Cluster Editing with Diamond-free Vertices

Damiaan Rhebergen*

Delft University of Technology

June 2021

Abstract

The Cluster Editing problem asks to transform a graph into a union of disjoint cliques in the minimum number of edge additions and deletions. The Cluster Deletion problem has the same goal, but allows edge deletions only. Recently, a connection between Cluster Editing on *diamond-free* graphs and Cluster Deletion was established. In this paper we examine this connection in depth and show that for vertices that are not part of a *diamond* subgraph, it suffices to consider only edge deletions. In doing so, we provide techniques that can be used to study properties of optimal solutions of Cluster Editing, and prove the existence of a *diamond-free* subgraph for which an optimal solution consisting of edge deletions only can be used as part of the solution to the whole graph.

Keywords: Cluster Editing, Cluster Deletion, Clustering, diamond-free Graphs, NP-hard problem

1 Introduction

The NP-hard Cluster Editing problem is a widely studied problem in complexity theory. Given a graph G, the goal is to transform G into a union of disjoint cliques by the minimum number of edge modifications. Here an edge modification means that we either delete an edge that is in G or insert an edge that is not in G. A union of disjoint cliques is called a cluster graph.

Clustering plays an important part in solving many real-world problems. In the field of computational biology, for instance, the weighted variant of Cluster Editing has been used to model the clustering of proteins and genes [1, 2]. Thus, finding ways to solve this problem efficiently could provide real benefit for these areas of study.

Unfortunately however, Cluster Editing has been shown to be NP-hard numerous times [3, 4], making it unlikely that an efficient algorithm for general graphs will be found. Still, a lot of progress has been made in mapping out the difficulty of the problem.

A problem is *fixed parameter tractable* (FPT) if there exists some parameter, that can be used as extra information in order to solve the problem in polynomial time. It is known that

*Email address: d.d.rhebergen@student.tudelft.nl

Cluster Editing is FPT when we take the optimal solution size k, that is the minimum number of edge modifications to transform a graph into a cluster graph, as parameter [5]. This means that Cluster Editing can be solved in polynomial time if already we know the optimal solution size k beforehand. Komusiewicz and Uhlmann have shown that if the maximum number of clusters d in the solution graph is given as parameter Cluster Editing is still NP-hard, and that the same is true for the maximum number of edge modifications t per vertex as parameter. They have also shown, however, that if these parameters are taken together, Cluster Editing is FPT [6].

Motivated by results from parameterized Vertex Cover, van Bevern et al studied the parameterization above some guaranteed lower bound of the solution size for Cluster Editing [7]. The idea here is that if a lower bound that is near the optimal solution size can be found efficiently, this bound could be exploited by algorithms that rely on iteratively trying higher values for the optimal solution size as a parameter.

Currently, the fastest know theoretical algorithm is due to Bökker and runs in $O(1.62^k + n + m)$ time, where k is the size of the optimal solution, and n and m are the size of the vertex set and the size of the edge set respectively [8].

For Cluster Editing, a *kernelization* algorithm transforms a graph G with with parameter k into a graph G' and parameter k' in polynomial time and in such a way that G can be transformed into a cluster graph by k edge modifications if and only if G' can be solved into a cluster graph by k' edge modifications. G' and k' then form a *kernel* of G and k. Ideally, the algorithm ensures the size of the kernel less than the size of the original instance.

The best known kernelization algorithms for Cluster Editing return a kernel with size $O(2k)$ [9, 10].

A problem closely related to Cluster Editing (CE) is the (also NP-hard) Cluster Deletion (CD) problem, which asks to transform G into a cluster graph by the minimum number of edge *deletions* only. Malek and Naanaa have shown that, for a graph that does not have a so called "diamond" as a subgraph, an optimal solution for CD is also an optimal solution for CE [11]. That is, for "diamond-free" graphs, we can get an optimal solution for CE just by deleting existing edges.

This raises the natural question of what structure in a graph makes an edge addition more optimal than just deleting edges. In other words, *to what extent do the optimal solutions of* Cluster Editing *and* Cluster Deletion *coincide?*

The exploration of this question is the main focus of this paper. The result on diamond-free graphs from [11] suggests that diamonds play a key role here. We present several observations about graphs that do contain diamonds, but also have a subgraph that is diamond-free. We show edges adjacent to vertices that are not part of any diamond can be treated as if we are solving the Cluster Deletion problem. Moreover, we show that in such graphs, there exists a subgraph containing all vertices that are not part of any diamond that can be solved optimally as part of the optimal solution of the whole graph. Note that, in general, an optimal solution to a subgraph is not included in an optimal solution to the whole graph [10].

The rest of the paper is structured as follows: The Section 2 is used to establish the notation used in this paper, and to give some preliminary definitions and results that will be used in

later sections. Then, in Section 3, we provide several Lemmas that give structure to clusters of optimal solutions, which can be used to prove more properties of optimal solutions. Section 4 contains the main results of this paper. Section 5 gives a direct consequence of these results. In Section 6 we discuss the results and their consequences, and the questions that arise and could not be answered here are stated and discussed. In Section 7 we briefly discuss responsible research and any ethical implications. Finally, in Section 8 we give a summary and end with some concluding remarks.

2 Preliminaries

2.1 Notation

In this paper, all graphs are undirected and simple. Let G be a graph. Then $V(G)$ denotes the set of vertices in G, and $E(G)$ denotes the set of edges in G.

Two vertices $u, v \in V(G)$ are said to be *adjacent* if $\{u, v\} \in E(G)$, that is, if there is an edge between them.

The set of all vertices adjacent to a vertex $v \in V(G)$ is called the *neighborhood* of v and is denoted as $N_G(v)$. The *closed* neighborhood of v is defined as

$$N_G[v] = N_G(v) \cup \{v\}.$$

The *distance* between two vertices $u, v \in V(G)$ is equal to the number of edges in the shortest path between u and v in G, and is denoted by $d_G(u, v)$. If there can be no confusion about the graph, the subscript G may be omitted for the neighborhood and distance.

G is *complete* if every two vertices in G are adjacent, that is $N[v] = V(G)$ for all $v \in V(G)$ or, alternatively, $d(u, v) = 1$ for all $u,\ v \in V(G)$.

For a subset $W \subseteq V(G)$, let $G[W]$ denote the subgraph of G *induced* by W, where

$$V(G[W]) = W \quad \text{and} \quad E(G[W]) = \{\{u, v\} \in E(G) :\ u, v \in W\}.$$

$\overline{W} = V(G) \setminus W$ denotes the *complement* of W and the neighborhood of W is defined as

$$N_G(W) = \bigcup_{v \in W} N_G(v) \setminus W,$$

with the closed neighborhood as

$$N_G[W] = \bigcup_{v \in W} N_G[v].$$

With $E(W, \overline{W})$ we denote the set of edges that has one endpoint in W and the other in $\overline{W}$

A *clique* in G is an induced subgraph that is complete. A clique of size n is denoted by K_n, a simple path of on n vertices is called a P_n and a simple cycle with n vertices is denoted C_n.

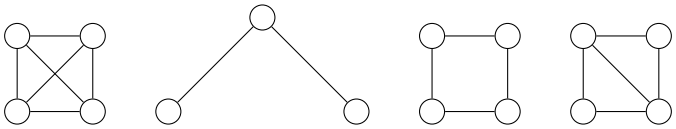

Figure 1: From left to right: K_4, P_3, C_4 and diamond

A graph with four vertices that misses only one edge of a K_4 is called a *diamond* (figure 1).

For a graph F, a graph G is said to be *F-free* if it does not contain F as a vertex induced subgraph. For instance, if G is diamond-free, then no set of four vertices in $V(G)$ induce a diamond in G.

For a graph G, let $S \subseteq V \times V$. Define $G \triangle S$ as the graph with

$$V(G \triangle S) = V(G)$$

and

$$E(G \triangle S) = E(G) \triangle S = (E(G) \setminus S) \cup (S \setminus E(G)).$$

The set S is called an *edge editing set*. S is a *solution* to G, if $G \triangle S$ is a union of disjoint cliques.

We denote an optimal solution to a graph G with $Opt(G)$. The size of $Opt(G)$ is denoted by $\omega(G)$.

An important characterization of a cluster graph is that it cannot have a P_3 as an induced subgraph. That is, G is a cluster graph if and only if it is a P_3-free graph.

2.2 Preliminary results

We now reference some preliminary results that come from other research and will be used throughout this paper.

First, we state explicitly the result from Malek and Naanaa about the correspondence between Cluster Deletion and Cluster Editing.

Lemma 2.1. *[11] Every diamond-free graph admits an optimal* Cluster Editing *solution with no added edges.*

A nice consequence of this correspondence of optimal solutions of CD and CE on diamond-free graphs is that several hardness results for CD carry over to CE on diamond-free graphs. In [11], for instance, it is shown that the polynomial time solvability of Cluster Deletion on (butterfly, diamond)-free graphs carries over to Cluster Editing, since Cluster Editing can be solved on diamond-free graphs by deleting edges only.

The next result is due to Bastos et al. [12]. It states that two vertices that are distance 3 apart from each other can never be in the same cluster if we cluster optimally.

Lemma 2.2. *[12] Let G be an undirected graph and let $u, v \in V(G)$ such that $d_G(u, v) \geq 3$. Then in any optimal solution S of G, vertices u and v belong to distinct clusters.*

This becomes more clear if we take into account that two vertices with distance 3 do not share any neighbors. Clustering the two vertices together then, would require an additional edge modification for all of the neighbors of both vertices; either we need to add in order to include it in the cluster, or we delete an edge to remove it from the cluster.

This last result will be used to prove properties of optimal solutions for Cluster Editing. Indeed, for any cluster C that is part of the graph $G \triangle S$, where S is some optimal solution to G, we can now assume, without loss of generality, that C does not contain two vertices that have distance 3 in G. Otherwise, S would not be optimal by Lemma 2.2. This gives as some structure to work with in trying to prove other properties.

3 Clustering Lemmas

In this section we add to the structure of optimal solutions that we started at the end of the last section. We do this by giving three more Lemmas that allow us to, for an optimal solution S to a graph G, restrict the clusters in $G \triangle S$ to only include vertices that fulfill certain conditions with regards to the other vertices in that cluster.

We start with a Lemma that generalizes Lemma 2.2 to include to possibility of two vertices sharing a single neighbor[1].

Lemma 3.1. *Let G be a graph, and $u, v, w \in V(G)$ such that $|N_G(u) \cap N_G(v)| \leq 1$. Then there exists an optimal solution where u and v are not in the same cluster.*

Proof. Observe that Lemma 2.2 already proves the case where $|N_G(u) \cap N_G(v)| = 0$. Thus, we assume that $|N_G(u) \cap N_G(v)| = 1$.

Let $w \in N_G(u) \cap N_G(v)$. Let S be a solution that puts u and v in the same cluster C in $G \triangle S$ and let S be optimal under that condition. We will show that we can find a solution S' that is at least as good as S, where u and v are not in the same cluster in $G \triangle S'$. Cluster C can be partitioned as follows: $C = C_0 \cup C_u \cup C_v \cup C_w \cup C_{u,w} \cup C_{v,w} \cup \{u, v, w\}$, where:

- $C_0 = \{x \in C \setminus \{u, v, w\} | \{x, u\} \notin E(G), \{x, v\} \notin E(G), \{x, w\} \notin E(G)\}$
- $C_v = \{x \in C \setminus \{u, v, w\} | \{x, u\} \notin E(G), \{x, v\} \in E(G), \{x, w\} \notin E(G)\}$
- $C_w = \{x \in C \setminus \{u, v, w\} | \{x, u\} \notin E(G), \{x, v\} \notin E(G), \{x, w\} \in E(G)\}$
- $C_{u,w} = \{x \in C \setminus \{u, v, w\} | \{x, u\} \in E(G), \{x, v\} \notin E(G), \{x, w\} \in E(G)\}$
- $C_{v,w} = \{x \in C \setminus \{u, v, w\} | \{x, u\} \notin E(G), \{x, v\} \in E(G), \{x, w\} \in E(G)\}$

Observe that by Lemma 2.2 and optimality of S, it holds that for all $x \in C_u \cup C_{u,w}$ ($y \in C_v \cup C_{v,w}$), $d_G(x, v) = 2$ ($d_G(y, u) = 2$). It follows that $(C_u \cup C_{u,w}) \cap (C_v \cup C_{v,w}) = \emptyset$, since u and v share no neighbors except for w. Assume, without loss of generality, that $|C_u \cup C_{u,w}| \geq |C_v \cup C_{v,w}|$. Consider the solution S' that differs from S only in that it isolates v from C, and compare the solution size of S and S':

$$\begin{aligned} |S| - |S'| =& 3|C_0| + 2|C_u| + 2|C_v| + |C_{u,w}| + |C_{v,w}| + |\{\{u, v\}\}| \\ & - (2|C_0| + |C_u + 3|C_v + 2|C_{v,w}| + |\{\{v, w\}\}| \end{aligned}$$

[1] It was later discovered that this Lemma was already proven by Komusiewicz and Uhlmann [13]

$$\begin{aligned} |S| - |S'| &= |C_0| + |C_u| + |C_{u,w}| - (|C_v| + |C_{v,w}|) \\ &\geq 0 \end{aligned}$$

So S' is at least as good as S. This implies that there exists a optimal solution that does not put v_1 and v_2 in the same cluster. □

Note that the lemma does not exclude the possibility of an optimal solution that does put the two vertices in the same cluster, but merely states that an optimal solution exists that does not put them in the same cluster.

As it turns out, the mere existence of such a solution is enough to provide the structure we need. This is because, in subsequent proofs of other properties, we just use the size of an optimal solution, assume that it does not fulfill the property, and then proceed to give a new solution that does fulfill the property and is guaranteed to be just as good as the optimal solution. Since it does not matter which optimal solution we take for this, we can simply assume that the condition of Lemma 3.1 holds.

In the proof of Lemma 3.1, we examined a single cluster of the solution. If we do this again for a different proof in which we want to use the structure that Lemma 3.1 gives, we need to make sure that the property does not only hold for the whole graph, but also in the subgraph that is induced by the cluster. Otherwise, it could well me the case that two vertices do indeed share more than one neighbor, but that these neighbors are not in the same cluster and thus can not add to the structure of the cluster.

The next two Lemmas ensure that if two vertices with distance 2 end up in the same cluster, we can always assume that at least two common neighbors also end up in that cluster.

Lemma 3.2. *Let G be a graph and let $u, v \in V(G)$ such that $d_G(u, v) = 2$. Then any optimal solution that puts u and v in the same cluster will also put at least one common neighbor of u and v in that cluster.*

Proof. Let $\omega(G)$ be the optimal solution size for G. Let S be a solution that puts u and v in a cluster that does not include any of their common neighbors. Assume for contradiction that S is optimal, that is, $|S| = \omega(G)$. Let $E_{u,v} = \{\{u, w\}, \{v, w\} \in E(G) : w \in N_G(u) \cap N_G(v)\}$. Then $E_{u,v} \subseteq S$. Let G' be the graph with exactly those edges missing. Note then, that $\omega(G') = \omega(G) - |E_{u,v}|$, since some optimal solution of G had those edges as a subset.

Let $S' = S \setminus E_{u,v}$. Note that S' is a solution of G'. Observe however, that $d_{G'}(u, v) \geq 3$, since they have no common neighbor anymore in G'. By Lemma 2.2, we get that no optimal solution of G' will put u and v in the same cluster, and thus S' cannot be an optimal solution for G'. Thus, $|S'| > \omega(G') = \omega(G) - |E_{u,v}|$. But then $|S| = |S'| + |E_{u,v}| > \omega(G)$, contradicting that S is an optimal solution to G. □

Lemma 3.3. *Let G be a graph and let $u, v \in V(G)$ such that $d_G(u, v) = 2$. For any solution that puts u and v in the same cluster with exactly one of its neighbors, there is a solution that is at least as good that does not put u and v in the same cluster.*

Proof. Let S be a solution that puts u and v in the same cluster C with exactly one vertex that is the neighbor of both u and v in G. Then, in the subgraph $G[C]$, u and v have exactly one neighbor, and thus by Lemma 3.1 there is an optimal solution for $G[C]$ such that u and v are not in the same cluster.

Let $S_{G[C]}$ denote the solution S restricted on $G[C]$. Let S_1 be an optimal solution to $G[C]$ that

does not keep u and v in the same cluster. Consider the solution $S' = (S \setminus S_{G[C]}) \cup S_1$. First note that this is actually a solution. Indeed, $G \triangle (S \setminus S_{G[C]})$ deletes all the edges with one endpoint in C and the other in $\overline{C}$ and transforms $G[\overline{C}]$ into a cluster graph. Then, since S_1 is a solution to $G[C]$, $G[C] \triangle S_1$ is a cluster graph. Therefore, S' is a solution to G. Second, to see that S' is at least as good as S, observe that $|S'| = |(S \setminus S_{G[C]}) \cup S_1| = |S| - |S_{G[C]}| + |S_1| \leq |S|$ since $|S_1| \leq |S_{G[C]}|$. □

With these Lemmas we have a decent amount of structure at our disposal, and can now turn our attention to the main results of this paper.

4 Diamond-free Vertices

In this section, we look to generalize the result of Lemma 2.1 from [11] to general graphs. What we are looking for are sufficient conditions for vertices to put those vertices in different clusters. For this, we already have two conditions in Lemma 2.2 and Lemma 3.1, but Lemma 2.1 suggests that we can take this further by looking specifically at diamonds.

To this end, we provide a new Lemma which states that a vertex that is not in any diamond will not end up in a cluster with vertices it is not connected to. Such vertices are from now on called *diamond-free* vertices.

Definition 4.1. *Let G be a graph, and $v \in V(G)$. If v is not in any diamond of G then v is called a **diamond-free vertex** of G.*

These diamond-free vertices have the following property that will be useful in proving the main Lemma of this section.

Property 4.2. *For a graph G, let $v \in V(G)$ be a diamond-free vertex. Then for $w_1, w_2, w_3 \in N(v)$, if w_1 is adjacent to w_2, and w_2 is adjacent to w_3, then w_1 is adjacent to w_3.*

Proof. Let $w_1, w_2 \in N(v)$ be two adjacent neighbors of v. Assume for contradiction that w_2 is adjacent to some $w_3 \in N(v)$, while w_1 is not adjacent to w_3. Then the vertex set $\{v, w_1, w_2, w_3\}$ forms a diamond in G, contradicting the condition that v was not in any diamond. □

What this property gives us is that the neighborhood of a diamond-free vertex induces a union of disjoint cliques in G. With this in mind, we give the main result of this paper.

Lemma 4.3. *Let G be a graph. Let v be a diamond-free vertex, and let $U = \{u \in V(G) : d_G(u, v) = 2\}$. Then there exists an optimal solution to G that will not put v in a cluster with any vertex $u \in U$.*

Proof. Let S be a solution that puts v in a cluster C with some set of vertices U such that $d(u, v) = 2$ for all $u \in U$ and let it be optimal under this condition. Without loss of generality, assume that for all $u \in C$ with $d_G(u, v) = 2$, $u \in U$. That is, the vertices in U are the only vertices in C that have distance 2 from v (This assumption can be made, because U is arbitrary).

Define $C_v = \{v' \in C : v' \in N(v) \text{ and } \forall u \in D,\ v' \notin N(u)\}$.

Since S is optimal under the condition that u and v are in the same cluster, Lemma 2.2 gives us that for all $u \in U,\ v' \in C_v,\ d_G(u, v') = 2$ and by Lemma 3.2 we get that at least one

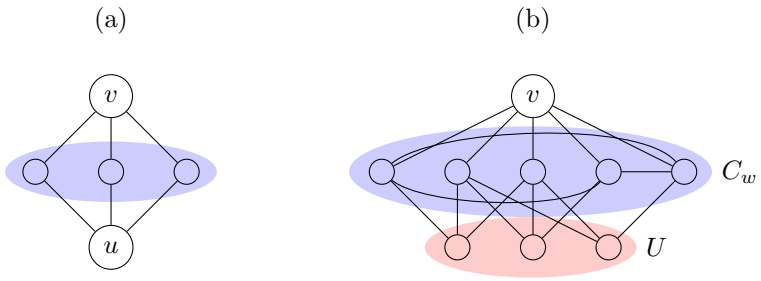

Figure 2: The two cases for Lemma 4.3. (a) is Case 1, (b) is Case 2

neighbor of u and v' must be in C, that is, $|N_G(u) \cap N_G(v')| \geq 1$.

For all $u \in U$ define

- $C_u = \{u' \in C : u' \in N(u),\ u' \notin N(v)$
- $C_{u,v} = \{w \in C : w \in N(v),\ w \in N(u)\}$
- $C_w = \bigcup_{u \in U} C_{u,v}$

Note that Lemma 3.1 already proves the case where $|C_w| = 1$.

Claim. $C_v = \emptyset$.
Note then for each $u \in U$ that $C_u \subseteq U \setminus \{u\}$. Indeed, if $u' \in C_u$, then u' is not adjacent to v and thus by Lemma 2.2 and optimality of S $d_{G[C]}(u', v) = 2$. Since we assumed that all vertices in the cluster with distance 2 from v are in U, this gives that $u' \in U$. It follows then that for $v' \in C_v$ and $u' \in C_u$, there can be no edge from v' to u' because then v' would be in the neighborhood $u' \in U$.

This observation gives us that v' can only have edges to vertices in $C_v \cup C_w$ (excluding of course its edge to v). In fact, since $d_{G[C]}(u, v') = 2$ for all $u \in U$, $v' \in C_v$, v' has to have an edge to some $w \in C_w$. Moreover, by Property 4.2 v' can only have an edge to w if either both w and v' don't have any other neighbors in $C_v \cup C_w$, or they share all neighbors in $C_v \cup C_w$.

Lastly, observe for any $u \in U$, a vertex $w_1 \in C_{u,v}$ can only have an edge with $w_2 \in C_w$ if $w_2 \notin C_{u,v}$ (since otherwise the vertex set $\{u, v, w_1, w_2\}$ would form a diamond in G) and thus w_2 is not adjacent to u.
Putting all of the above together, we find that for each $v' \in C_v$ and $u \in U$, $|N_{G[C]}(u) \cap N_{G[C]}(v')| = 1$ and thus by Lemma 3.3 we can always choose S is such a way that $C_v = \emptyset$. This proves the claim.

We divide into the case where $|U| = 1$ and the case where $|U| \geq 2$ (see figure 2).

Case 1. $|U| = 1$.
Let $u \in U$ be the only vertex with distance 2 from v in C. Observe that in this case, there can be no edge between two $w_1, w_2 \in C_w$, otherwise $\{u, v, w_1, w_2\}$ would induce a diamond in

G. The size of the solution S that put $u,\ v$ and the vertices of C_w in the same cluster is thus given by

$$|S| = \ |\{u, v\}| + \binom{|C_w|}{2} + |S_0|,$$

where $S_0 \subseteq S$ with the edges with at least one endpoint in $\overline{C}$.

Let $w_1, w_2 \in C_w$ with $w_1 \neq w_2$. Consider instead the solution S' to G that differs from S only in that it solves $G[C]$ by deleting all edges from u to C_w and v to C_w except for $\{u, w_1\}$ and $\{v, w_2\}$. Note that S' is indeed a solution, since $G[C] \triangle S'$ consists of isolated vertices and two cliques of size 2, while $G[\overline{C}] \triangle S'$ is exactly the same as $G[\overline{C}] \triangle S$ and S' deletes all edges between C and $\overline{C}$, giving that $G \triangle S'$ is a cluster graph. The size of S' is given by

$$|S'| = \ 2|C_w| - 2 + |S_0|.$$

Letting $n = |C_w|$, the difference in size between S and S' is

$$\begin{aligned} |S| - |S'| &= \ |\{u, v\}| + \binom{n}{2} - 2n + 2 \\ &= \ \frac{1}{2}n^2 - \frac{5}{2}n + 3. \end{aligned} \tag{1}$$

The roots of (1) are at $n = 2$ and $n = 3$, which gives us that for all integer $n \geq 1$,

$$|S| - |S'| \geq \ 0.$$

This proves Case 1.

Case 2. $|U| \geq 2$.
For $w \in C_w$ define

- $C_U^w = \{u \in U : \{u, w\} \in E(G)\}$
- $X_U^w = \{u \in U : \{u, w\} \notin E(G)\} = U \setminus C_U^w$

Observe that for $w_1 \in C_{u_1,v}$ and $w_2 \in C_{u_2,v}$, if w_1 and w_2 share an edge, then $u_1 \neq u_2$. Otherwise the vertex set $\{u_1, v, w_1, w_2\}$ would induce a diamond in G. Furthermore, if w_1 and w_2 share an edge, and w_2 also shares an edge with $w_3 \in C_w$ ($w_1 \neq w_3$), then by Property 4.2 w_1 and w_3 must also share an edge. It follows that $G[C_w]$ is a union of disjoint cliques, where two vertices in the same clique do not share neighbors in U.

Therefore, define the following equivalence relation: For $w_1, w_2 \in C_w$, we say that $w_1 \sim w_2$ if and only if w_1 and w_2 are in the same clique in $G[C_w]$. Then for $a \in C_w$, $[a]$ denotes the equivalence class $\{w' \in C_w : w' \sim a\}$. Let $p_a = |[a]|$ denote the number of vertices that are equivalent to a for all $a \in C_w/\sim$.
Observe now that the amount of missing edges between in $G_[C_w]$ is equal to

$$\binom{n}{2} - \sum_{[a] \in G[C_w]/\sim} \binom{p_a}{2}.$$

Let $w_1, w_2 \in C_w$ such that $w_1 \sim w_2$ and $w_1 \neq w_2$. Recall that, since w_1 and w_2 share an edge, they cannot be adjacent to the same $u \in U$. Therefore, $C_U^{w_1} \cup C_U^{w_2} = \emptyset$ and thus $X_U^{w_1} \cup X_U^{w_2} = U$. It follows that

$$|X_U^{w_1}| + |X_U^{w_2}| = |U| + |X_U^{w_1} \cap X_U^{w_2}|.$$

The size of S is then given by

$$\begin{aligned}
|S| &= |U| + \sum_{\substack{[a],[b] \in G[C_w]/\sim \\ a \not\sim b}} p_a p_b + \sum_{[a] \in G[C_w]/\sim} \sum_{i=1}^{p_a - 1} (|X_U^w w_{2i-1}| + |X_U^w w_{2i}|) + \\
&\quad |S_0| \\
&= |U|(1 + \sum_{[a] \in G[C_w]/\sim} \frac{p_a}{2}) + \sum_{\substack{[a],[b] \in G[C_w]/\sim \\ a \not\sim b}} p_a p_b + \\
&\quad \sum_{[a] \in G[C_w]/\sim} \sum_{i=1}^{p_a - 1} |X_U^{w_{2i-1}} \cap X_U^{w_{2i}}| + |S_0|,
\end{aligned}$$

where $S_0 = \{\{x, y\} \in S : x, y \notin C_w\} \cup \{\{x, y\} \in S : x \notin C_w,\ y \notin U \cup \{v\}$.

Consider now the solution S' to G that deletes all edges between C_w and U, and deletes all edges from v to C_w except for the edges to the vertices that form the largest clique in $G[C_w]$.

Let $p = \max_{[a] \in G[C_w]/\sim} p_a$ be the size of the largest clique in $G[C_w]$ and a_{max} be any representative vertex for some equivalence class of size p. If there are multiple cliques of size p, then S' deletes the edges from v to all but one of those cliques. The size of S' is given by

$$|S'| = 2|C_w| - p.$$

The difference between the sizes of S and S' is then given by

$$\begin{aligned}
|S| - |S'| = \ & |U|(1 + \sum_{[a] \in G[C_w]/\sim} \frac{p_a}{2}) + \sum_{\substack{[a],[b] \in G[C_w]/\sim \\ a \not\sim b}} p_a p_b \ + \\
& \sum_{[a] \in G[C_w]/\sim} \sum_{i=1}^{p_a - 1} |X_U^{w_{2i-1}} \cap X_U^{w_{2i}}| \ - \\
& 2|C_w| + p.
\end{aligned}$$

Let $|C_w| = n$, and recall that $|U| \geq 2$ and we can assume $n \geq 2$ by Lemma 3.3. Then

$$\begin{aligned}
|S| - |S'| \geq\ & 2 + \sum_{[a] \in G[C_w]/\sim} p_a + \sum_{\substack{[a],[b] \in G[C_w]/\sim \\ a \not\sim b}} p_a p_b\ + \\
& \sum_{[a] \in G[C_w]/\sim} \sum_{i=1}^{p_a - 1} |X_U^{w_{2i-1}} \cap X_U^{w_{2i}}|\ + \\
& - 2n + p \\
\geq\ & 2 + n + \sum_{\substack{[a],[b] \in G[C_w]/\sim \\ a \not\sim b}} p_a p_b\ - 2n + p \\
\geq\ & 2 + \sum_{\substack{[a] \in G[C_w/\sim \\ a \not\sim a_{max}}} p_a p\ - n + p \\
=\ & 2 + p(n-p) - (n-p) \\
=\ & 2 + (p-1)(n-p) \\
\geq\ & 2 > 0
\end{aligned}$$

For $1 \leq p \leq n$. Since p is defined as the size of the largest clique in $G[C_w]$, this always holds. Thus S' is a better solution than S, which proves Case 2. □

A consequence of Lemma 4.3 is that from now on, as with the Lemmas 3.1 and 3.3, we can assume that an optimal solution will put vertices with distance 2 in the same cluster only if neither is diamond-free, whenever it suits us.

In the next two Corollaries we use this extra structure that Lemma 4.3 gives us.

Corollary 4.4. *Let $u, v \in V(G)$ such that $d_G(u, v) = 2$. If they have only one common neighbor that is not diamond-free, there is an optimal solution that puts u and v in different clusters.*

Proof. Observe that for two vertices $w_1, w_2 \in N_G(u) \cap N_G(v)$, $d_G(w_1, w_2) = 2$. Indeed, since at least one of them is diamond-free they are not adjacent, otherwise they would both be in the diamond induced by $\{u, v, w_1, w_2\}$.
This means that by Lemma 4.3 there exists an optimal solution that does not put w_1 and w_2 in the same cluster. It follows that such a solution will put u and v in the same cluster with at most one of their common neighbors. Then, by Lemma 3.3, there exists an optimal solution that does not put u and v in the same cluster. □

Corollary 4.5. *Let $v \in V(G)$ be a vertex that is part of a diamond in G. Let $u \in V(G)$ such that $d_G(u, v) = 2$. If, in each diamond v is part of, there exists a vertex $v' \neq v$ such that at least one of the following holds:*

(i) $|N_G(u) \cap N_G(v')| \leq 1$,

(ii) *u and v' have at most one common neighbor that is not diamond-free.*

Then there exists an optimal solution to G that does not put u and v in the same cluster.

Proof. Let S be an optimal solution to G that puts u and v in the same cluster. Let W be the set of vertices in the diamonds that v is part of that fulfill either condition (i) or (ii).

If $v' \in W$ fulfills condition (i), then by Lemma 3.1, we can assume that S does not put v' in the same cluster as u. If v' fulfills condition (ii), then by Corollary 4.4 we can assume that S does not put v' in the same cluster as u. Thus $W \not\subseteq C$.

This gives us that for each diamond that v is a part of, at least one vertex is not in C. It follows that in $G[C]$, v is a diamond-free vertex. Therefore, by Lemma 4.3 there exists an optimal solution S_1 to $G[C]$ that does not put u and v in the same cluster.

Let $S_{G[C]} \subseteq S$ be the part of S such that each edge in $S_{G[C]}$ has both endpoints in C.

Then the edge modification set $S' = (S \setminus S_{G[C]}) \cup S_1$ is a solution to G that puts u and v in different clusters. Furthermore, since S_1 is an optimal solution to $G[C]$, $|S'| \leq |S|$. By assumption S is optimal and thus, S' is an optimal solution to G. □

We can see from these Corollaries that Lemma 4.3 is stronger than what it states.

5 Diamond-free Subgraphs

In this section we give a few observations about subgraphs of a graph containing diamond-free vertices that follow from Lemma 4.3.

First, we state the simple consequence that we can cluster a graph optimally such that no vertex in the set of all diamond-free vertices gets clustered with a vertex that is not in the closed neighborhood of that set.

Corollary 5.1. *Let G be a graph, let $W \subset V(G)$ be the set of all diamond-free vertices in G, and let $X = \overline{N_G[W]}$. Then any optimal solution to G will put the vertices in W in different clusters than the vertices in X.*

Proof. Let $u \in X$ and $v \in W$. Since $u \notin N_G[W]$, we can get that $d_G(u, v) \geq 2$. If $d_G(u, v) > 2$, Lemma 2.2 gives us that no optimal solution puts them in the same cluster. If $d_G(u, v) = 2$, note that v is not in any diamond, and thus certainly not in the same diamond as u. Then Lemma 4.3 gives us that no optimal solution will put u and v in the same cluster. □

From this, it follows that, for some subset of the closed neighborhood of this set of diamond-free vertices, we can delete all edges that have exactly one endpoint in this set.

An optimal solution can then be found be solving the two separated graphs independently.

Corollary 5.2. *let $W \subset V(G)$ be the set of all diamond-free vertices in G. Then there exists a subset $F \subseteq N_G[W]$ such that $Opt(G[\overline{F}]) \cup Opt(G[F]) \cup E(F, \overline{F})$ is an optimal solution to G.*

Proof. From Lemma 4.3 we build an optimal solution to G as follows: By Lemma 4.3, there exists an optimal solution S^* to G such that for any $x \in X$, $w \in W$, x and w do not end up in the same cluster.

Since $W \subset \overline{X}$, there exists a subset $F \subseteq \overline{X}$ that does not such that no vertex from F ends up in a cluster with a vertex from X.

If we take F to be maximal in the sense that there is no vertex not in F that ends up in a cluster with some vertex of F, then it follows that the optimal solution S^* needs to cut all the edges between $G[F]$ and $G[\overline{F}]$, so $E(F, \overline{F})$ must be in our optimal solution.

Then S^* will only be optimal if it contains some optimal solution $Opt(G[F])$ to $G[F]$ and some optimal solution $Opt(G[\overline{F}])$ to $G[\overline{F}]$. Notice that $G[F] \triangle Opt(G[F])$ and $G[\overline{F}] \triangle Opt(G[\overline{F}])$ are both cluster graphs, and since $F \cup \overline{F} = V(G)$, $G \triangle S^*$ is a cluster graph.

Thus S^* is a solution to G that makes no sub-optimal edge modifications. That is, S^* is an optimal solution to G. □

The next logical step would be to find this subgraph explicitly.

6 Discussion

This section contains a brief discussion on the open questions that are raised by the results from sections 4 and 5 to which no answer is provided in this paper. These questions can be investigated in further research.

In the Section 5 we established the existence of a subgraph of a graph G induced by a set that is contained in the closed neighborhood of the set W of diamond-free vertices in G (and which itself contains W), that can be solved optimally as part of the optimal solution.

Interestingly, this subgraph is actually diamond-free, and hence, Lemma 2.1 tells us that it can be solved optimally by edge deletions only.

The difficulty now lies in finding actually finding the vertices that belong to the subgraph in polynomial time. Naturally, it contains all diamond-free vertices, but for each of the vertices in $N(W)$ the decision has to be made to include or exclude them from the subgraph.

Note that, although $N[W]$ induces a diamond-free graph, its optimal solution is not necessarily part of any optimal solution of G. To see this, imagine that some vertex w in $N_G(W)$ gets clustered with some vertex u in $\overline{N_G[W]}$ by all optimal solutions of G. It can happen that an optimal solution S of $G[N[W]]$ keeps the edge between w and some vertex in W intact. Then the solution to G that includes S can not be optimal, since it will need to delete the edge u and w or it will have to connect u and v, which is sub optimal by Lemma 4.3. Therefore, we can not assume that all of $N[W]$ is included in the subgraph.

From the proof of Lemma 4.3 we can see that two vertices in $N(W)$ can only be adjacent to the same diamond-free vertex if they are not adjacent to each other. Then, because the subgraph we are looking for is diamond-free and can thus be solved optimally by deleting edges only, we can make the following observation.

Observation 6.1. *For two vertices in $N(W)$ that are adjacent to the same diamond-free vertex, there exists an optimal solution that does not put both of them in the same cluster as the diamond-free vertex they are connected to.*

This gives us that for two such vertices, if neither of them is connected to any other diamond-free vertices, we can assume that at least one of them is not included in the diamond-free subgraph.

In order to decide more explicitly which vertices belong in the subgraph, we suspect that more structure is needed similar to that given by Lemma 4.3. Since diamonds still seem to play

a large role in this decision, it seems like a good idea to look for this structure there.

It would be interesting to see how much more two vertices can be connected by diamonds before no optimal solution puts them in different clusters.

What happens, for instance, with two vertices that are not adjacent and where one of the vertices is not in a diamond with any of the neighbors of the other? Or what if no neighbor of the one vertex is in a diamond with any neighbor of the other vertex? is it possible to find a similar here result as with Lemma 4.3?

If it is, then we there would be a lot more structure to use in finding the diamond-free subgraph.

Regardless of whether these questions hold or not, they seem to be of interest to the question raised in the Introduction:

To what extent do the optimal solutions of Cluster Editing *and* Cluster Deletion *coincide?*

As a separate remark, it would also be interesting to see if the results of Section 4 can be used by themselves to improve existing algorithms for Cluster Editing. In theory they could be used to give a lower bound for size of the optimal solution, but it is likely that a similar lower bound can already be found by existing techniques. Indeed, the neighborhood of a diamond-free vertex forms a union disjoint cliques (Property 4.2), which means that in practice, it could often be groups of densely vertices that are connected sparsely to the rest of the graph. The kernelization algorithm of [10] already handles these types of graphs well.

7 Responsible Research

Since many NP-hard problems lie at the foundation of the worlds most complex challenges, including the protection of digital privacy, protein folding and scheduling, it is important to be aware of the consequences of a major breakthrough in this field.

And although the nature of NP-hard problems makes it unlikely that such a breakthrough will result in rendering the way we currently handle privacy obsolete completely, in practice it could still bring significant advancements in unexpected places.

Therefore, we commit ourselves to be conscientious about our work, even though we submit that in the case of this research it is unlikely that such a breakthrough will occur.

8 Conclusion

In this paper the correspondence between solutions of Cluster Editing and Cluster Deletion has been examined in order to get a better understanding of where they agree.

To that end, it is shown in Section 4 that for vertices that are not part of any diamond, it is possible only look for edge deletions, excluding the possibility of adding an edge. This result has consequences for the way vertices that are clustered together can be connected, as is also shown in Section 4.

These can be used to provide extra structure to clusters of optimal solutions. This structure can then be exploited in order to prove new properties of optimal solutions of Cluster Editing.

Furthermore, the result from Section 4 is used in Section 5 to prove the existence of a subgraph to which an optimal Cluster Deletion solution is part of the optimal solution to the whole graph.

Although the question of whether it is possible to find this subgraph efficiently remains open, a further exploration towards the link between the absence of diamonds and Cluster Deletion seems promising. These open questions are discussed in more detail in Section 6.

Further research into the role that diamonds play in the decision to cluster vertices together or not could be aided by the results from Section 4, and would add to the understanding of the correspondence between Cluster Editing and Cluster Deletion. As discussed in Section 6, this research could also provide more structure similar to that of Section 4, expanding the toolbox for discovering new properties of the Cluster Editing problem.

9 Acknowledgements

I would hereby like to thank my supervisor Dr. Emir Demirović for his valuable feedback, guidance and mentorship for the duration of this research project. Furthermore, I would like to thank my fellow peers Angelos Zoumis, Imko Marijnissen en Lucas Holten for the helpful discussions and feedback.

References

[1] Richard Röttger, Prabhav Kalaghatgi, Peng Sun, Siomar de Castro Soares, Vasco Azevedo, Tobias Wittkop, and Jan Baumbach. Density parameter estimation for finding clusters of homologous proteins—tracing actinobacterial pathogenicity lifestyles. *Bioinformatics*, 29(2), 1 2013.

[2] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering Gene Expression Patterns. *Journal of Computational Biology*, 6(3-4), 10 1999.

[3] Mirko Kfivfinek and Jaroslav Morfivek. NP-Hard Problems in Hierarchical-Tree Clustering. 323:311–323, 1986.

[4] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004.

[5] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Graph-Modeled Data Clustering: Fixed-Parameter Algorithms for Clique Generation. 2003.

[6] Christian Komusiewicz and Johannes Uhlmann. Alternative Parameterizations for Cluster Editing . Technical report, 2011.

[7] René van Bevern, Vincent Froese, and Christian Komusiewicz. Parameterizing Edge Modification Problems Above Lower Bounds. *Theory of Computing Systems*, 62(3):739–770, 4 2018.

[8] Sebastian Böcker. A golden ratio parameterized algorithm for Cluster Editing. *Journal of Discrete Algorithms*, 16:79–89, 2012.

[9] Jie Chen Jianer
}and Meng. A 2k Kernel for the Cluster Editing Problem. In Sartaj Thai My T.
}and Sahni, editor, *Computing and Combinatorics*, pages 459–468, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

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

[11] Sabrine Malek and Wady Naanaa. A new approximate cluster deletion algorithm for diamond-free graphs. *Journal of Combinatorial Optimization*, 39(2):385–411, 2020.

[12] Lucas Bastos, Luiz Satoru Ochi, Fábio Protti, Anand Subramanian, Ivan César Martins, and Rian Gabriel S. Pinheiro. Efficient algorithms for cluster editing. *Journal of Combinatorial Optimization*, 31(1):347–371, 2016.

[13] Christian Komusiewicz and Johannes Uhlmann. Cluster editing with locally bounded modifications. *Discrete Applied Mathematics*, 160(15):2259–2270, 2012.