', E')$ of G with merge relation $R : V' \to 2^V$ we have that*

$$E' = \{u, v \in V' : \exists u' \in R(u)\ \exists v' \in R(v) \text{ such that } u'v' \in E\}\ .$$

Proof. If $V = V'$ then no merge has occurred and so this is trivially so, otherwise we proceed by induction on $k = |V| - |V'|$

Suppose that $G' = (V', E')$ is a k-merge-graph of G with merge relation $R^k : V' \to 2^V$ and suppose that G'' is a 1-merge-graph of G' and a $(k+1)$-merge-graph of G with relations $R' : V'' \to 2^{V'}$ and $R^{k+1} : V'' \to 2^V$. Without loss of generality suppose that G'' was produced by merging vertices u and v of G' into w. Then, by definition,

$$E'' = E' \backslash (\{vx \in E' | x \in V'\} \cup \{ux \in E' | x \in V'\}) \cup \{wx | x \in (N_{G'}(u) \cup N_{G'}(v))\})\,.$$

Therefore we deduce that the edge set E'' is the same as E' except on those edges incident to u and v, which were merged into W. However, $R'(x) = \{x\}$ for all vertices $x \neq w$ and $R'(w) = \{u, v\}$ and so,

$$E'' = \{u'', v'' \in V'' : \exists u' \in R'(u'')\ \ \exists v' \in R'(v'')\ \ u'v' \in E'\}.$$

By our induction hypothesis we also know that

$$E' = \{u', v' \in V' : \exists u \in R^k(u')\ \ \exists v \in R^k(v')\ \ uv \in E\}.$$

By merging these two equations together (noting that we can disclose the initial definition of u and v in the second equation) to give

$$E'' = \{u'', v'' \in V'' : \exists u' \in R'(u'')\ \exists v' \in R'(v'')\ \exists u \in R^k(u')\ \exists v \in R^k(v')\ uv \in E\}.$$

By re-ordering the order of the existential operators we can obtain

$$E'' = \{u'', v'' \in V'' : \exists u' \in R'(u'')\ \exists u \in R^k(u')\ \exists v' \in R'(v'')\ \exists v \in R^k(v')\ uv \in E\}.$$

This is the same as saying,

$$E'' = \{u'', v'' \in V'' : \exists u \in \bigcup_{u' \in R'(u'')} R^k(u') \;\; \exists v \in \bigcup_{v' \in R'(v'')} R^k(v') \;\; uv \in E\}.$$

By the definition or R^{k+1} this means

$$E'' = \{u'', v'' \in V'' : \exists u \in R^{k+1}(u'') \;\; \exists v \in R^{k+1}(v'') \;\; uv \in E\}$$

as required.

3.3 Representation of edit-sequences as resultant graphs and merge relations

We can see that a vertex merge is the time-reversed image of a vertex splitting, in as much as that any sequence of splits will correspond to a time-reversed sequence of merges and the converse. Thus we can use vertex mergers to prove the following.

Lemma 6. *For any collection of edit-sequences in add-delete-split form which are equivalent to some edit-sequence S, the graph G_{R_S} immediately preceding the vertex splitting is the same for all members of the class in that form. Further if the split relation for this equivalence class is R and the graph $G' = (V'E')$ resulting from S are known, then*

$$E' = \{u, v \in V' : \exists u' \in R(u) \; \exists v' \in R(v) \text{ such that } u'v' \in E\} \;\; .$$

Proof. This follows directly from Lemma 5, and Theorem 1.

We are now ready to prove the following lemma:

Lemma 7. *For any graph $G = (V, E)$ there is a computable bijection between pairs $(G' = (V', E'), f : V \to 2^{V'})$ of resultant graphs and split-relations and equivalence classes of edit-sequences. Further there is an algorithm to compute a min-edit-sequence from the resultant graph/split-relation pair for this class in $O((|V'| - |V|)\Delta(G) + |V| + |E| + |V'| + |E'|)$ time.*

Proof. The edit-sequence to graph/relation direction has been proved above, further we have proved that if two edit-sequences have the same graph/split-relation pair then they are equivalent. Thus all that remains to be proved is that we can always construct a min-edit-sequence from an input graph to a valid resultant graph/split-relation pair. We note:

- As the split-relation f can be represented as a merge-relation it is possible by Lemma 5 to construct a graph G_R such that G_R has the same vertex set as G and there is an edit-sequence consisting of only vertex divisions from G_R to G' and with relation-relation f. Further this can be done in $O((|V'| - |V|)\Delta(G) + |V| + |E| + |V'| + |E'|)$ time, and

– As G_R shares the same vertex set as G it is possible to construct an optimal edit-sequence from G to G_R with only edge additions and deletions (by looking at the edge sets of G and G_R). Further we can do this in $O(|E|)$ time.

So we can construct an edit-sequence from G to G' with split-relation f in $O((|V'| - |V|)\Delta(G) + |V| + |E| + |V'| + |E'|)$ time. Further by Lemma 6 this graph G_R is fixed for all edit-sequences in add-delete-divide form, of which one is minimal. Thus as the initial add-divide sequence is minimal by construction, and all division sequences are minimal, this sequence is a min-edit-sequence from G to G' with split relation f as required.

3.4 Representation of optimal Cluster Graph edit-sequences by coverings

Consider the CEVS problem for a graph G and an edit distance k. If there is a solution edit-sequence S for this problem, we may also represent the resulting graph G' by a covering of the vertices in the original graph. As G' is a cluster graph every pair of vertices from a clique are joined by an edge, and we can reconstruct G' and $f : V(G) \to 2^{V(G')}$, the corresponding vertex relation. And so by Lemma 7 we can represent the search space for optimal CEVS edit-sequences by coverings of the vertex set of G, and evaluate the min-edit distance for each of them in $O((|V'| - |V|)\Delta(G) + |V| + |E| + |V'| + |E'|)$ time.

4 Critical Cliques

Originally introduced by Lin et al. [21], critical cliques provide a useful tool in understanding the clusters in graphs. A *critical clique* of a graph $G = (V, E)$ is a maximal induced subgraph C of G such that:

– C is a complete graph.
– There is some subset $U \subseteq V$ such that for every $v \in V(C)$, $N[v] = U$.

It was shown in [21] that each vertex is in exactly one critical clique. Let the critical clique containing a vertex v be denoted by $CC(v)$. The critical clique graph $CC(G)$ can then also be defined as a graph with vertices being the critical cliques of G, having edges wherever there is an edge between the members of the critical cliques in the original graph [21]. That is to say, that the critical clique graph $G' = (V', E')$ related to the graph $G = (V, E)$ is the graph with $V' = CC(G)$ and edges $E' = \{uv | \forall x \in V_C(u). \forall y \in V_C(v). xy \in E\}$ Furthermore, the vertices in G' are given as a weight the number of vertices they represent in the original graph, similarly for the edges.

The following lemma, dubbed "the critical clique lemma" is adapted from Lemma 1 in [18], with a careful restatement in the context of this new problem.

Lemma 8. *Any covering $C = (S_1 \dots S_l)$ corresponding to a solution to* CEVS *for a graph $G = (V, E)$ that minimizes k will always satisfy the following property: for any $v \in G$, and for any $S_i \in C$ either $CC(v) \subseteq S_i$ or $CC(v) \cap S_i = \emptyset$.*

Proof omitted for length reasons.

By the critical clique lemma, the CEVS problem is equivalent to a weighted version of the problem on the critical clique graph.

Lemma 9. *If there is a solution to* CEVS *on (G, k) then there are at most $4k$ non-isolated vertices in $CC(G)$. Moreover, there are at most $3k+1$ vertices in any connected component of $CC(G)$ and there are at most k connected components in $CC(G)$ which are non-isolated vertices.*

Proof. l. We follow an approach similar to that taken by Fellows et al. and Gou. Let S_{opt} be an optimal solution of CEVS and partition the vertex set of $CC(G)$ into 4 sets W, X, Y and Z. Let W be the set of vertices which are the endpoint of some edge added by S_{opt}, Let X be the subset of vertices which are the endpoint of some edge deleted by S_{opt} and not in W, Let Y be the subset of vertices which are split by S_{opt} and not in $W \cup X$, finally let Z be all other non-isolated vertices in G. As each vertex in W, X, and Y is affected by some operation in S_{opt} and any operation in S_{opt} can affect at most 2 vertices, if $|S_{opt}| < k$ then $|W \cup X \cup Y| < 2k$. Let us now consider Z. Suppose that $u, v \in V_{S[opt]} \cap Z$ are in the same clique in $G_{S_{opt}}$, then as they are in Z they are adjacent to exactly every vertex in Y as they are not involved in any edge addition, deletion or vertex splitting. However as they are adjacent to each other and have the same neighborhood, apart from each other, they are in a critical clique together. i.e. $u = v$. Thus there can be at most one vertex in Z for any connected component of $G_{S_{opt}}$.

Now every vertex in Z is adjacent to a vertex in $W \cup X \cup Y$. To see this suppose that $z \in Z$ is not, then by the above lemma it is the only vertex in Z in its clique in $G_{S_{opt}}$, however as it is in Z then it has neither been split nor been severed from any vertex, thus it is an isolated vertex of $CC(G)$. This is a contradiction.

As each connected component of $G_{S_{opt}}$, which is not an isolated vertex in $CC(G)$, contains at least one vertex of $W \cup X \cup Y$ there are at most $2k$ vertices in Z and there are at most $4k$ non-isolated vertices in $CC(G)$.

As each connected component $CC(G)$ has to be separated into at most $k+1$ cliques in $G_{S_{opt}}$ there can be at most $k+1$ elements of Z in any connected component of $CC(G)$, thus there can be at most $3k+1$ vertices in any connected component of $CC(G)$.

As no clique in $G_{S_{opt}}$ has members from two connected components of G, and as if any connected component component of G is not an isolated vertex in $CC(G)$ there is at least one edit performed to some vertex in that connected component there can be at most k such connected components which are non isolated vertices in $CC(G)$

5 A $4k(k+1)$ vertex kernel

From the result in Section 4 we can devise a polynomial size kernel for CEVS. To achieve this we propose three reduction rules, prove that they are valid, and that their application gives a kernel as required.

Reduction Rule 1 *Remove all isolated Cliques.*

Lemma 10. *Reduction Rule 1 is sound.*

Proof. As no optimal solution has a final clique which bridges two connected components of a graph G and an isolated clique needs no edits to make it complete, the clique will remain in all optimal solutions and as such can be removed without affecting the result.

Reduction Rule 2 *Reduce all critical cliques with more than $k+1$ vertices to $k+1$ vertices.*

Lemma 11. *Reduction Rule 2 is sound.*

Proof. As no solution clique in an optimal solution partially contains a critical clique and the cost to delete the edges incident to, or add edges to all of, or to divide the vertices of such a critical clique is greater than k is too high to be allowed we only need to maintain this invariant. Thus we can remove vertices from any critical clique with more than $k+1$ vertices until there are at most $k+1$ vertices in a critical clique and this will not affect the result.

Reduction Rule 3 *If there are more than $4k$ non-isolated critical cliques reduce the graph to a P_3 and set $k=0$ in this case.*

Lemma 12. *Reduction Rule 3 is sound.*

Proof. As proved in Lemma 9 if there are more than $4k$ non-isolated critical cliques then there is no solution, thus we can emit a trivial No-instance.

Theorem 2. *There exists a polynomial-time reduction procedure that takes an arbitrary instance of the* Cluster Editing with Vertex Splitting *problem and produces an equivalent instance whose order (number of vertices) is bounded above by $4k(k+1)$. In other words, CEVS admits a quadratic-order kernel.*

Proof. As shown in the previous lemmas, reduction rules 1, 2 and 3 are well founded, and as after having applied them exhaustively, there are at most $4k$ critical cliques in the input graph. Due to reduction rule 2, there is no critical clique with more than $k+1$ vertices, and therefore, the application of these reduction rules results in a $4k(k+1)$ vertex kernel.

6 Conclusion

By allowing a vertex to split into two vertices we extend the notion of Cluster Editing in an attempt to better-model clustering problems where a data element may have roles in more than one cluster. The corresponding new version of Cluster Editing is shown to be fixed-parameter tractable via a new approach that is based on edit-sequence analysis.

The vertex splitting operation may also be applicable to other classes of target graphs, including bipartite graphs, disjoint complete bipartite graphs (bi-cluster graphs), chordal-graphs, comparability graphs, and perfect graphs. The results in Section 3 are directly applicable to these other classes, and it may be possible to find an analog to the critical clique lemma for bi-cluster graphs.

The work reported in this paper did not consider the exclusive version of vertex splitting, where the two vertices which result from a split must additionally have disjoint neighborhoods. This is ongoing research at this stage.

Acknowledgments: Serge Gaspers is the recipient of an Australian Research Council (ARC) Future Fellowship (FT140100048) and acknowledges support under the ARC's Discovery Projects funding scheme (DP150101134). Alexis Shaw is the recipient of an Australian Government Research Training Program Scholarship.

References

1. Abu-Khzam, F.N., Baldwin, N.E., Langston, M.A., Samatova, N.F.: 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)
2. Abu-Khzam, F.N.: On the complexity of multi-parameterized cluster editing. Journal of Discrete Algorithms 45, 26–34 (2017)
3. Abu-Khzam, F.N., Fernau, H.: Kernels: Annotated, proper and induced. In: Parameterized and Exact Computation, Lecture Notes in Computer Science, vol. 4169, pp. 264–275. Springer Berlin Heidelberg (2006)
4. Böcker, S.: A golden ratio parameterized algorithm for cluster editing. Journal of Discrete Algorithms 16, 79 – 89 (2012)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences 75(8), 423 – 434 (2009)
6. Cai, L.: Fixed-parameter tractability of graph modification problems for hereditary properties. Information Processing Letters 58(4), 171–176 (May 1996)
7. Chen, J., Meng, J.: A 2k kernel for the cluster editing problem. Journal of Computer and System Sciences 78(1), 211–220 (2012)
8. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong computational lower bounds via parameterized complexity. Journal of Computer and System Sciences 72(8), 1346 – 1367 (2006)
9. Cygan, M., Fomin, F.V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)

10. D'Addario, M., Kopczynski, D., Baumbach, J., Rahmann, S.: A modular computational framework for automated peak extraction from ion mobility spectra. BMC Bioinformatics 15(1), 25 (2014)
11. Dehne, F., Langston, M.A., Luo, X., Pitre, S., Shaw, P., Zhang, Y.: The cluster editing problem: Implementations and experiments. In: Parameterized and Exact Computation, pp. 13–24. Springer Berlin Heidelberg (2006)
12. Downey, R.G., Fellows, M.R.: Fundamentals of Parameterized Complexity. Texts in Computer Science, Springer London (2013)
13. Fadiel, A., Langston, M.A., Peng, X., Perkins, A.D., Taylor, H.S., Tuncalp, O., Vitello, D., Pevsner, P.H., Naftolin, F.: Computational analysis of mass spectrometry data using novel combinatorial methods. AICCSA 6, 8–11 (2006)
14. Fellows, M., Langston, M., Rosamond, F., Shaw, P.: Efficient parameterized preprocessing for cluster editing. In: Fundamentals of Computation Theory. pp. 312–321. Springer (2007)
15. Fellows, M.R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Graph-based data clustering with overlaps. Discrete Optimization 8(1), 2 – 17 (2011), parameterized Complexity of Discrete Optimization
16. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer (2006)
17. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. Theory of Computing Systems 38(4), 373–392 (2005)
18. Guo, J.: A more effective linear kernelization for cluster editing. Theoretical Computer Science 410(8-10), 718 – 726 (2009)
19. Hüffner, F., Komusiewicz, C., Moser, H., Niedermeier, R.: Fixed-parameter algorithms for cluster vertex deletion. Theory of Computing Systems 47(1), 196–217 (2010)
20. Křivánek, M., Morávek, J.: NP-hard problems in hierarchical-tree clustering. Acta Informatica 23 (3), 311–323 (1986)
21. Lin, G.H., Kearney, P.E., Jiang, T.: Phylogenetic k-root and Steiner k-root. In: Algorithms and Computation, pp. 539–551. Springer (2000)
22. Niedermeier, R.: An Invitation to Fixed-Parameter Algorithms. Oxford University Press (2006)
23. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: Popular nearest neighbors in high-dimensional data. Journal of Machine Learning Research 11(Sep), 2487–2531 (2010)
24. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics 144(1-2), 173–182 (Nov 2004)
25. Tomašev, N., Radovanović, M., Mladenić, D., Ivanović, M.: The role of hubness in clustering high-dimensional data. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 183–195. Springer (2011)