An overview of different kernelization algorithms for the cluster editing problem

Joost Besseling (4153553)
Bètafaculteit Universiteit Utrecht
Supervisor: Prof. Dr. H. L. Bodlaender

June 16, 2016

Abstract

In the cluster editing problem, we try to transform a given graph G into a disjoint union of cliques using less than k edge modifications. In this paper we will focus on the kernelization of the input (G, k). Kernelization is a pre-processing step in which the size of a given input (G, k) is reduced to a kernel (G', k') with $|G'| < |G|$ and $k' < k$. After the kernelization, the kernel is processed by the normal algorithm, and the optimal solution should be the same as (or easy to find from) the optimal solution of the original problem. In this paper we will give an overview of different kernelization algorithms for the cluster editing problem.

1 Introduction

Cluster editing has plenty of applications, one of the applications is in computational biology. For example, when analysing gene expression data, it is important to identify groups of genes that have similar expression patters. These groups can be identified using cluster editing [6].

We can informally define the cluster editing problem as the question how to transform an input graph to a cluster graph in less than a given amount of edge modifications, an edge modification is either an addition or deletion of an edge in the graph. The cluster editing problem is an NP-hard problem [3]. It is a well studied problem, in this paper an overview of four different kernelizations will be given.

Kernelization is a preprocessing step, that is applied before the main algorithm, which is faster. The goal is to reduce the size of the input problem to a smaller "kernel" of the problem. After that the normal algorithm will be applied to the kernel, instead of to the original input. Solving the problem for the kernel should lead to a solution that is easily transformed into a solution of the input graph. In this paper we will be looking at graphs.

The problem of cluster editing is fixed-parameter tractable, this means that we can guarantee a bound, depending on an input parameter, on the size of the kernel. And we can achieve the kernel in polynomial time.

In this paper an overview of a few kernelization algorithms will be given. Namely, those defined by Guo [4], Chen and Meng [3], and Komusiewicz and Uhlmann[5]. In order to do this, first the necessary rules will be introduced, with a proof of the validity of the rule. After that it will be shown that these rules lead to certain bounds on the kernel size. The structure of some proofs is slightly changed, such that the structure remains the same for all proofs when possible.

In Section 2 all preliminaries necessary for the following sections will be introduced. In Section 3 we will introduce four sets of rules that, when applied, lead to smaller problem kernels. In Section 4 it will be shown that the rules introduced in Section 3 lead to kernels of a specified size, $6k$, $4k$, $2k$ and $4dt$ respectively, where k is the number of edge modifications, d denotes the maximum number of clusters and t denotes the local modification bound. Finally, some concluding remarks will be given.

2 Definitions

The graphs we will be looking at throughout this paper are undirected graphs, there will be no loops in the graphs we consider. There will be at most one edge between two vertices. We will denote the neighbourhood of a vertex v in a graph G as $N_G(v)$ (or $N(v)$ when it is clear which graph is meant), this denotes the set of vertices adjacent to v in G. The closed neighbourhood of v will be denoted by $N_G[v]$, which is $N_G(v) \cup v$. With $N_G^2(v)$ we mean the set of vertices which have a distance of precisely 2 from v. For a set of vertices $V \in G$, we define $N_G(V)$ to mean the set of vertices adjacent to V, but not the vertices contained in V. $N^2(V)$ is defined to mean the neighbours of $N(V)$, without V itself: $N^2(V) = N(V \cup N(V))$. We will be using n to denote the number of vertices in G and m to denote the number of edges.

Now we will give an extensive list of definitions that will be used throughout this paper.

Definition 1. *A critical clique of a graph is a clique K where the vertices of K all have the same closed neighbourhoods and is maximal under this property, that is: all vertices $u, v \in K$ have the property that $N_G[v] = N_G[w]$ and there does not exist a vertex w in $G \setminus K$ such that $N[v] = N[w]$.*

Definition 2. *The editing degree $p_k(v)$ of a vertex $v \in N_G(K)$ with respect to a critical clique K, is defined to be the number of edges that have to be added to make v part of the critical clique K, plus the number of edges that have to be deleted to make v part of the critical clique K.*

Informally, the editing degree $p_K(v)$ can be described as the distance a certain vertex v has to the critical clique K: how many operations need to be performed to make v part of the critical clique K. For example, if we consider figure 1 we can calculate the editing degree of a, b and c as follows: $p_K(a) = 2$, because the only missing edges are $\{a, b\}$ and $\{a, c\}$, and a has no edges with vertices in $N^2(K)$. $p_K(b) = 2 + 1$, because it has one edge with a vertex in $N^2(K)$. Finally, $p_K(c) = 4$.

Definition 3. *If G is a graph, and S is the set of edge modifications. Then S is locally t-bounded, if for every vertex in G, the edge is incident to at most t edge modifications.*

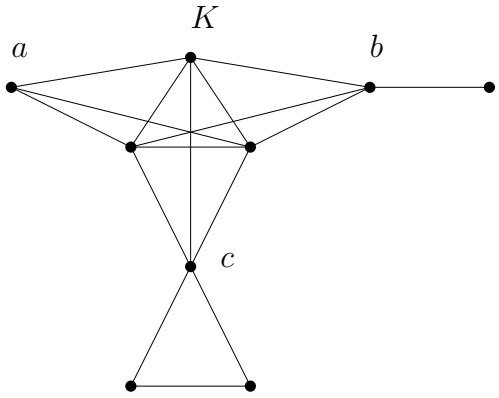

Figure 1

The following lemmas will also be used multiple times.

Lemma 1. *There is no optimal solution that splits a critical clique.*

For a proof of this lemma, see Lemma 1 in [4].

Lemma 2. *Let K be a critical clique, with $|K| \geq |N(K)|$. Then there exists an optimal solution S, such that the clique C containing K is contained in $K \cup N(K)$, which means that $K \subseteq C \subseteq K \cup N(K)$.*

Proof. Since K is a critical clique, we know that K is contained in a cluster C in an optimal solution. Let S' be an optimal solution, such that $C \not\subseteq K \cup N(K)$. This means that there are vertices contained in C that are not in $K \cup N(K)$, that is $D = C \setminus (K \cup N(K))$ and $D \neq \emptyset$. To construct the clique C, the $|K| \cdot |D|$ edges between K and D have to be inserted. But now it is simple to construct a better solution. Let S be another solution, which is the same as S', except for C. Add another cluster C' to the solution, such that $C' = D$, and remove D from C in S . If we compare the two solutions, the second solution uses the same edge modifications, except for adding $|K| \cdot |D|$ edges between K and D, we remove at most $|N(K)| \cdot |D|$ edges between $N(K)$ and D. Since we assumed that $|K| \geq |N(K)|$ this solution is at least as good as S', meaning that S is an optimal solution. □

3 Rules

In this section rules to reduce the input graph will be introduced. We will introduce rules for a number of different algorithms. After proving the rules, it will be proven that these rules lead to a kernel of at most a given size.

3.1 Rules for a kernel size of $6k$

First, the rules leading to a $6k$ kernel. These rules come from Guo [4]. The concept for the proof of Rule 2 also comes from Guo [4].

Rule 1. *Remove all isolated critical cliques K from G.*

Proof. All isolated critical cliques already satisfy the property of being a cluster. We do not need to look at these any more, since we are trying to find an optimal solution in terms of edge modifications. Adding or deleting a vertex to or from a cluster is never optimal. □

Rule 2. *If there exists a critical clique K in G, such that $|K| > |N(K)| + |N^2(K)|$. Then remove $K \cup N(K)$ from G and decrease k by the amount of edge modification necessary to make $K \cup N(K)$ a cluster.*

Proof. For the proof of this rule, we will be comparing two solutions. First, assume, by contradiction, that there does exist an optimal solution $S = \{C_1, \cdots, C_j\}$ that splits $N(K)$. Let K as in the preconditions of this rule. That is $|K| > |N(K)| + |N^2(K)|$. From lemmas 1 and 2, it follows that K is contained in a single cluster C_1 in the optimal solution S, and that $C_1 \subseteq K \cup N(K)$. Now we suppose that $C_1 \subsetneq K \cup N(K)$. It follows from Lemma 1 that there must be a non-empty set of critical cliques A_1 that are not contained in C_1. Let $A_2 = A \setminus A_1$, the vertices from $N(K)$ that are contained in C_1. In order to achieve this solution, the following edge modifications must be performed:

Edge operations splitting $N(K)$.

1. Delete edges between A_2 and A_1.
2. Delete edges between K and A_1.

Deleting the edges between A_1 and K, costs $|A_1| \cdot |K|$. This solution uses at least $|A_1| \cdot |K|$ edge operations.

We will compare this with another solution S' in which $C_1 = K \cup N(K)$. These are the different operations:

Edge operations keeping $N(K)$ together.

1. Add edges in A_1.
2. Add missing edges between A_1 and A_2.
3. Remove edges between A_1 and $N^2(K)$.

The first two operations cost at most $|A_1| \cdot |N(K)|$. The third operation will cost no more than $|A_1| \cdot |N^2(K)|$. Combining this, we find that S' uses at most $|A_1| \cdot (|N(K)| + |N^2(K)|)$ operations that are not used in S. S uses at least $|A_1| \cdot |K|$. But by the conditions of this rule $|K| > |N(K)| + |N^2(K)|$, so $|A_1| \cdot (|N(K)| + |N^2(K)|) < |A_1| \cdot |K|$. This means that S' is an even better solution, contradicting the assumption that S is an optimal solution. We can conclude that there exists an optimal solution such that $K \cup N(K)$. Is contained in a single cluster, and that cluster contains no other vertices. □

Rules for a kernel size of $4k$

In this section we will introduce a new set of rules that, together with Rule 1 will lead to a kernel size of $4k$. Which is $2k$ smaller than in the last section. We will show that it is possible to find stronger bounds for the size of $|V_2|$.

Firstly, a new operator will be introduced: E. Given a critical clique G and $K' \in N(K)$ let $E_{K',N^2(K)}$ denote the set of edges between the vertices in K'

and the edges in $N^2(K)$ and let $E_{K',N(K)}$ denote the set of edges needed to connect the vertices in K' to the vertices in $N(K)$.

The rules in this subsection also come from Guo [4]. The proofs are based on the ideas in [4], but slightly differ in the way the proofs are presented. Guo introduces the critical clique graph, but this is not necessary to proof the correctness of the rules. In fact, it complicates the proofs. It is necessary to obtain a better running time, but the running time is not considered in this paper.

Rule 3. *If there exists a critical clique K in G, such that $|K| \geq |N(K)|$. And if there also exists a critical clique K' in $N(K)$, such that $N(K') \cap N^2(K) \neq \emptyset$ and $|K| \cdot |K'| \geq E_{K',N^2(K)} + E_{K',N(K)}$. Then we remove all edges in $E_{K',N^2(K)}$ and decrease k. If $k < 0$, then there is no solution with less than k edge modifications.*

Proof. For the proof of this rule, we will again compare two solutions. First, assume, by contradiction, that there exists an optimal solution that splits $N(K)$. Let K be a critical clique such that $|K| \geq |N(K)|$, and let K' be a critical clique contained in $N(K)$ such that $N(K') \cap N^2(K) \neq \emptyset$. And $|K| \cdot |K'| \geq |N(K') \cap N^2(K)| + E_{K',N(K)}$, in which $E_{K',N(K)}$ denotes the edges that are missing between K' and $N(K)$. Suppose $S = \{C_1, \cdots, C_j\}$ is an optimal solution that splits K' from K and we have K contained in C_1, and $K' \subset C_2$. And let $S' = \{C_1', C_2', C_3, \cdots, C_j\}$ be a solution that is equal to S, except for clusters C_1 and C_2: $C_1' = C_1 \cup K'$ and $C_2' = C_2 \cap K'$.

Edge operations splitting $N(K)$.

1. Delete edges between K and K'.
2. Delete edges between K' and $N(K) \cap C_1$.

And let $S' = \{C_1', C_2', C_3, \cdots, C_j\}$ be a solution that is equal to S, except for clusters C_1 and C_2: $C_1' = C_1 \cup K'$ and $C_2' = C_2 \cap K'$.

Edge operations keeping $N(K)$ together.

1. Delete existing edges between K' and $N^2(K)$.
2. Add edges between K' and $N(K) \cap C_1$.

Now let us compare these two solutions: S deletes at least $|K| \cdot |K'|$ edges between K and K'. For the solution S', the first operation deletes $|N(K') \cap N^2(K)|$ edges. Operation 2 adds at most all missing edges between K' and $N(K)$. But we assumed that $|K| \cdot |K'| \geq |N(K') \cap N^2(K)| + E_{K',N(K)}$. But this forms a contradiction, since S' is a better solution than the optimal solution. So we can conclude that K' can indeed be added to K.

□

Rule 4. *Let K be a critical clique, as in Rule 3, that is: $|K| \geq |N(K)|$, but this time $N^2(K) = \emptyset$. Then remove $K \cup N(K)$ from G. We decrease k by the number of edges modification needed to make a critical clique from $K \cup N(K)$.*

Proof. This proof will have the same structure as the previous proof.

First, assume by contradiction, that there exists an optimal solution that splits $N(K)$. Let K be a critical clique such that $|K| \geq |N(K)|$ and $N^2(K) = \emptyset$. Suppose that the optimal solution splits $N(K)$. Let $A_1 \subset N(K)$ be a set

containing the critical cliques not in the same clique as K. Let $A_2 = N(K) \setminus A_1$. To get this solution, at least the following edge operations have to be performed:

Edge operations splitting $N(K)$.

1. Delete edges between A_1 and A_2.
2. Delete edges between K and A_1.
3. Insert missing edges in A_2.

Now consider the solution that keeps $N(K)$ in the same cluster.

Edge operations keeping $N(K)$ together.

1. Add edges between A_1 and A_2.
2. Insert edges in A_1.

This amounts to at most $|A_1| \cdot |N(K)|$ edge modifications. But to split $N(K)$ at least $|K| \cdot |A_1|$ edge deletions are necessary. But given the fact that $|K| \geq |N(K)|$, we know that $|K| \cdot |A_1| \geq |N(K)| \cdot |A_1|$. This means that the first solution is not an optimal solution, so we can conclude that $K \cup N(K)$ forms a cluster. □

3.2 Rules for a kernel size of $2k$

In this section a new set of rules will be introduced. These rules are more focused on the concept of editing degree, as introduced in the preliminaries, see definition 2.

All the rules in this subsection come from Chen and Meng [3]; the correctness proofs of the rules are based upon the proofs from Cheng and Meng [3].

Rule 5. *If a critical clique K is larger than k, the editing parameter, then remove all edges between $N(K) \cup K$ and G, insert the edges necessary to make $N(K) \cup K$ a critical clique , decrease k accordingly, and remove $N(K) \cup K$ from G.*

Proof. Since K is larger than the parameter k, it is impossible to add one or more vertices to the critical clique. It is also impossible to remove one of the neighbours from $N_G(K)$, since removing a vertex $v \in N_G(K)$, involves deleting more than k edges. Which would mean that there is no solution using this approach. We can conclude that the only way to proceed is as described in the rule. □

Rule 6. *If a critical clique K has the following properties: $|K| \geq |N(K)|$ and $|K \cup N(K)| > \sum_{v \in N_G(K)} p_K(v)$, then remove all edges between $N(K) \cup K$ and G, insert the edges necessary to make $N(K) \cup K$ a cluster, decrease k accordingly, and remove $N(K) \cup K$ from G.*

Before we prove Rule 6, we will prove the following lemma. This will help in proving the rule.

Lemma 3. *If K is a critical clique with $|K| \geq |N(K)|$ and for all v in $N(K), p_K(v) \leq |K|$, then there exists an optimal solution in which $K \cup N(K)$ is a clique.*

Proof. Suppose that S is a solution graph for G, and K a critical clique in G. We know that there is a partition $C_1, \cdots, C_j$, with $C_i, 1 \leq i \leq j$ clusters of S. Lemma 1 shows that a critical clique in G will never be split apart, so we can assume without loss of generality that $K \subseteq C_1$. Now we define the following sets of vertices: let $N_i = N(K) \cap C_1$, the neighbours of K in one of the C_i, let $R = C_1 \setminus (K \cup N(K))$. Now we can define another solution graph S', where $K \cup N(K)$ is a separate clique, based on the introduced notation as follows: $S' = \{R, K \cup N_1 \cup \cdots \cup N_i, C_2 \setminus N_2, C_3 \setminus N_3, \cdots, C_j \setminus N_j\}$, in figure 2 a schematic overview of these definitions is given. In the following text, these two solutions will be compared and this comparison will lead to a proof of the rule.

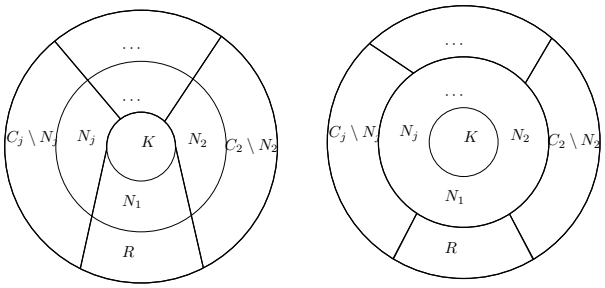

Figure 2

Each of these solution has it own set of edge modifications. Some edge operations like inserting edges inside each N_i will be performed in both solutions. The edge operations that are different are:

Edge operations for solutions S.

1. Insert edges between K and R.
2. Insert missing edges between N_1 and R.
3. Insert edges between $C_j \setminus N_j$ and N_j.
4. Delete edges between K and $N(K) \setminus N_1$.
5. Delete edges between N_i and N_l for $i \neq j$.

Edge operations for solutions S'.

1. Add missing edges between N_i, N_j for $i \neq j$.
2. Remove edges between N_i and $C_j \setminus N_j$
3. Remove edges between N_1 and R.

Firstly, assume that for the critical clique K the following conditions hold: $|K| > |N(K)|$ and for all vertices $v \in N(K)$ the editing degree is at least as small as K. Now we will count the difference in the amount of edge operations between the two solutions.

Operation 1 for the solution S costs exactly $|K| \cdot |R|$, since there are by definition of R no edges between K and R.

Operation 4 costs $|K| \cdot |N(K) \setminus N_1|$ since each vertex in $N(K) \setminus N_1$ had an edge with all vertices in K. We can conclude that S has at least $|K| \cdot |R| + |K| \cdot |N(K) \setminus N_1|$ edge modifications.

Clearly, the editing cost of the first two operations is bound by the sum of the editing degrees of the vertices in the affected sets. So $\sum_{v \in N(K) \setminus N_1} p_K(v)$ gives a bound on the amount of edge modifications. Because of the assumption that $p_K(v) \leq |K|$, $\sum_{v \in N(K) \setminus N_1} p_K(v) \leq |K| \cdot (N(K) \setminus N_1)$

Operation 3 of S' costs $|N_1| \cdot |R|$, but since it is assumed that $|N(K)| \leq |K|$, $|N_1| \cdot |R| \leq |K| \cdot |R|$.

By the combination of these bounds, we find that $S' \not\geq S$, since is S is optimal, so is S'.

□

Proof of Rule 6. Now, by the conditions of Rule 6 ($|K| \geq |N(K)|$ and $|K| + |N(K)| - 1 \geq \sum_{v \in N_G(K)} p_K(v)$), we can deduce that for each vertex u in $N(K)$ the editing degree is bound by:

$$\begin{aligned} p_K(u) &= \sum_{v \in N_G(K)} p_K(v) - \sum_{v \in N_G(K) \setminus \{u\}} p_K(v) \\ (*) &\leq \sum_{v \in N_G(K)} p_K(v) - (|N(K)| - 1) \\ &\leq |K| + |N(K)| - 1 - (|N(K)| - 1) = |K| \end{aligned}$$

For inequality $(*)$ we used that $p_K(v) \geq 1$ for all $v \in N(K)$. So $P_K(v) \leq |K|$ and by definition of the rule we also have $|K| > |N(K)|$. Using Lemma 3 we can conclude that Rule 6 is valid. □

Rule 7. *If there exists a critical clique K with $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$, and if there is no vertex $u \in N^2(K)$ with $|N(u) \cap N(K)| > (|K \cup N(K)|)/2$. Then make $K \cup N(K)$ a disjoint clique and remove $K \cup N(K)$ from G, decrease k accordingly.*

Rule 8. *If there exists a critical clique K with $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$, and if there does exist a vertex $u \in N^2(K)$ with $|N(u) \cap N(K)| > (|K \cup N(K)|)/2$, then make a clique of $K \cup N(K)$, by inserting the missing edges in $N(K)$ and remove the edges between $N(K)$ and $N^2(K) \setminus \{u\}$, decrease k accordingly.*

Before we prove that both Rule 7 and Rule 8 are valid, we will prove some intermediate results.

Lemma 4. *Let K be a critical clique, for which the following conditions of Rule 7 and Rule 8 hold: $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$. Then there is an optimal solution S in which $K \cup N(K)$ is contained in a single cluster.*

The proof of this lemma is found in [3], Lemma 2.6.

Lemma 5. *Let K be a critical clique as in the previous lemma, that is $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$. Then there is exactly one vertex $u \in N^2(K)$, such that $|N(u) \cap N(K)| > (|K| + |N(K)|)/2$. There also exists an optimal solution in which either $K \cup N(K)$ or $K \cup N(K) \cup \{u\}$ is a cluster, in the last case, the vertex u is the vertex such that $|N(u) \cap N(K)| > (|K| + |N(K)|)/2$.*

Proof. Suppose there are two vertices, u_1 and u_2 such that $|N(u_i) \cap N(K)| > (|K| + |N(K)|)/2$ for $i = 1, 2$, then there are at least $2 \cdot (|K| + |N(K)|)/2 = |K| + |N(K)|$ edges between $N(K)$ and $N^2(K)$. But the amount of edges is also bound by $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$ which gives us a contradiction. We can conclude that there is no more than one vertex in $N^2(K)$ for which the aforementioned conditions hold.

Again, we use the same notation as in figure 2. In Lemma 4 it was shown that either S' was optimal and $K \cup N(K)$ was contained in a cluster in the solution, or S was optimal and $K \cup N(K) \cup R$ was contained in a cluster.

as defined in the proof for Lemma 3. In Lemma 4 it was shown in the proof of Lemma 4 that either solution S' is optimal, and $K \cup N(K)$ is contained in the solution, or S is optimal, and $K \cup N(K) \cup R$ is contained in a single cluster in the optimal solution.

If $|R| = 1$ then it is clear that $K \cup N(K)$ is contained in a cluster in an optimal solution. Assume $|R| \geq 1$. We know that the operation on S' costs at most $\sum_{v \in N_G(K)} p_K(v) \leq |K| + |N(K)| - 1$ edge modifications. In order to make the solution S viable, we need to add the edges between K and R, which costs $|K| \cdot |R|$ since there are no connections beforehand. The number of edges between $N(K)$ and R is bound by $\sum_{v \in N_G(K)} p_K(v) \leq |K| + |N(K)| - 1$, so adding all edges between $N(K)$ and R costs at least $|N(K)| \cdot |R| - |K| + |N(K)| - 1$ edge modifications. Combining these numbers and the fact that S should be optimal, we find that:

$$
\begin{aligned}
|K| + |N(K)| - 1 &\geq |R| \cdot (|K| + |N(K)|) - (|K| + |N(K)| - 1) \\
2 \cdot (|K| + |N(K)| - 1) &\geq |R| \cdot (|K| + |N(K)|)
\end{aligned}
$$

which is not possible when $|R| > 2$, so $|R| = 1$. So R contains a single vertex u. We already know that there was a single cluster C_1 that consisted of $K \cup N(K) \cup R$, so now we can replace R with a single vertex u. Now consider this vertex u, suppose that there are less vertices in $K \cup N(K)$ adjacent to u than vertices that are adjacent to u. In this case, a solution that removed u from the cluster would be at least as good as a solution that would, so there exists an optimal solution with $K \cup N(K)$ as a cluster. So we can assume that more than half of the vertices in $K \cup N(K)$ is adjacent to u, or $N(u) \cap N(K) > (|K| + N(K)|)/2$. This last property also implies that u is indeed an element of $N^2(K)$. □

With these two lemma's we are ready to prove that both Rule 7 and Rule 8 are valid.

Proof. Both rules have some conditions in common: let K be a critical clique with $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$. By Lemma 4 there is an optimal solution in which $K \cup N(K)$ is entirely contained in a single cluster C_1. But with Lemma 5 we also now that the remainder of the vertices in C_1 is either a single vertex u, or nothing. In the latter case Rule 7 holds. Since we know that if Rule 7 holds, there is no vertex u such that $|N(u) \cap N(K)| > (|K| + |N(K)|)/2$ holds. In lemma 5 we saw that the only vertex that could

possibly be in the cluster C_1 was exactly this vertex. In other words, Rule 7 holds.

Now we can assume that there does exist such a vertex u. By Lemma 4 there only exists one vertex with these properties and there exists a cluster that consists only of $K \cup N(K)$ or of $K \cup N(K) \cup \{u\}$. So we can remove the edges between the vertices in $N^2(K) \setminus \{u\}$ and $N(K)$. We can conclude that Rule 8 also holds. □

In order to introduce the next rule, we have to introduce the Pendulum Algorithm, as defined in [3], first:

Algorithm 1: The Pendulum Algorithm

Input : (G, k) and K: (G, k) is the instance of the cluster editing problem. K is a critical clique, with $N(K)$ a single critical clique, and $N^2(K)$ is a single vertex u

Output: (G', k'): a reduced instance of equivalent to (G, k)

1 **if** $|K| \geq |N(K)|$ **then**
2 | make $K \cup N(K)$ a disjoint clique
3 | **return** $(G \setminus (K \cup N(K)), k - |N(K)|)$
4 **if** $|K| < |N(K)|$ **then**
5 | Let $U \subset N(K)$ be $|K|$ random vertices from $N(K)$
6 | **return** $(G \setminus (K \cup U), k - |K|)$

Rule 9. *If for a critical clique K such that $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$, and rules 5 to 8 can not further reduce G, apply the Pendulum Algorithm to K.*

Lemma 6. *Let (G, k) be an instance of the cluster editing problem for which none of the rules 5 to 8 can reduce the instance. If K is a critical clique such that $|K| < |N(K)|$ and $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$. Then $N^2(K)$ consists of a single vertex u and $N(K)$ is a critical clique.*

Proof. Because Rule 7 cannot reduce the instance, there must be a vertex u in $N^2(K)$ such that $|N(u) \cap N(K)| > (|K \cup N(K)|)/2$, otherwise Rule 7 could still be applied. From Lemma 5 it follows that there are no other vertices with this property. If there are other vertices in $N^2(K)$, Rule 8 would still reduce the instance by deleting edges between $N(K)$ and $N^2(K) \setminus \{u\}$. $N(K)$ is a clique in G, since Rule 8 cannot reduce the instance, and this rule would add the edges missing from $N(K)$ in order to make it a clique. The only vertex in $N^2(K)$ is u. Since K is a maximal clique, and all vertices in $N(K)$ are adjacent to K, there can not be a vertex in $N(K)$ that is not adjacent to u, since that vertex would form a critical clique with K. Which contradicts the maximality of K. So $N(K)$ is also a critical clique. □

Lemma 7. *The Pendulum algorithm is correct*

Proof. First we will look at the first part of the algorithm (lines 1 through 3). By the precondition of this rule, we have $|K| \geq |N(K)|$. Since there is only a single vertex in $N^2(K)$ and since $N(K)$ is already a clique, the editing degree of a vertex in $N(K)$ is exactly 1, $p_K(v) = 1 \leq |K|$. By Lemma 3 there is

an optimal solution that has $K \cup N(K)$ as a cluster. It is also clear that the second step is correct. All that an optimal solution has to do, is delete all edges between $N(K)$ and u, thus we decrease k with $|N(K)|$.

Proving that the next couple of lines are also correct is more difficult. This will happen in two steps. Let (G, k) be the input instance, and let (G', k') be the instance that results from applying the second part of the algorithm to it. We will show that an optimal solution for G contains no more than k edge operation if and only if an optimal solution for G' contains no more than $k' = k - |K|$ edge operations.

First we will prove that an optimal solution for G' will use at most $k' = k - |K|$ edge modifications. Note that for a vertex v in $N(K)$ $p_K(v) = 1$, so clearly $\sum_{v \in N_G(K)} p_K(v) = |N(K)| < |K| + |N(K)|$. With the condition of this rule, we also have that $|K| < |N(K)|$. With Lemma 5 we find that there is an optimal solution S for G that has either $K \cup N(K)$ or $K \cup N(K) \cup u$ as a cluster C_1. We assume that S uses no more than $k_1 \leq k$ edge modifications.

Suppose U is the set of $|K|$ vertices that were removed from G, there is a different optimal solution for $G' = G \setminus U$. Now let S' be a solution for G' that has all the same clusters as S, except for the cluster C_1 that contains U in S. Instead S' has the cluster $C_1' = C_1 \setminus U$. Now the only difference in edge operations between these two solutions comes from C_1. If C_1 contains u, than C_1 needs to insert the $|K|$ edges between K and u. This is not necessary in C_1'. If $C_1 = K \cup N(K)$, than S deletes the edges between U and u, since $|U| = |K|$, $|K|$ edge deletions are necessary. We can conclude that solution S' needs no more than $k_1' = k_1 - |K| \leq k - |K|$ edge operations. So an optimal solution for G' contains no more than $k' = k - |K|$ edge operations.

Second, we will prove that an optimal solution for G will use at most k edge operations, if an optimal solution for G' uses no more than $k' = k - |K|$ operations. Let $S_2' = \{C_1', C_2', \cdots, C_j'\}$. $N(K) \setminus U$ is a critical clique in G', from Lemma 1 it follows that $N(K) \setminus U$ is contained in a single cluster C_1'.

If C_1' contains a vertex v, such that $v \neq u$ and $v \notin N(K) \setminus U$, then there is another optimal solution obtained from S_2', $S_3' = \{N(K) \setminus U, C_1' \setminus (N(K) \setminus U), C_2', \cdots, C_j'\}$. First, we will prove that this solution is also optimal. Suppose that S_2' uses $k_2' \leq k' = k - |K|$ edge operations. We can obtain solution S_2' by not inserting the edges between $N(K) \setminus U$ and $C_1' \setminus (N(K) \setminus U)$, which saves at least $|N(K) \setminus U|$ edge operations, because $C_1' \setminus (N(K) \setminus U)$ contains at least the vertex v. If C_1' does contain the vertex u, then the edges between $N(K) \setminus U$ and u have to be deleted, which costs an additional $|N(K) \setminus U|$ edge operations. Combining these operations, we can conclude that S_2' uses at most k_2' edge operations, thus proving its optimality.

From the previous, we know that there always is an optimal solution $S_4' = \{C_1'', \cdots, C_j''\}$ that has either $N(K) \setminus U$ or $N(K) \setminus U \cup \{u\}$ as a cluster C_1'. Because either in the optimal solution S_2' C_1' does not contain a vertex v, which means that C_1' is equal to $N(K) \setminus U$ or $N(K) \setminus U \cup \{u\}$. Or there is another optimal solution S_3' that has $N(K) \setminus U$ as a cluster.

Now consider the solution $S_4 = \{C_1'' \cup U \cup K, C_2'', \cdots, C_j''\}$. This solution can be obtained from S_4'. If $C_1'' = N(K) \setminus U$, then S_4 has to delete the $|U| = |K|$ edges between U and u. Otherwise, if $C_1'' = N(K) \setminus U \cup \{u\}$, then S_4 inserts the K edges between K and u. We can conclude that this solution costs at most $k_2' + |K| \leq k' + |K| = k$ edge operations, which concludes the proof.

Combining this, we can conclude that we have shown that: G contains no more than k edge operation if and only if an optimal solution for G' contains no more than $k' = k - |K|$ edge operations. This means that the pendulum algorithm is indeed correct.

□

Proof of Rule 9. We can now combine these two lemmas, and prove Rule 9. By the conditions of the rule, $|K|$ is a critical clique such that $|K| < |N(K)|$, $|K| + |N(K)| > \sum_{v \in N_G(K)} p_K(v)$ and (G, k) can not be reduced. Using Lemma 6, we find that $N(K)$ forms a critical clique and that $N^2(K)$ consists of a single vertex u. This means that the conditions of the pendulum algorithm are met. From Lemma 6 it follows that we can apply the pendulum algorithm safely. We can conclude that Rule 9 is correct. □

3.3 Rules for a kernel size of $4dt$

Because the parameter k is not necessarily small, it is interesting to look for other parameters. Komusiewicz and Uhlman [5] came up with another approach. They look at the local modification bound. Unfortunately, this did not lead to useful results. However, when combining the local modification bound with a maximal number of clusters d. They named this (d, t)-constrained cluster editing. I will show their reduction, that leads to a problem kernel containing at most $4dt$ vertices. The rules in this subsection were first defined in [5].

As before, we will introduce the necessary reduction rules first.

Rule 10. *If there are two vertices u and v, which are adjacent to each other and $N(u)$ contains more than $2t$ vertices that are not in $N[v]$, then remove the edge between these vertices. Set $\tau(v) = \tau(v) - 1, \tau(u) = \tau(u) - 1$.*

Here, τ is defined as the function that keeps track of the local number of edge modifications. Initially, $\tau(v)$ is set to t for every $v \in G$. Every time an edge modification is performed with adjacent vertices u_1 and u_2, both $\tau(u_1)$ and $\tau(u_2)$ are reduced with 1.

Proof. Let u and v be such vertices. Suppose that G' is a solution of the reduced instance in which u and v are not separated. Let K be a cluster in G' that contains both vertices. From the fact that at most t edges are inserted that are incident with v it follows that $|K \cap N(u) \setminus N[v]| \leq t$. But this means that more than t edges incident to u have to be deleted. □

And a similar second rule:

Rule 11. *If there are two vertices u and v, which are not adjacent to each other, but they have more than $2t$ neighbours in common, then add the edge between these vertices. Set $\tau(v) = \tau(v) - 1, \tau(u) = \tau(u) - 1$.*

Proof. Let u and v be such vertices. Suppose that G' is a solution of the reduced instance in which u and v are not in the same cluster. Let K be a cluster in G' that contains u. Since u is incident to at most t edge deletions, and u and v have more than $2t$ neighbours in common, $|N_G(u) \cap N_G(v) \cap K| > t$. But this is impossible, since v is also incident to at most t edge modifications, but this implies that more than t edges are deleted. We can conclude that there is no solution that separates u and v. □

Rule 12. *If a vertex v is encountered for which $\tau(v) < 0$, then there is no solution to the problem.*

Rule 13. *If there is an isolated clique K, with $K > 2t$, then remove K from G and decrease d with* 1.

Proof. Since K is an isolated clique with more than $2t$ vertices, there is at least a vertex $u \in K$, that is still adjacent to at least t vertices originating from K in every solution G' that is the result of (d, t)-Constrained cluster editing.

So there exists a cluster K', which contains u and thus also contains $t + 1$ other vertices from K. Every vertex in K is adjacent to at least $t + 1$ vertices in K', which means that all these vertices are contained in K' and we find that $K \subseteq K'$. But it is impossible that K' contains vertices that are not in K, because that would require more than $t + 1$ edge insertions. We can conclude that K stays an isolated clique in the solution. □

3.4 Similarities

Knowing all the rules, it becomes clear that these rules are quite similar to each other. If we compare the rules leading to $2k, 4k$ or $6k$ kernels, it becomes clear that the rules differ only in small aspects. For example, all rules reason about critical cliques. This is of course logical, since these are the smallest sets of vertices which we can separate, we can not separate two vertices from within the same critical clique. Another example is the fact that all methods optimizing on edge modifications, have a rule that removes an already existing cluster. Both $4k$ and $6k$ kernels have a rule that states this. The $2k$ kernel has Rule 6, which does remove clusters. Even the $4dt$ kernel has a similar rule, Rule 13.

4 Using the rules

In this section we will use the rules defined in the previous section to show that these rules lead to different kernelizations. It will also be shown what the size of the resulting kernels is. The proofs of the following Theorems are based upon the proofs from respectively Guo [4], Chen and Meng [3] and Komusiewicz and Uhlmann [5].

4.1 A $6k$ kernelization algorithm

Algorithm 2: Reduction leading to a kernel size of at most $6k$

Input : (G, k) an instance of the cluster editing problem: G is a graph and k denotes the maximum number of edge modifications.

Output: A kernelization of the given input, with a size of at most $6k$ or "no" if the given instance has no solution.

```
while Rule 1 or Rule 2 can reduce the instance do
    Reduce the instance with Rule 1 and Rule 2
return the reduced instance or "no" if one of the rules lead to an invalid instance.
```

Theorem 1. *If an instance (G, k) of the cluster editing problem can not be reduced by any of the rules 1 or 2, then (G, k) has either less than $6k$ vertices, or it has no solution.*

Before proving Theorem 1, we will prove an intermediate result. Which will also be used later in this section.

Lemma 8. *If a cluster C in an optimal solution S of G has unaffected vertices, then these vertices form a single critical clique in G.*

Proof. Consider a cluster C in the solution graph, that has unaffected vertices. Let K be the set of unaffected vertices. Since all vertices in K are unaffected, we now that $N_G(K) = N_S(K)$. Since the vertices in K are all part of the same cluster, they must have the same closed neighbourhood. Now we want to prove that they also form a critical clique in G, to do this we have to prove the maximality of K. But we know with Lemma 1 that a solution will never split a critical clique, so two vertices with the same neighbourhood will never be split. Which means that K is a critical clique. □

Proof of Theorem 1. Suppose that G has a solution set with at most k edge modifications. Now we look at the cluster graph S that is de result of the edge modifications. We can partition S into two sets of vertices: V_1 the set of affected vertices and V_2 the set of unaffected vertices. We can clearly see that the upper bound on V_1 is $2k$, which happens if there are exactly k edge modifications that all touch two unique vertices.

Now we will give an upper bound on V_2. Consider a cluster in S. This cluster might have a set of unaffected vertices K. By Lemma 8 K is a critical clique. Because Rule 1 can not be applied $N(K)$ is not empty. Now consider $N(K)$. Since Rule 2 can not be applied,it follows that $K \leq |N(K)| + |N^2(K)|$, because all instances where the reverse $K > |N(K)| + |N^2(K)|$ is true, would be removed by Rule 2 . This gives us an upper bound on K. Now we can calculate the size of V_2 by iterating over all kernels of untouched vertices. Suppose there are l cliques in the output graph: $C_1, \cdots C_l$, let K_i be the untouched kernel in each clique. Then

$$|V_2| = \sum_{i=0}^{l} K_i \leq \sum_{i=0}^{l} (|N(K_i)| + |N^2(K_i)|).$$

If two clusters have at least one edge between them in G, then some affected vertices could be counted twice in $(|N(K)| + |N^2(K)|)$. Once in $N(K)$ and once in $N^2(K)$). But if one vertex is counted more than twice, it has to be touched by more than 1 modified edge. But if the same vertex is touched twice by a modified edge, we find a new bound on the size of V_1, $V_1 = 2k - 1$. In the worst case, every vertex is counted twice. We know that $|V_1| \leq 2k$, so

$$V_2 \leq 2 \cdot V_1 \leq 4k.$$

This gives a bound for the total size of the kernel of

$$|G| = V_1 + V_2 \leq 2k + 4k = 6k.$$

So if G has a solution, it has at most $6k$ vertices. □

Corollary 1. *Algorithm 2 is correct, and produces a $6k$ kernel.*

Proof. Applying the algorithm to an instance produces an equivalent, smaller instance, since we have proven that all rules are correct. With Theorem 1, we have proven that if there exists a solution, it has a size of $6k$. □

4.2 A $4k$ kernelization algorithm

Using the two new rules 3 and 4, we can achieve an even smaller kernel of size $4k$, for the cluster editing problem. First, we give the algorithm:

Algorithm 3: Reduction leading to a kernel size of at most $4k$

Input : (G, k) an instance of the cluster editing problem: G is a graph and k denotes the maximum number of edge modifications.
Output: A kernelization of the given input, with a size of at most $4k$ or "no" if the given instance has no solution.

```
while Rule 1, Rule 3 or Rule 4 can reduce the instance do
    Reduce the instance with one of the rules.
return the reduced instance or "no" if one of the rules lead to an invalid instance.
```

Theorem 2. *If a graph G can not be reduced by any of the rules 1, 3 or 4, and it has more than $4k$ vertices. Then there is no solution for cluster editing with at most k edge modifications.*

Proof. First, suppose that there exists a solution of the reduced instance with at most k edge modifications, we denote it by S. From this we obtain a cluster graph with j cliques, $S = \{C_1, C_2, \ldots, C_j\}$. Now we partition the set of vertices of the solution into two parts: V_1 and V_2, where V_1 holds the affected vertices, and V_2 the unaffected vertices. Clearly, $V_1 \leq 2k$.

As we have proven in Lemma 8 all the unaffected vertices form at most j critical cliques, such that $K_i \subset C_i$. Let $\mathcal{K} = \{K_1, \cdots, K_j\}$. We now partition $\mathcal{K}$ into two sets of critical cliques: $\mathcal{K}_1$ and $\mathcal{K}_2$, where $\mathcal{K}_1$ holds the cliques K, such that $K < N(K)$. And $\mathcal{K}_2 = \mathcal{K} \setminus \mathcal{K}_1$ holds the remaining cliques.

First, we will consider $\mathcal{K}_1$. Because Rule 1 can not be applied, we know that $N(K) \neq \emptyset$, and all vertices in $N(K)$ are affected. From the fact that all vertices in $N(K)$ are affected, it follows that $|N(K_i)| \leq 2 \cdot |E_i^+| + |E_i^-|$, where E_i^+ is the set of edges that are inserted by the solution with both ends in C_i and E_i^- is the set of affected edges with at least one end in C_i, these edges are deleted.

Now for $\mathcal{K}_2$. Because rules 1 and 4 can not be applied, we know that $|N(K)| \neq \emptyset$. Since Rule 3 can not reduce G, we know that there exists a critical clique K' in $N(K)$, such that $N(K') \cap N^2(K_i) \neq \emptyset$ and $|K_i| \cdot |K'| < |E_{K',N(K_i)}| + |N(K') \cap N^2(K_i)|$. We also know that all vertices in K_i are unaffected, so $N^2(K)$ is not part of cluster C_i and all vertices in $N(K_i)$ are. From this we get

$$|K_i| < (|E_{K',N(K_i)}| + |N(K') \cap N^2(K_i)|)/|K'| \leq |E_i^+| + |E_i^-| \leq 2 \cdot |E_i^+| + |E_i^-|$$

Now we can give an upper bound on V_2, we define E^+ to be the set of edges

inserted by the solution, and E^- to be the set of edges that are deleted:

$$|V_2| = \sum_{i=1}^{l} K_i \leq \sum_{i=1}^{l} 2 \cdot |E_i^+| + |E_i^-| = 2 \cdot |E^+| + \sum_{i=1}^{l} |E_i^-|$$

In $\sum_{i=1}^{j} E_i^-$ all edges are counted twice, since every edge that is deleted, also has an end in another cluster. In that cluster we count the deletion too. So $\sum_{i=1}^{l} |E_i^-| = 2 \cdot E^-$. If we use this, we get

$$|V_2| \leq 2 \cdot |E^+| + \sum_{i=1}^{l} |E_i^-| = 2 \cdot |E^+| + 2 \cdot |E^-| = 2k$$

Combining this bound with the bound $V_1 \leq 2k$, we can conclude that the size of the kernel is at most $|V_1| + |V_2| \leq 2k + 2k = 4k$. So if there does exist a solution, the kernel is smaller than $4k$. □

Corollary 2. *Algorithm 3 is correct, and produces a $4k$ kernel.*

Proof. Applying the algorithm to an instance produces an equivalent, smaller instance, since we have proven that all rules are correct. With Theorem 2, we have proven that if there exists a solution, it has a size of $4k$. □

4.3 A $2k$ kernelization algorithm

Algorithm 4: Reduction leading to a kernel size of $2k$

Input : (G, k) an instance of the cluster editing problem: G is a graph and k denotes the maximum number of editing steps.
Output: A kernel of the given input of size $2k$ or "no", when the given instance has no solution.

```
while the rules 5 to 9 can reduce the instance do
    Reduce the instance with the rules 5 to 9
return the reduced instance or "no" if one of the rules returned "no"
```

Theorem 3. *If (G, k) is an instance of the cluster editing problem which cannot be reduced by any of the rules 5 to 9, and if $|G| > 2k$. Then it has no solution.*

Proof. Let S be an optimal solution of (G, k). A vertex u in G is touched if it is incident to an edge modification (either an insertion or a deletion) of S, u is untouched otherwise. We define the operation $p_0(u)$ to be the number of edge modification resulting from S, incident to u. We can now partition S into two sets: S_1, the clusters in which all vertices are touched and S_2, the remaining clusters. Since all vertices in a cluster in S_1 are touched, so $p_0(v) \geq 1$ for all these vertices, we have

$$\sum_{C \in S_1} |C| \leq \sum_{C \in S_1} \sum_{v \in C} p_0(v) \tag{1}$$

It is clear that some of the vertices in S_2 are untouched, and that these vertices must form a critical clique, as we have seen before in Lemma 8. So

for a cluster C in S_2, we know that there must be a set of untouched vertices K, forming a critical clique in C. So we know that $C = K \cup N(K)$. Since we also know that none of the rules can be applied, we find that $|K| + |N(K)| \leq \sum_{v \in N(K)} p_K(v)$. In this case it also holds that $p_K(v) = p_0(v)$ for $v \in N(K)$ by definition of $p_K(v)$. If we combine this with the fact that all vertices in K are untouched, we find

$$|C| = |K| + |N(K)| \leq \sum_{v \in N(K)} p_K(v) = \sum_{v \in N(K)} p_0(v) = \sum_{v \in C} p_0(v).$$

So if we now sum over all clusters in S_2, we find that

$$\sum_{C \in S_2} |C| \leq \sum_{C \in S_2} \sum_{v \in C} p_0(v) \tag{2}$$

so,

$$\begin{aligned} |G| &= \sum_{C \in S_1} + \sum_{C \in S_2} \\ &\leq \sum_{C \in S_1} \sum_{v \in C} p_0(v) + \sum_{C \in S_2} \sum_{v \in C} p_0(v) \\ &= \sum_{C \in S} \sum_{v \in C} p_0(v) \end{aligned}$$

Since $p_0(v)$ counts how many vertices are affected, and how many times each vertex is affected.

We can conclude that there are at most $2k$ vertices in G. □

Corollary 3. *Algorithm 4 is correct, and produces a $2k$ kernel.*

Proof. Applying the algorithm to an instance produces an equivalent, smaller instance, since we have proven that all rules are correct. With Theorem 3, we have proven that if there exists a solution, it has a size of $2k$. □

4.4 A $4dt$ kernelization algorithm

Algorithm 5: Reduction leading to a kernel size of $4dt$

Input : (G, d, t, k) an instance of the (d, t)-constrained cluster editing problem: G is a graph, d denotes the maximum number of cliques, t denotes the local modification bound and k denotes the maximum number of editing steps

Output: A kernel of the given input of size $4dt$ or "no"

```
while The rules can reduce the instance do
    Reduce the instance with the rules 10 to 13
return the reduced instance or "no" if one of the rules returned "no"
```

Theorem 4. *If (G, d, t, k) is an instance of the (d, t)-constrained cluster editing problem, which can not be reduced by any of the rules 10 to 13 and which is bigger than $4dt$. Then it has no solution.*

Proof. Since reducing an instance with the given rules, leads to an equivalent instance. We know that the result of applying algorithm 5 is equivalent to the input instance.

We will show by contradiction that all cliques are smaller than $4t$. Assume that this is not the case, and let K be a cluster with $|K| > 4t$. Then, by Rule 13 this clique can not be an isolated clique in G', so there is either an edge missing between to vertices in K, or there exists at least one edge between a vertex in K and a vertex in $G' \setminus K$.

Let u and v be vertices in K, with no edge between u and v. Then both u and v have at most $t-1$ edges missing between themselves and the vertices of K. If more edges were missing, it would be impossible to add u or v to K. But that would mean that both u and v are adjacent to $|K|-(t-1)$ vertices in $K \setminus \{u, v\}$. Since $K > 4t$, this would mean that they have $2t+1$ common neighbours. But that would mean that the conditions of Rule 11 apply: a contradiction. We can conclude that there are no vertices as u and v in K.

Let u in K and v in $G' \setminus K$, such that u and v are adjacent to each other. Since v is not contained in K, v was adjacent to at most $t-1$ vertices in K. Since u is contained in K, u was adjacent to at least $|K|-(t-1)$ vertices in K. So u was adjacent to at least $2t+1$ vertices, that were not adjacent to v. But that would mean that Rule 10 would apply, which leads again to a contradiction, since the rules could no longer affect G'.

We can conclude that there are no cliques in G'' that are bigger than $4t$. Together with the maximum number of clusters d, this leads to a bound of $4t \cdot d$. □

Corollary 4. *Algorithm 5 is correct, and produces a $4dt$ kernel.*

Proof. Applying the algorithm to an instance produces an equivalent, smaller instance, since we have proven that all rules are correct. With Theorem 4, we have proven that if there exists a solution, it has a size of $4dt$. □

5 Conclusions

We have seen four different kernelizations for the cluster editing problem. Three of them were relatively the same, and the fourth kernelization took a different approach. Guo [4] first introduced critical cliques to get a good upper bound on the size. Chen and Meng [3] introduced introduced the editing degree of a vertex, which led to an even smaller kernel. Komusiewicz and Uhlmann [5] used a different approach by looking at two different parameters, the total number of clusters, and the local modification bound. This approach gave a different bound on the kernel.

In this paper only a small overview of one aspect of Cluster Editing is given. Only kernels were considered. Of course, this problem has a broader scope than only its kernels. Other aspect of the problem are algorithms for solving the cluster editing itself. There are many papers that use an experimental approach to the problem, such as [1].

It is also possible to look beyond the scope of just Cluster Editing. There are a lot of small variations, such as Cluster Deletion and Weighted Cluster Editing. Weighted Cluster Editing is a more general version of Cluster Editing. Since

setting all weights to 1, leads to a normal version of Cluster Editing. Weighted Cluster Editing is studied in [2].

There is even more if one looks beyond the scope of Cluster Editing. But we will not go into that here.

References

[1] Sebastian Böcker, Sebastian Briesemeister, and Gunnar W. Klau. Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, 60(2):316–334, 2009.

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

[3] Jianer Chen and Jie Meng. A 2k kernel for the cluster editing problem. In *Lecture Notes in Computer Science*, pages 459–468. Springer Science + Business Media, 2010.

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

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

[6] R. Sharan, A. Maron-Katz, and R. Shamir. CLICK and EXPANDER: a system for clustering and visualizing gene expression data. *Bioinformatics*, 19(14):1787–1799, sep 2003.