University of Bergen
Department of informatics

Parameterized Complexity of Fair Graph Clustering

Author: Joakim Sunde
Supervisors: Petr Golovach

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

August, 2022

Abstract

The problem of $\alpha-$ *BALANCED CLUSTER VERTEX DELETION* where $\alpha \geq 1$ is some constant, asks whether it is possible to delete at most k vertecies from a vertex colored graph such that that it becomes a cluster graph, where for each cluster, the ratios between the number of vertecies with each color is bounded by the constant α. We study the problem from a Parameterized Complexity point of view with the parameters k and l, where k is the solution size and l is the number of colors. We present kernels with $\mathcal{O}(k^2 l)$ and $\mathcal{O}(k^4)$ vertecies and an improved $\mathcal{O}(k^3)$ kernel in the special case when $\alpha = 1$. We also provide an $\mathcal{O}(3^k(|V| + |E|))$ FTP algorithm.

Acknowledgements

I wanna thank my supervisor Petr Golovach for his guidance, patience and for bringing some much need structure in writing this thesis. I would also like to thank my friends at the 3th floor for a lot of fun and table tennis.

Joakim Sunde
29 August, 2022

Contents

Chapter 1

Introduction

1.1 Motivation

In this thesis, we will study the problem of clustering which we can describe informally as grouping similar things together. There are many different types of clustering but here we will study clustering on graphs and our clusters will be cliques in these graph which means that in a cluster all the nodes are connected to all other nodes. One natural thing to model using graphs are social networks where the edges represent a friendship between two people. In this case a cluster would be a group of friends where everybody is friends with everybody. A natural question is whether we can modify the social network in some way (usually as little as possible) so that the remaining graph is a cluster graph. There are multiple possible ways to define such transformation. One way to to do is if we are allowed to delete edges. This problem is called *CLUSTER EDGE DELETION*. Another way to define this transformation is where we are allowed to delete the nodes. This version of the problem is called *CLUSTER VERTEX DELETION* and is a well studied problem.

Now we could require our resulting cluster graph to satisfy certain properties. For example, if we view our graph as a social network, we might want our resulting clusters to have some sort of gender balance. A natural way to model this is to give the nodes of the graph colors and define some constant $\alpha \geq 1$ which describes how gender balanced our cliques has to be. For example, $\alpha = 2$ would mean that there can be at most two times the number of people with one gender compared to the other gender in all the resulting cliques. This is the problem we will study here, but we can have any number of colors and the ratios of the number of vertecies between each pair of colors have to be within some constant α for each clique.

Another potential application similar to Huffner et al. [6] is if we have some DNA samples some of which are from the same species and we have a method to check the equivalence of two samples. A graph is then made by creating a vertex for each sample and an edge whenever 2 samples are equivalent. If there are no errors then we have a cluster graph, but some samples may be contaminated so that they produce inconsistent results so we want to remove these to obtain a cluster graph. Now if we know that for each species we have approximately the same number of samples from each gender (and a way to test if two samples have the same gender). Then we can color the vertecies and potentially obtain a better clustering by picking some appropriate α.

1.2 Related works

Many variants of clustering have been studied. For *CLUSTER VERTEX DELETION* the first study from a Parameterized Complexity point of view was done by Huffner et al. [6] which gave a Fixed-Parameter Algorithm running in time $\mathcal{O}(2^k k^9 + n^3)$, and the fastest known algorithm is given by Tsur [11] which runs in time $\mathcal{O}^*(1.811^k)$. For kernelizations of *CLUSTER VERTEX DELETION*, the best known kernel is given by Le et al. [8] which has $\mathcal{O}(k^{\frac{5}{3}})$ vertecies. Several different versions of this problem have been studied. One is where there is a bound on the number of cliques d. Huffner et al. [6] give a $\mathcal{O}(k^3)$ vertex kernel for the weighted version of this problem. There are also variants where the cliques have to be balanced in terms of size. Steinvik [10] gives a kernel of $\mathcal{O}(k^4)$ vertecies for two different variants of this problem.

Clustering have also been heavily studied in the context of unsupervised machine learning. Here the points are in some metric space and the goal is to create clusters minimizing some objective function. There is a sub area of this kind of clustering called fair clustering where the clusters have to satisfy some additional "fairness" requirements. For instance there is the requirement of balanced representation studied by Chierichetti et al. [2]. Here each point has a color and the ratio of points from each color has to be within some constant. This have been generalized by Bera et al. [1] where each point is allowed to have multiple colors.

1.3 Our Results

A cluster graph is a graph where all the components are cliques. In the *CLUSTER VERTEX DELETION* (CVD) problem, we are given a graph and integer k and want to find out if it's possible to remove at most k vertecies such that the resulting graph is a cluster graph. We will study a variant of this problem for colored graphs called $\alpha-$*BALANCED CLUSTER VERTEX DELETION*.

Let $\alpha \geq 1$ be some rational constant. We denote a pair (G, c) where G is a graph and $c : V(G) \to \{1, ..., l\}$ is a function assigning each vertex a color, as a colored graph. A colored graph (G, c) is an $\alpha-$balanced cluster graph if

- G is a cluster graph.
- For every connected component C of G, $\frac{1}{\alpha}|Color_j(C)| \leq |Color_i(C)| \leq \alpha|Color_j(C)|$ for every $i, j \in \{1, ..., l\}$, where $Color_i(C)$ is the set of vertecies with color i in C.

We will study the following problem from a Parameterized Complexity point of view with the parameters k and l, which represent the solution size and number of colors respectively.

α-Balanced Cluster Vertex Deletion ($\alpha-BCVD$)	
Input:	A colored graph (G, c) and integer k.
Output:	Does there exist a set $X \subseteq V(G)$ where $\|X\| \leq k$ so that $G-X$ is an α-balanced cluster graph.

This is just the *CLUSTER VERTEX DELETION* problem when $l = 1$. One of the main methods used in kernels for *CLUSTER VERTEX DELETION* is identifying modules with more than k vertecies, and then deleting one of these vertecies. This does no longer work in the same way since the vertecies may have different colors. Another kind of method used in [8] to obtain an $\mathcal{O}(k^{\frac{5}{3}})$ kernel is using an expansion lemma and finding some part of the graph that must be included in an optimal solution. This argument also no longer works since it might no longer be optimal due to the colors.

We will prove the following theorems. We start by using the well-known branching technique for *CLUSTER VERTEX DELETION* to obtain an *FTP* algorithm.

Theorem 1. *There is a FTP algorithm for $\alpha-$BALANCED CLUSTER VERTEX DELETION, running in time $\mathcal{O}(3^k(|V| + |E|))$.*

Then we show that the problem admits a polynomial kernel when parameterized by k and l.

Theorem 2. *There is a kernel with $\mathcal{O}(k^2 l)$ vertecies for $\alpha-$BALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V|+|E|))$.*

We also give a polynomial kernel when the problem is parameterized by only k.

Theorem 3. *When $\alpha > 1$ there is a kernel with $\mathcal{O}(k^4)$ vertecies for $\alpha-$BALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V|+|E|))$.*

Finally for the special case when $\alpha = 1$ we are able to improve the kernel.

Theorem 4. *When $\alpha = 1$ there is a kernel with $\mathcal{O}(k^3)$ vertecies for αBALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V|+|E|))$.*

1.4 Thesis Outline

In Chapter 2, we will give some preliminaries and present the notation we will use and in Chapter 3 we will give some simple and useful result we will use throughout the thesis. Then we will prove the hardness in Chapter 4 and give a $\mathcal{O}(3^k(|V|+|E|))$ *FTP* algorithm in Chapter 5. We then give kernels with $\mathcal{O}(k^2 l)$ and $\mathcal{O}(k^4)$ vertecies in Chapter 6 and 7. For the special case when $\alpha = 1$, we will give a improved $\mathcal{O}(k^3)$ kernel in Chapter 8. Finally we mention some related open problems in Chapter 9.

Chapter 2

Preliminaries

2.1 Algorithms

2.1.1 Problems

An alphabet Σ is a set of symbols. We can use these symbols to construct strings which are a sequence of symbols. We denote all possible sequences of symbols from a given alphabet as Σ^*. A language L is some subset of all such sequences $L \subseteq \Sigma^*$. We will not give a detailed definition of a Turing machine since for our purposes its enough to think about a Turing machine as something with a finite set of instructions of how to solve a given problem. We will call this an algorithm. We define a decision algorithm for some language L as an algorithm which takes an input $x \in \Sigma*$, halts and returns whether or not $x \in L$. A problem is then recognizing whether or not any given input is in a language. If we have an instance x of some problem we say that x is a *YES*-instance if x is in the language, otherwise we say that x is a *NO*-instance.

2.1.2 Complexity

We need a way to measure the running time of our algorithms. We do this by counting the number of steps our algorithm needs to solve a given problem. The notion of steps is defined formally as the number of times our tape head in the Turing machine moves when the tape is initialized with the input. But for our use it is enough to think of one step as anything that can be done in constant time. That means something which can done in the same amount of time regardless of what the input is. So given an algorithm $\mathcal{A}$ and some instance of a problem x we can define $g(n)$ to be the number of steps A needs to decide x.

This is useful but we really want to see how good algorithm is in more general so for this we look at the worst case input for our algorithm for a given input size. We can define this as $f(n) = \max_{x \in \Sigma^*, |x| \leq n}\{g(x)\}$.

There is a useful notion of $\mathcal{O}$ notation which allows us to not worry about constants in our worst case running times. Given two functions f and g we say that $f = \mathcal{O}(g)$ if there exist some constants c and N such that $f(n) \leq cg(n)$ for all $n \geq N$. This means that the function g gives an upper bound for f when the input gets larger.

Some problems turns out to be easy to solve and some problems seems to be hard to solve. The main way we distinguish between hard and easy problems is whether or not the problem can be solved in polynomial time. This simply means that there is an algorithm which solves the problem in time $\mathcal{O}(n^c)$ for some $c \in \mathbb{N}$ where n is the size of the input. We denote the set of all such problems as P. We then have another class of problems which we can't necessary solve in polynomial time but we can at least verify a solution in polynomial time. We denote this set of problems as NP. Clearly $P \subseteq NP$ but whether or not $NP \subseteq P$ as well is the famous P vs NP problem. Something we will use to show that the problems we are working on are probably not in P is the notion of $NP-$hardness. We say that a problem is $NP-$hard if it's at least as hard as all other problems in NP. To show that a problem is $NP-$hard is enough to show that if we can solve the problem in polynomial time then we can also solve some other problem already know to be $NP-$hard in polynomial time. In practice we do this using what is called polynomial reductions. To show that some problem A is $NP-$hard we design an algorithm which takes as input an instance x of some known $NP-$hard problem and in polynomial time returns an instance x' of A where x' is a *YES*-instance if and only if x is a *YES*-instance. We say that a problem is $NP-$complete if it's both $NP-$hard and in NP.

2.1.3 Parameterized Complexity

Even if some problem $X \notin P$, that does not mean that all instances of problems in X are hard. For example there are a lot of graph problems which are not generally in P but if we restrict the graphs to being just trees then they are in P. So if we have some language $L \notin P$ we could still find some $L' \subset L$ where $L' \in P$. One way to formalize this is in the language of parameterized complexity theory which we will now introduce.

A parameterized problem is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance $(x, k) \in \Sigma \times \mathbb{N}$ we call k the parameter. A Fixed-Parameter Algorithm (FTP Algorithm) for a parameterized problem $L \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm which takes as input an instance $(x, k) \in \Sigma^* \times \mathbb{N}$

and correctly decides whether or not $(x, k) \in L$ in time bounded by $f(k)|(x, k)|^c$ for some c and computable function f. If there exist such an algorithm problem we say that the problem is in *FTP*. The idea is that k tells us something about the structure of the input so we can solve the problem in polynomial time when k is fixed. Be briefly note that there is something analogous to $NP-$hardness in Parameterized Complexity which is a whole hierarchy of hardness called the $W-$hierarchy for problems believed to not be in *FTP*. See the book by Cygan et al. [3] for more on parameterized complexity.

2.1.4 Kernelization

The central idea of a kernelization is simply reducing the size of the problem. For our usage a kernelization algorithm for a parameterized problem L is a algorithm which takes as input an instance $(x, k) \in L$ and return in time polynomial in $|(x, k)|$ a instance (x', k') where $x', k' \leq h(k)$ for some computable function h and $(x, k) \in L$ if and only if $(x', k') \in L$. $h(k)$ is called the size of the kernel. So if we find a kernel we have managed to bound the size of the problem by some function of k. Our goal is in general to give a polynomial bound but this is not always possible. We will structure our kernelization algorithms as a set of reduction rule which given certain conditions are satisfied takes in an instance (x, k) and returns a new smaller instance (x', k'). We say that the reduction rule is safe if $(x, k) \in L$ if and only if $(x', k') \in L$. See the book by Fomin et al. [4] for more about kernelization.

2.2 Graphs

A graph is a tuple $G = (V, E)$, where V and E are finite sets and the elements of E are unordered distinct pairs from V. The elements in the sets V and E are called vertecies and edges respectively, and denoted $V(G)$ and $E(G)$. We denote the elements of $E(G)$ by the 2 elements of $V(G)$. For example, $v_1v_2 \in E(G)$ is the edge between $v_1, v_2 \in V(G)$. The graphs we consider are simple which means they do not have self edges, so $vv \notin E(G)$ for all $v \in V(G)$. Given a graph G, we say that a graph H is a subgraph of G, denoted $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. If $H \subseteq G$ we say that H is an induced subgraph if for any edge $uv \in V(G)$ we have that $uv \in E(H)$ if and only if $u, v \in V(H)$. For a vertex set $U \subseteq V(G)$ the graph induced by U written $G[U]$ is the subgraph with vertex set V and edge set all including all edges which have both endpoints in V. If $X \subseteq V(G)$ we let $G - X$ denote the induced subgraph $G[V(G) \backslash X]$.

The complete graph with n vertecies, denoted K_n is the graph where every pair of distinct vertecies are connected by an edge. A clique is an induced subgraph which is complete. For a integer i let $[i] = \{1, ..., i\}$. A path in a graph is a sequence of vertecies $v_1....v_n$ where $v_i v_{i+1} \in E(G)$ for all $i \in [n-1]$. If such a path exist we say that v_1 and v_n are connected. A component in a graph is a subgraph induced by a maximum set of vertecies which are all connected. A cluster graph is a graph where all components are cliques. Let P_3 denote the graph with 3 vertecies u, v, w where $uv, vw \in E(P_3)$ and $uw \notin E(P_3)$. We usually denote this P_3 as uvw. We say that a graph G contain a P_3 if G contains a P_3 as an induced subgraph.

Given a vertex v and graph G, the open neighborhood of v, written $N^G(v)$ is defined to be the set of all vertecies $u \in V(G)$ such that $uv \in E(G)$, and the closed neighborhood is defined as $N^G[v] = N(v) \cup \{v\}$. We usually omit G when it's clear by the context. We say that u and v are twins if $N[u] = N[v]$. Given a set of vertecies U we let $N[U] = \bigcup_{v \in U} N(v)$ and $N(U) = N[U] \backslash U$. A module is a is a set of vertecies M such that $N(u) \backslash M = N(v) \backslash M$, for all $u, v \in M$.

2.2.1 Colored graphs

A *colored graph* is a tuple (G, c) where G is a graph and c is a function $c : V(G) \to \mathbb{N}$. We call $c(v)$ the color of the vertex v. Also we let $|c|$ denote the size of the image of c. We will refer to $|c|$ as l. In general we assume c is a function $c : V \to [l]$, since we can always relabel the colors. If $H \subseteq G$, let $Color_i(H)$ be the set $\{v \in V(H) | c(v) = i\}$ and let $Color_i^*(H)$ be the set $Color_i(N_G[V(H)])$. We say that H is balanced for some rational constant $\alpha \geq 1$ if $|Color_i(H)| \leq \alpha |Color_j(H)|$ for all $i, j \in [l]$ otherwise we say that H is unbalanced. We also say that two colors i and j are balanced in H if $\frac{1}{\alpha}|Color_i(H)| \leq |Color_j(H)| \leq \alpha |Color_i(H)|$, otherwise we say that i and j are unbalanced in H. We introduce the notation $d_H(i, j)$ to mean $||Color_i(H)| - |Color_j(H)||$ and $d_H^*(i, j)$ to be $||Color_i^*(H)| - |Color_j^*(H)||$, when the context is clear we sometimes just write $d(i, j)$ and $d^*(i, j)$. And finally we write $i = max(H)$ if $|Color_i(H)| \geq |Color_j(H)|$ for all $j \in [l]$ and $min(H)$ the same just smallest color class. $max^*(H)$ and $min^*(H)$ are the same just with $Color^*(H)$ instead.

For the rest of the paper we assume $\alpha \geq 1$ is a rational constant, $\alpha = \frac{a}{b}$, where $a \geq b \geq 1$ are integers with common factors.

Chapter 3

Some basic results

We will now presents some useful results which are all well known and used in kernelizations of the *CLUSTER VERTEX DELETION* problem, but we provide proofs for completeness. We usually write *CVD* for the *CLUSTER VERTEX DELETION PROBLEM* and $\alpha-BCVD$ for the $\alpha-$*BALANCED CLUSTER VERTEX DELETION* problem.

Lemma 3.1. *[3] A graph G is a cluster graph if and only if G contain no P_3 as an induced subgraph.*

Proof. First assume G is a cluster graph. If there is any $P_3 = uvw$ in G, then $uw \notin E(G)$ and since u and w are in the same component, this component can not be a clique. So G is not a cluster graph.

Now assume G has no P_3. Then if any component is not a clique it must contain two vertecies u, w where $uw \notin E(G)$, and since u and w are in the same component there must be a path $uv_1...v_nw$ between them. Now let v_i be the last vertex on the path, where $uv_i \in E(G)$, then uv_iv_{i+1} where v_{i+1} could be equal to w is a P_3. □

Lemma 3.2. *If u has a twin in $V(G)$, then $G - \{u\}$ has a P_3 if and only if G has a P_3.*

Proof. Assume that $G - \{u\}$ has a P_3, then clearly G does as well. Now if G has a P_3 and $G\backslash\{u\}$ does not then the P_3 must have included u, but since u has a twin, call it v. Any P_3 with u is still a P_3 if u is replaced with v, so G also has a P_3. □

Lemma 3.3. *Let G be a graph with a clique module U and $U' \subset U$, where $U - U' \neq \emptyset$, then G has a P_3 if and only if $G - U'$ has a P_3.*

Proof. Assume that G has a P_3. There must be some vertex $v \in U$ which is also in $G - U'$. Label the vertecies in U' as $v_1, ..., v_n$ and let $G_i = G - \{v_1, ..., v_i\}$. For any $i \in [n]$, v_i has a

twin v in G_{i-1} since U is a module which means that v_i and v has the same neighborhood outside U, and since U is a clique v and v_i has the same neighborhood in U as well. So G_i has a P_3 as well by Lemma 3.2. On the other hand if $G - U'$ has a P_3 then clearly G does as well. □

Lemma 3.4. *If G is a graph with a clique module U, $U' \subseteq U$ and $|U \backslash U'| > k$, then S^* is a solution to the CVD instance (G, k) if and only if S^* is a solution to the CVD instance $(G - U', k)$.*

Proof. Let $G' = G - U'$. Start by assuming (G, k) is an *YES*-instance with solution S^*, then clearly S^* is a solution to (G', k) as well since $G' \subseteq G$.

For the other direction assume that (G', k) is a *YES*-instance with solution S^*. We claim that S^* is a solution to (G, k) as well. To see why assume that S^* is not a solution to (G, k) , which by Lemma 3.1 means that $G - S^*$ has some P_3. Since $G' - S = (G - S^*) - U'$ we see that if we apply Lemma 3.3 to the graph $G - S^*$ that $G - S$ must have have P_3, since $(U \backslash S^*) \backslash U' \neq \emptyset$ since $|U \backslash U'| > k$ and $|S^*| \leq k$ (which are the conditions to apply Lemma 3.3). This contradicts the fact that S^* is a solution to (G', k). □

The following Lemma is from Koana and Nichterlein in [7].

Lemma 3.5. *We can find an induced P_3 in G in time $\mathcal{O}(|V| + |E|)$.*

Chapter 4

Hardness

In this section we will prove the hardness of $\alpha-BCVD$. We will do this by reducing from the well known *NP*-complete problem *CLUSTER VERTEX DELETION* (*CVD*) [5].

CLUSTER VERTEX DELETION	
Input:	A graph $G = (V, E)$ and integer k.
Output:	Does there exist a set $X \subseteq V$, where $\|X\| \leq k$, so that $G - V$ is a cluster graph

Algorithm 4.1
Input: Instance of *CVD* (G, k), integer l and rational number $\alpha = \frac{a}{b} \geq 1$.
Output: Instance of $\alpha-BCVD$ (G', k').

Given an instance of *CVD* (G, k), and the value of l. We create a instance of $\alpha-BCVD$ (G', k') with coloring c, the following way.

- For each $v \in V(G)$, create and add the vertecies $v_1, v_2, ..., v_l$ to $V(G')$, where $c(v_i) = i$ for all $i \in [l]$. Also add the edges to $E(G')$ such that $G'[v_1, ..., v_l]$ becomes an induced clique.
- For each $uv \in E(G)$ put $u_i v_j$ in $E(G')$, for all $i, j \in [l]$.
- Finally let $k' = lk$.

Theorem 5. *α-BALANCED CLUSTER VERTEX DELETION is NP-complete for any value of l and $\alpha = \frac{a}{b} \geq 1$.*

Proof. First it is easy to see that $\alpha-BCVD$ is in *NP* since given a proposed solution X, we can simply check that all the components C of $G - X$ are cliques, and that $\frac{1}{\alpha}|Color_i(C)| \leq$

$|Color_j(C)| \leq \alpha |Color_i(C)|$ for i, j in $[l]$. This can clearly be done in polynomial time.
We now want to show that $\alpha-BCVD$ is *NP*-hard by reducing from the *NP*-Hard problem *CVD*. Given an instance (G, k) of *CVD*, apply Algorithm 4.1 and let (G', k') be the resulting instance of $\alpha-BCVD$. We now want to show that (G, k) is a *YES*-instance of *CVD* if and only if (G', k') is a *YES* of $\alpha-BCVD$.

First assume (G, k) is a *YES*-instance with solution X. We then create a solution X' to (G', k') the following way. For each $v \in X$, add $v_1, ..., v_l$ to X'. To see that X' is a solution we observe that if there are any $P_3 = u_a v_b w_c$ in $G' - X'$ then u, v, w are all distinct vertecies since otherwise $u_a w_c$ is an edge in $E(G')$. So this P_3 correspond to the P_3 uvw in $G - X$ since $u, v, w \notin X$ and the fact that $ua, u_b \in E(G'), u_a wc \notin E(G')$ means that $uv, vw \in E(G), uw \notin E(G)$. This a contradiction. For the color ratios of the clusters in $G' - X'$, we observe that the clusters in $G' - X'$ have the same number of vertecies from each color, since for each $v \in X$ we put one vertex from each color in X' and these vertecies must all be in the same cluster in $G' - X'$ since $G[v_1, ..., v_l]$ is a clique. So the ratios will be satisfied for any α. We also see that $|X'| \leq lk = k'$, since l vertecies are added for each vertex in X and $|X| \leq k$.

Now assume (G', k') is a *YES*-instance with solution X'. We construct a solution X to (G, k) by inserting $v \in V(G)$ into X, for all $v \in V(G)$ if $\{v_1, ..., v_l\} \subseteq X'$. So see why X is a solution observe that if there is any $P_3 = uvw$ in $G - X$ then there must be corresponding vertecies $u_i, v_j, w_k \notin X'$. Then By the way G' was constructed $u_i v_j, v_j w_k \in E(G')$ and $u_i w_k \notin E(G')$, so this is a P_3 in $G' - X'$, a contradiction. We also see that $|X| \leq \frac{lk}{l} = k$ since there must be at least l vertecies in X' for every vertex we add to X. $\square$

Chapter 5

FTP-algorithm for $\alpha-BCVD$

We will give a simple branching algorithm for $\alpha-BCVD$. The algorithm will be presented as a series of steps to be applied exhaustively. We let (G, k) be an instance of $\alpha-BCVD$.

(Rule 5.1) If $k < 0$, then return *NO*.

(Rule 5.2) If there exist a $P_3 = uvw$ in G, branch to the 3 instances $(G - \{u\}, k - 1), (G - \{v\}, k - 1)$ and $(G - \{w\}, k - 1)$.

Lemma 5.1. *Rule 5.2 is safe and can be done in time* $\mathcal{O}(|V| + |E|)$.

Proof. By Lemma 3.5 we can find a P_3 in $\mathcal{O}(|V| + |E|)$ time. For the safeness we note that if one of the branches is a *YES*-instance then clearly (G, k) is as well. For the other direction if (G, k) is a *YES*-instance and there is a P_3 in G, then one of the vertecies in this P_3 clearly have to be deleted, so we try all 3 possibilities. □

By applying rules 5.1 and 5.2 we obtain a cluster graph. However this graph may have unbalanced cliques. To fix this we apply the following algorithm.

Algorithm 5.1 Fix Clique
Input: Clique C.
Output: A set $X \subseteq V(C)$, where $C - X$ is balanced

1: $X \leftarrow$ Empty set.
2: **while** $|Color_i(C) \backslash X| < \frac{1}{\alpha}|Color_j(C) \backslash X|$ for some $i, j \in [l]$ **do**
3: $\quad X \leftarrow X \cup \{v\}$ for some $v \in Color_j(C) \backslash X$
4: **return** X.

Lemma 5.2. *Algorithm 5.1 can be applied in time $\mathcal{O}(|V(C)|)$, where C is the clique in the input.*

Proof. Start by sorting the colors using count sort in $O(|V(C)|)$ time. Then starting with the biggest color simply delete vertecies until this color is balanced with the smallest color. Then move to the next biggest color and repeat this process until C is balanced. □

(Rule 5.3) For each component C in G which is not balanced, apply Algorithm 5.1 on C to get X_C and let $X = \bigcup X_C$. If $|X| \leq k$ then return *YES* otherwise return *NO*.

Lemma 5.3. *Reduction 5.3 is safe and can be applied in time $\mathcal{O}(|V| + |E|)$.*

Proof. First we need to compute the components of G. This can be done in $\mathcal{O}(|V| + |E|)$ time with a simple DFS-search. Then to find a clique C which is not balanced we sort the colors in each clique using count sort which is done inn $\mathcal{O}(|V(C)|)$ time for each clique so $\mathcal{O}(|V|)$ time in total. For the safeness we clearly need to balance all the cliques and the only way to do this is to delete vertecies from the colors which have to many vertecies in some clique. If we have to do more than k deletions to balance all the cliques then we clearly have a *NO*-instance. □

We can now prove theorem 1.

Theorem 1. *There is a FTP algorithm for $\alpha-$BALANCED CLUSTER VERTEX DELETION, running in time $\mathcal{O}(3^k(|V| + |E|))$.*

Proof. Rule 5.2 is first applied exhaustively. The result is up to 3^k branches where G is a cluster graph and $k \geq 0$ since any branch with to many deletions is killed by Rule 5.1. For each branch, Rule 5.3 gets applied in $\mathcal{O}(|V| + |E|)$ time and returns either *YES* or *NO*. □

Chapter 6

$\mathcal{O}(k^2l)$ kernel for $\alpha-BCVD$

6.1 Overview

We give a overview of the method we will use to obtain a $\mathcal{O}(k^2l)$ vertex kernel. We start by constructing a $3k-$approximate solution S which we will use as a modulator. We then look at the cliques in $G - S$ which we denote as the set $\mathcal{C}$. Our next step is to construct a set $M \subseteq V(G)$ of what we call marked vertecies, and bound the size of this set. We then also bound the number of cliques in $\mathcal{C}$. All of these steps are already used in the best kernel for the *CVD* problem.

The reason for constructing the set M is that the vertecies in a clique $C \in \mathcal{C}$ which are not in M, form a module. Most of the chapter is concerned with bounding the number of these unmarked vertecies.

To bound the number of unmarked vertecies we split the cliques $C \in \mathcal{C}$ into two cases. The first is where the distances between the number of vertecies in two colors are large. We can then remove a unmarked vertecies from colors which have a lot of vertecies and b unmarked vertecies for colors which have few vertecies, as long as there are a least a unmarked vertecies with each color to delete, and that we do not bring the number of unmarked vertecies in the clique below k when we delete vertecies.

If some color i have less that a unmarked vertecies we denote C as a missing cliques, and we are able to bound the number of vertecies in these cliques by the number of marked vertecies, since if C contain a lot of vertecies with color i, then it must contain a lot of marked vertecies.

If we can not delete unmarked vertecies since the number of unmarked vertecies in the clique would become less than k, the situation is more complicated. We define something

called linear cliques, which are cliques with less than θlk vertecies (Where θ is a constant), and argue that there are at most $5k$ cliques which are not linear and are missing unmarked vertecies because these cliques must contain a lot of marked vertecies. This give us the desired bound for these kind of cliques. For linear cliques we divide the cliques up even further.

We say that a linear clique $C \in \mathcal{C}$ is connected if the unmarked vertecies in C are connected to S. We are able to bound the vertecies in these cliques by arguing that if we have a lot of such vertecies there is some vertex in S which have to many connections to at least two different cliques in $\mathcal{C}$.

For cliques which are not connected we define what we call hidden cliques which are cliques C where the unmarked vertecies are not connected to S, C is balanced and C contains a lot more unmarked vertecies than marked vertecies. We are able to bound the number of vertecies in hidden cliques by again dividing the cliques in the 2 cases when the color distances are either small or large, but now we are able to remove the requirement that we do not bring the number of unmarked vertecies below k to obtain a better bound than before.

For cliques C which are not hidden one of the following conditions are true. C must be connected, C is unbalanced (at most $4k$ such cliques) or the number of unmarked vertecies is bounded by a factor times the number of marked vertecies in the clique. We are able to deal with all these cases.

Now if the distances between colors are small, then we argue that if we have a lot of vertecies with each color the clique must be "very" balanced so we can safely remove an unmarked vertex. If one of the conditions for applying this reduction fails the analysis is almost the same as when the distances are large.

6.2 Marked vertecies

We call the the instance of $\alpha-BCVD$ with $l = 2, k = 0$ and the graph being a single vertex of color 1 the trivial *NO*-instance, and we call the instance where the vertex set of the graph is empty instead the trivial *YES*-instance. In this section we will give some basic reduction rules that are all well known from work on the regular *CVD* problem. We apply all reduction rules from the top exhaustively, and whenever we apply an reduction we also go back to the start and recompute everything. We will generally assume the data structure of the graph is an adjacency list for the runtime analysis.

(Rule 6.1) If $k < 0$, return the trivial *NO*-instance.

Now a lot of the reduction rules are based on first applying the following trivial $3k$ approximation algorithm the same way as done by Le et al. [8].

Algorithm 6.1 Approximation algorithm

$X \leftarrow \{\}$
while There exist $P_3 = uvw$ in $G - X$ **do**
  $X \leftarrow X \cup \{u, v, w\}$
return X

Lemma 6.1. *Algorithm 6.1 is a 3-approximation algorithm for CVD and can be applied in time* $\mathcal{O}(|V|(|V| + |E|))$.

Proof. Let X be the set returned by the algorithm. It easy to see there are no P_3 in $G - X$,since one of it's vertecies must be in X. Let S be a solution to the CVD instance (G, k). For each set of vertecies u, v, w that is added to X at least one of these vertecies must be in S, otherwise there would a P_3 in $G - S$. Therefore we see that $|S| \geq \frac{|X|}{3}$. We can find a P_3 in a graph in time $\mathcal{O}(|V| + |E|)$ by Lemma 3.5, so we get a running time bounded by $\mathcal{O}(|V|(|V| + |E|))$. □

Now we define S be the approximate solution (modulator) returned by Algorithm 6.1, and let $\mathcal{C}$ be the set of cliques in $G - S$, where each element $C \in \mathcal{C}$ is a component of $G - S$. Whenever we apply any reduction rule we recompute S and $\mathcal{C}$.

We need a way to reason about the cliques in $C \in \mathcal{C}$ after applying some solution to G. Given a solution S^* there is a natural mapping from C to some clique in $G - S^*$ that contain C. We make the following definition. If $X \subseteq V(G)$, $C' \subseteq C$ for some $C \in \mathcal{C}$ and $G' \subseteq G$, we define the mapping $f_X^{G'}(C') = G'[N_{G'}[C] \backslash X]$. We usually omit G' and just write $f_X(C')$ when the context is clear. So f is more or less a mapping from a clique in $\mathcal{C}$ to the clique in $G - X$ containing some of this clique.

If our approximate solution is too big, then we have a *NO*-instance.

(Rule 6.2) If $|S| > 3k$ then return the trivial *NO*-instance.

Lemma 6.2. *Reduction rule 6.2 is safe and can be applied in time* $\mathcal{O}(|V|)$.

Proof. By lemma 6.1 Algorithm 6.1 is a 3-approximation algorithm, so if $|S| > 3k$ then there is no solution of size at most k. Also we can check the size of S in time $\mathcal{O}(|V|)$ time since $|S| \leq |V|$. □

For any clique C we say that color i and j are *balanced* in C if $\frac{1}{\alpha}|Color_i(C)| \leq |Color_j(C)| \leq \alpha|Color_i(C)|$, otherwise we say that i and j are *unbalanced*. If C have any unbalanced colors we say that C is *unbalanced* otherwise we say that C is *balanced*.

In the *CVD* problem, isolated cliques can simply be removed from G without decreasing k, since these cliques will not affect the problem at all, but in $\alpha-BCVD$ we need to balance the isolated clique before deleting it. This means we need to spend some of our deletion budget if the isolated clique is not balanced.

(Rule 6.3) If $C \in \mathcal{C}$ is a isolated clique, then apply Algorithm 5.1 on C to get X, and return $(G - V(C), k - |X|)$.

Lemma 6.3. *Reduction rule 6.3 is safe and can be applied in time* $\mathcal{O}(|V| + |E| \log |V|)$.

Proof. Since C is an isolated clique it does not matter what vertex we delete when it comes to deleting P_3-s. Now if two colors are not balanced, the only way to fix this is to remove a vertex from the largest of these color (the color with the most vertecies). This is what the algorithm does. After the Algorithm finishes $C \backslash X$ must be balanced by the stopping condition of the loop in Algorithm 5.1.

To find an isolated clique we simply go through each clique in $C \in \mathcal{C}$ and see if there is an edge between C to a vertex outside C. This can be done in time $\mathcal{O}(|V|+|E|)$ by checking each edge and marking a clique as not isolated whenever an edge goes from C to a vertex outside C. Then apply Algorithm 5.1 to a isolated clique in time $\mathcal{O}(|V|)$ if one is found. Finally deleting the vertecies is done in $\mathcal{O}(|E| \log |V|)$ time by going through the adjacency list using a hash set. □

Now we will introduce a algorithm called Mark. This algorithm is used in the current best known kernel for *CVD* in [8]. We keep a set of edges $mark(s)$ for each $s \in S$ with the property that for each edge $uv \in mark(s)$, suv is a P_3. We also let M be the set of vertecies which which is contained in an edge in $mark(s)$, for some s, and we only add an edge uv to $mark(s)$, if neither u or v is in M.

Algorithm 6.2 Mark

1: $M \leftarrow \emptyset$
2: $mark(s) \leftarrow \emptyset$ for all $s \in S$
3: **for** $i = 1, .., k+1$ **do**
4: **for** $s \in S$ **do**
5: **if** $\exists C \in \mathcal{C}\ \exists uv \in E(C)$ so $su \in E(G)$ and $sv \notin E(G)$ and $\{u, v\} \cap M = \emptyset$ **then**
6: $M \leftarrow M \cup \{u, v\}$
7: $mark(s) \leftarrow mark(s) \cup \{uv\}$

Lemma 6.4. *Algorithm 6.2 can be applied in time* $\mathcal{O}(|V|^2(|V| + |E|))$.

Proof. Keep a list of all vertecies not in M, since those are the only vertecies we need to consider in each iteration of the inner loop. To find the vertecies u and v we run a BFS from s. If we find a vertex u with a neighbor u with distance 1 more from s than itself, and where u and v are in the same clique. These two vertecies become u and v. This BFS can be done in time $\mathcal{O}(|V| + |E|)$ so we get a total time of $\mathcal{O}(|V|^2(|V| + |E|))$, since the loop starting on line 5 will run a total of less than $|V|^2$ times. □

We will for the rest of the paper denote the set M as the result of running Algorithm 6.2. Whenever we apply any of the reduction rules from now on we will rerun algorithm 6.2 and get a new set M. We call the vertecies of M *marked vertecies*, and the vertecies of $V(G) \backslash M$ *unmarked vertecies*.

If a vertex s is in a part of more than k P_3-s, which has no vertex other than s in common, then s must be deleted.

(Rule 6.4) If $|mark(s)| \geq k + 1$ for some $s \in S$, then return $(G - s, k - 1)$.

Lemma 6.5. *Reduction 6.4 is safe and can be applied in time* $\mathcal{O}(|V| + |E|)$

Proof. Assume that $|mark(s)| \geq k + 1$ for some $s \in S$. We want to prove that if S^* is a solution then $s \in S^*$. So assume that $s \notin S^*$. For any edges $uv, xy \in mark(s)$ we have that $\{u, v\} \cap \{x, y\} = \emptyset$ since M contains the vertecies of one of these edges when the other is considered, then since $s \notin S^*$ for each $uv \in mark(s)$, $u \in S^*$ or $v \in S^*$, since suv is a P_3. So $|S^*| \geq k + 1$ which is a contradiction.
To apply the algorithm we need to check the size of $mark(s)$ for each $s \in S$. So we simply count the size of these sets. The total size of all of them can be at most $|V|$. So this takes at most $\mathcal{O}(|V|)$ time. Deleting a vertex is done in $\mathcal{O}(|E|)$ time in an adjacency list. □

For any $s \in S$, let $N_{\mathcal{C}}(s) \subseteq \mathcal{C}$ be the set of all cliques $C \in \mathcal{C}$ where s is connected to C. We will use the following rule to bound the number of cliques in $\mathcal{C}$.

(Rule 6.5) If some $s \in S$ satisfies $|N_{\mathcal{C}}(s)| > k+1$, then return $(G-s, k-1)$.

Lemma 6.6. *Reduction rule 6.5 is safe and can be applied in time $\mathcal{O}(|V|+|E|)$.*

Proof. Assume S^* is a solution to (G,k) and $s \notin S^*$. If $|N_{\mathcal{C}}(s)| > k+1$ then s is connected to at least $k+2$ cliques $C_1, ..., C_{k+1}, C_{k+2} \in N_{\mathcal{C}}(s)$. Since $|S^*| \leq k$ at least 2 of the cliques C_a, C_b have no vertecies in S^*. This means that there is a P_3 in $G - S^*$.
To apply the Reduction rule we need to calculate $|N_{\mathcal{C}}(s)|$ for each $s \in S$. We go through all the cliques $C \in \mathcal{C}$. If there is any vertex $v \in C$ where $vs \in E(G)$ for some $s \in S$, then we increase the count of $|N_{\mathcal{C}}(s)|$ by one. This can all be done in time $\mathcal{O}(|V|+|E|)$. In the end we just need to check the size of $|N_{\mathcal{C}}(s)|$ and potentially delete a vertex in time $\mathcal{O}(|E|)$. □

Reduction rule 6.3 and 6.5 then gives us the following bound on the number of cliques in $\mathcal{C}$.

Observation 6.1. $|\mathcal{C}| \leq 3k^2 + 3k$.

Proof. From Reduction 6.3 there are no isolated cliques, which means that all cliques in $\mathcal{C}$ must be connected to some vertex in S. Then by Reduction rule 6.5 any vertex in S is connected to at most $k+1$ different cliques. Then since $|S| \leq 3k$ by Reduction rule 6.2 there are at most $(k+1)3k = 3k^2 + 3k$ cliques in $\mathcal{C}$. □

The number of marked vertecies is also bounded.

Observation 6.2. $|M| \leq 6k^2$

Proof. Every time a pair of vertecies is added to M an edge is also added to $mark(s)$ for some $s \in S$. Since for each $s \in S$, $|mark(s)| \leq k$ and $|S| \leq 3k$ from Reduction rules 6.4 and 6.2 respectively, there can be at most $3k(2k) = 6k^2$ vertecies in M. □

We already now have a good bound on the number of marked vertecies. The rest of the chapter will be concerned with giving a bound on the number of unmarked vertecies. The reason for dividing up the vertecies this way is the very useful and well known fact that the unmarked vertecies of the cliques in $\mathcal{C}$ are modules.

Observation 6.3. *For any $C \in \mathcal{C}$, $V(C) \backslash M$ is a module.*

Proof. Assume that there is some $C \in \mathcal{C}$ where $C - M$ is not a module, then there are two vertecies $u, v \in V(C) \backslash M$ so that $su \in E(G)$ and $sv \notin E(G)$, but then there vertecies u and v satisfies the requirements on line 6 of Algorithm 6.2. If the loops ends before the vertecies can get marked ,then Reduction rule 6.4 would be applied, and algorithm 6.2 would be run again. □

6.3 Main Rules

In this section we present the two main Reduction rules. One of the rules is for when the maximum distance (in terms of number of vertecies) between two colors are large and the other one is for when this distance is small. We already obtain a $\mathcal{O}(k^3 l)$ using these rules and an a couple of Observations.

We note that a and b are the integers from the definition $\alpha = \frac{a}{b}$. If some colors i and j are balanced in some clique C and $|Color_i(C)| \geq |Color_j(C)|$, then if we add a vertecies with color i and b vertecies with color j to C, color i and j will still be balanced. This works in the other direction as well (deleting vertecies instead) if certain conditions are satisfied. We note that we are talking about any cliques, and not just cliques in $\mathcal{C}$.

Lemma 6.7. *If for any cliques C and C', $|Color_i(C')| = |Color_i(C)| + a$ and $|Color_j(C')| = |Color_j(C)| + b$, then if $|Color_i(C)| \geq |Color_j(C)|$, and i and j are balanced in C, they are also balanced in C'. If also $|Color_i(C')| \geq |Color_j(C')| + a$, then i and j are balanced in C' if they are balanced in C.*

Proof. Let C and C' be the cliques where $|Color_i(C')| = |Color_i(C)| + a$, $|Color_j(C')| = |Color_j(C)| + b$ and $|Color_i(C)| \geq |Color_j(C)|$. First assume that i and j are balanced in C. It is clear that $|Color_j(C')| \leq \alpha |Color_i(C')|$, since $|Color_i(C)| \geq |Color_j(C)|$ and $a \geq b$ (Since we "add" at least as many vertecies with color i as j). To verify that $|Color_i(C')| \leq \alpha |Color_j(C')|$ as well we have.

$$\begin{aligned} |Color_i(C')| &= |Color_i(C)| + a \\ &\leq \frac{a}{b}|Color_j(C)| + a \\ &\leq \frac{a}{b}(|Color_j(C')| - b) + a \\ &= \frac{a}{b}|Color_j(C')| - \frac{ab}{b} + a \\ &= \alpha |Color_j(C')|. \end{aligned} \tag{1}$$

We use the assumption that i and j are balanced in step (1). So we can conclude that i and j are balanced in C' as well.

Now for the second part assume that $|Color_i(C')| \geq |Color_j(C')| + a$ and that i and j are balanced in C'. We clearly have that $|Color_j(C)| \leq \alpha |Color_i(C)|$, since $|Color_i(C')| \geq |Color_j(C')| + a$. For the other direction we get.

$$\begin{aligned}
|Color_i(C)| &= |Color_i(C')| - a \\
&\leq \frac{a}{b}|Color_j(C')| - a \\
&= \frac{a}{b}(|Color_j(C)| + b) - a \\
&= \alpha |Color_j(C)| + a - a \\
&= \alpha |Color_j(C)|
\end{aligned}$$

So we can conclude that i and j are balanced in C as well. □

We only really need to check that the biggest and smallest color (with respect to the number of vertecies) is balanced to conclude that a clique is balanced.

Lemma 6.8. *For any clique C, if the color $max(C)$ and $min(C)$ are balanced, then C is balanced.*

Proof. For any colors $i, j \in [l]$, we have that $|Color_j(C)| \leq |Color_{max}(C)| \leq \frac{1}{\alpha}|Color_{min}(C)| \leq \frac{1}{\alpha}|Color_i(C)|$. This means that C is balanced. □

In the next reduction rule we will utilize Lemma 6.7 to delete unmarked vertecies. The set of unmarked vertecies in a clique form a module, so they can in some sense be treated as being the same vertex. So our only concern is the color ratios and bringing the number of unmarked vertecies in a clique C below k because then we can delete this "vertex" where as we could not before. The idea of the Reduction rule is then to delete a vertecies with colors which could become the maximal color after applying a solution, and b vertecies from any color which could potentially become the minimal color after applying a solution. Then if we add these vertecies back, the same solution still works. Also if we have a solution for the original instance which is disjoint of any deleted vertex, then the same solution works for the reduced instance.

(Rule 6.6) If some $C \in \mathcal{C}$ satisfies the following conditions:

- $d^*_C(min^*(C), max^*(C)) > 2k + 2a$,
- $|Color_i(C) \backslash M| > a$,
- $|V(C) \backslash M| > k + a * l$,

Then we construct a set of vertecies A the following way. For each $i \in [l]$

- If $d^*_C(i, max^*(C)) \leq k + a$, then put a vertecies from $Color_i(C) \backslash M$ in A.
- If $d^*_C(i, min^*(C)) \leq k + b$, then put b vertecies from $Color_i(C) \backslash M$ in A.

Finally return $(G - A, k)$.

Lemma 6.9. *Reduction rule 6.6 is safe and can be applied in time $\mathcal{O}(|V| + |E| \log |V|)$.*

Proof. We first note that if the conditions for applying Reduction rule 6.6 is satisfied, then for any $i \in [l]$ we can not have both $d^*_C(i, max^*(C)) \leq k + a$ and $d^*_C(i, min^*(C)) \leq k + b$ be true at the same time, since $d^*_C(min^*(C), max^*(C)) > 2k + 2a$.

Let $G' = G - A$, $C \in \mathcal{C}$ be the clique which Reduction rule 6.6 is applied to and $C_R = C - A$.

First assume (G', k) is a *YES*-instance with solution S^*, we claim that S^* is a solution to (G, k) as well. We first need to prove that there is no P_3 in $G - S^*$. We use Lemma 3.4 on the graph $G - S^*$ with $U = V(C) \backslash M$ (which is a module by Observation 6.3, and clearly a clique) and $U' = A$. Then $|U \backslash U'| > k$, since $|U| > k + al$ and $|U'| \leq al$. So since $(G' - S^*) = (G - S^*) - U'$, Lemma 3.4 and Lemma 3.1 gives us that $G - S^*$ has no P_3 since $G' - S^*$ does not.

Now let $C' = f^G_{S^*}(C)$ and $C'_R = f^{G'}_{S^*}(C_R)$. To verify that C' is balanced we only need to check that $min(C')$ and $max(C')$ are balanced by Lemma 6.8. It is clear that only colors i where $d^*_C(i, max^*(C)) \leq k$ or $d^*_C(i, min^*(C)) \leq k$ can become $max(C')$ and $min(C')$ respectively, since at most k vertecies can be deleted from any one color in $N[C]$. So if we let $i = max(C')$ and $j = min(C')$ we know that there are a vertecies with color i and b vertecies with color j in A. This means that $|Color_i(C'_R)| + a = |Color_i(C')|$ and $|Color_j(C'_R)| + b = |Color_j(C')|$. We now want to apply Lemma 6.7 to the cliques C' and C'_R. We know that i and j are balanced in C'_R, also since $d^*_{C_R}(i, j) \geq k$ we get that $|Color_i(C'_R)| \geq |Color_j(C'_R)|$, since at most k vertecies can be deleted from $Color^*_i(C_R)$. So by Lemma 6.7 color i and j are balanced in C' as well.

Now assume (G, k) is a yes instance with solution S^* and let $C' = f^G_{S^*}(C)$. We first

want to modify S^* such that $S^* \cap A = \emptyset$, where A is the set of vertecies from the Reduction rule. To do this we do the following. For each $v \in S^* \cap A$ with color i remove v from S^* and replace it with any vertex in $Color_i(C) \backslash (A \cup S^*)$. First assume there are enough vertecies in $Color_i(C) \backslash (A \cup S^*)$ for all i to do this replacement, and call the new solution S'.

To see that S' is still a solution we see that the color ratios of C' did not change. Let U' be the set of vertecies we removed from S^*, and let $U = V(C) \backslash M$. We then have that $U \backslash U' \neq \emptyset$ (since $|U'| \leq |S^*| \leq k$), and since $(G - S') - U' \subseteq G - S^*$ means that $(G - S') - U'$ does not have a P_3 since $G - S^*$ does not. This means that $G - S'$ does not have a P_3 by Lemma 3.3 and Lemma 3.1.

If this replacement was not possible due to too few vertecies in $Color_i(C) \backslash (A \cup S^*)$ for some i, then $|Color_i(C')| \leq 2a$ since if $|Color_i(C')| > 2a$ then there must be at least a vertecies in $Color_i(C') \backslash A$, which means the replacement would be possible. In this case we simply let $S' = S^* \backslash (Color_i(C) \cup A)$ (For all colors i where the replacement was not possible). To see that S' is still a solution let $C'' = f_{S'}(C)$ and let $U = V(C) \backslash M$ and U' be the set of all vertecies we removed from S^* to create S'. We observe that $U \backslash U \neq \emptyset$ and that $(G - S') - U' \subseteq G - S^*$ which means $(G - S') - U'$ does not have a P_3, so $G - S'$ does not have a P_3 by Lemma 3.3.

For the color ratios we increased the the size of $|Color_i(C')|$ so the only thing we need to verify is that $|Color_i(C'')| \leq |Color_{max(C')}(C')|$, which is true since $d^*_C(min^*(C), max^*(C)) > 2k + 2a$, which means $d_{C'}(max(C'), min(C')) > k + 2a \geq 2a$. So since $|Color_{max(C')}(C')| > 2a$ we get $|Color_i(C'')| < |Color_{max(C')}(C')|$, since $|Color_i(C')| \leq 2a$. Now let $S^* = S'$, $C' = f^G_{S^*}(C)$ and $C'_R = f^{G'}_{S^*}(C_R)$.

We want to prove that S^* is a solution to (G', k) as well. There is no P_3 in $G' - S^*$ since $G' - S^* \subset G - S^*$. Now for the color ratios let $i = max(C'_R)$ and $j = min(C'_R)$. We now want to apply the second part of lemma 6.7 to C'. We know that $d^*_{C_R}(i, max^*(C_R)) \leq k$ and $d^*_{C_R}(j, min^*(C_R)) \leq k$, which means $d^*_C(i, max^*(C)) \leq k + a$ and $d^*_C(j, min^*(C)) \leq k + b$. So there are a vertecies in $Color_i(C) \cap A$ and b vertecies in $Color_j(C) \cap A$ which means that $|Color_i(C'_R)| + a = |Color_i(C')|$ and $|Color_j(C'_R)| + b = |Color_j(C')|$. We also know that $|Color_i(C')| \geq |Color_j(C')| + a$ since that fact that $d^*_C(i, j) \geq k + a$ means that $d_{C'}(i, j) > a$. So all the conditions for Lemma 6.7 are satisfied which means that C'_R is balanced since C' is.

To apply the reduction rule we first need to determine if any cliques satisfy the first distance requirement. To this for any clique we simply sort the colors using count sort in $\mathcal{O}(|V(C)| + |N(C)|)$ time. We can do this for all cliques in total time of $\mathcal{O}(|V| + |E|)$ since

the total size of $N(C)$ for all cliques is bounded by $|E|$. Then we can then check the 2 other requirements in $\mathcal{O}(|V|)$ time. Then we simply go through the rest of the colors and check whether or not we need to delete any vertecies. In total we clearly delete at most $|V|$ vertecies, so the deletions can be done in $\mathcal{O}(|E| \log |V|)$ if we do them all at the same time by going through the adjacency list using a dictionary with all the vertecies to be deleted. □

We are now going to look at the case when Reduction rule 6.6 can not be applied due to the distance between the maximal and minimum color is too small. The idea is that if the distances are small but all the colors have a lot of vertecies, then the clique has to be balanced anyway.

Lemma 6.10. *If for any clique C we have that $|Color_i(C)|, |Color_j(C)| \geq c$ for some constant c, then if $d_C(i,j) \leq \frac{c(a-b)}{a}$, i and j are balanced in C.*

Proof. Assume without loss of generality that $|Color_i(C)| \geq |Color_j(C)|$. We first show that $|Color_i(C)| \leq \alpha |Color_j(C)|$.

$$\begin{aligned}
|Color_i(C)| &= \frac{a}{b}|Color_i(C)| - \frac{a-b}{b}|Color_i(C)| \\
&\leq \frac{a}{b}(|Color_j(C)| + \frac{c(a-b)}{a}) - \frac{a-b}{b}|Color_i(C)| && (1)\\
&\leq \frac{a}{b}(|Color_j(C)| + \frac{c(a-b)}{a}) - c\frac{a-b}{b} && (2)\\
&= \frac{a}{b}|Color_j(C)| + c\frac{a-b}{b} - c\frac{a-b}{b} && (3)\\
&= \alpha |Color_j(C)|
\end{aligned}$$

In step (1) we use the assumption that $d_C(i,j) \leq \frac{c(a-b)}{a}$ and $|Color_i(C)| \geq |Color_j(C)|$, so we get that $|Color_i(C)| \geq |Color_j(C)| + \frac{c(a-b)}{a}$. In step (2) we use our assumption that $|Color_i(C)| \geq c$. Finally in step (3) we simply multiply out $\frac{a}{b}$ in the first term.
Also since we assumed $|Color_i(C)| \geq |Color_j(C)|$ clearly $|Color_j(C)| \leq \alpha |Color_i(C)|$, which means that i and j are balanced in C. □

If $\alpha > 1$, then let $\beta = \frac{5a^2}{a-b} + 1$. We will now use Lemma 6.10 in the following Reduction rule which compliments Rule 6.6.

(Rule 6.7) If $\alpha > 1$ and for some $C \in \mathcal{C}$ the following conditions are satisfied:

- $d^*_C(min^*(C), max^*(C)) \leq 2k + 2a$,
- $|Color^*_i(C)| \geq \beta k$,
- $|V(C) \backslash M| > 1 + k$,

then let v be any vertex in $V(C) \backslash M$, and return $(G - \{v\}, k)$.

Lemma 6.11. *Reduction rule 6.7 is safe and can be applied in time* $\mathcal{O}(|V| + |E| \log |V|$.

Proof. Let $G' = G - v$, and let C be the clique the Reduction rule was applied to.
First assume S^* is a solution to (G, k). We then claim S^* is a solution to (G', k) as well. Clearly $G' - S^*$ does not have a P_3 since $G - S^*$ does not.
Now for the color ratios let $C' = f_{S*}(C)$ and $C'_R = f^{G'}_{S^*}(C_R)$. For any $i, j \in [l]$ we see that $d_{C_R}(max^*(C_R), min^*(C_R)) \leq 2k + 2a + 1$ since removing 1 vertex from each color increase this distance by at most 1 which means $d_{C'_R}(max(C'_R), min(C'_R)) \leq 3k + 2a + 1$, since S^* deletes at most k vertecies, so increases the distance by at most k. We now want to apply Lemma 6.10. Letting $c = \frac{(3k+2a+1)a}{a-b}$ Lemma 6.10 tells us that if $d_{C'_R}(i, j) \leq \frac{\frac{(3k+2a+1)}{a-b}(a-b)}{a} = 3k + 2a + 1$ and $|Color_i(C'_R)|, |Color_j(C'_R)| \geq c$, then i and j are balanced in C'_R. We know that $|Color_i(C'_R)| \geq |Color^*_i(C_R)| - k \geq \beta k - 1 - k = \frac{5a^2}{a-b}k + k - 1 - k \geq \frac{(ka+ka+ka+ka+ka)a}{a-b} \geq \frac{(3k+2a+1)a}{a-b} = c$ for all $i \in [l]$, which means C'_R is balanced.

For the other direction assume S^* is a solution to (G', k). And we will show S^* is a solution to (G, k) as well. let $C' = f_{S^*}(C)$ and $C'_R = C' - v$. For the color ratios it is clear that $d_{C'}(max(C'), min(C')) \leq 3k+2a$ then we can use precisely the same argument as above, but now for C'. To see that are no P_3 in $G - S^*$ we apply Lemma 3.3 with $U = V(C) \backslash M$ and $U' = \{v\}$ to the graph $G - S^*$. Then since $G' - S^* = (G - S^*) - U'$, $G - S^*$ does not have a P_3, since $(G - S^*) \backslash U'$ does not.

Applying the reduction can be done in time $\mathcal{O}(|V| + |E| \log |V|)$ the same way as reduction rule 6.6. □

We will in the rest of the chapter look at the several cases for when Reduction rules 6.6 and 6.7 can not be applied, and deal with these cases.

6.4 Missing cliques

We will start with one of the conditions for reduction rule 6.6 being applied which is that all of the colors have more than a unmarked vertecies.

Denote a clique $C \in \mathcal{C}$ where $|Color_i(C) \backslash M| < a$ for some $i \in [l]$ as a *missing clique*. Our goal in this section is to bound the number of vertecies in these cliques. Our first step is the Observation that if we have to do too many operations to balance all the cliques in $\mathcal{C}$, then we know we have a *NO*-instance.

(Rule 6.8) If $\sum_{C \in \mathcal{C}} |Color_{max(C)}(C)| - \alpha |Color_{min(C)}(C)| > 4\alpha k$, then return the trivial no instance.

Lemma 6.12. *Reduction rule 6.8 is safe and can be applied in time $\mathcal{O}(|V|)$.*

Proof. Assume $\sum_{C \in \mathcal{C}} |Color_{max(C)}(C))| - \alpha |Color_{min(C)}(C)| > 4\alpha k$. Then for any clique C, let $m = |Color_{max(C)}(C)| - \alpha |Color_{min(C)}(C)|$. If $m > 0$, C is not balanced since $|Color_{max(C)}(C)| > \alpha |Color_{min(C)}(C)|$. To balance C we need to either delete vertecies from $Color_{max(C)}(C)$ or bring vertecies with color $min(C)$ from S into C. We can do at most $4k$ of these operation since $|S| \leq 3k$ and we can delete at most k vertecies. Each of these operation can decrease m by at most α. So if the sum is greater than $4\alpha k$ not all cliques can be balanced.

To apply the reduction rule for each clique C we need to sort the colors in $O(|V(C)|)$ time using count sort and add the difference between the smallest and largest color to the total sum. So we get $\mathcal{O}(|V|)$ total time. □

We will use reduction rule 6.8 to bound the number of unmarked vertecies in missing cliques. The idea is that since one color i have very few unmarked vertecies, if there are a lot of unmarked vertecies in this clique, then i have to have a lot of marked vertecies, otherwise the clique become unbalanced. So in some sense we have to spend marked vertecies to add unmarked vertecies. For each $\alpha(l-1)$ unmarked vertecies a clique have, we have to to give i one marked vertex if we are not to increase the "unbalance of the clique". Then reduction 6.8 gives a bound of how much "unbalance" we can have in total.

Observation 6.4. *For any missing clique $C \in \mathcal{C}$, $\frac{|V(C) \backslash M| - a}{l-1} \leq \alpha(|V(C) \cap M| + a) + |Color_{max(C)}(C)| - \alpha |Color_{min(C)}(C)| + a$.*

Proof. Assume $C \in \mathcal{C}$ is a missing clique where i is the color such that $|Color_i(C) \backslash M| < a$. First of all by the pigeon hole principle we get that $|Color_{max(C)}(C)| \geq \frac{|V(C) \backslash M| - a}{l-1}$, because at most a vertecies of $|V(C) \backslash M|$ have color i, so the rest most be distributed among the $l-1$ other colors. Also since color i only have marked vertecies except for the at most a

unmarked, we get that $|Color_i(C)| \leq |V(C) \cap M| + a$. This gives us.

$$|Color_{max(C)}(C)| - \alpha|Color_{min(C)}(C)| \geq |Color_{max(C)}(C)| - \alpha|Color_i(C)| \quad (1)$$

$$\geq \frac{|V(C)\backslash M| - a}{l-1} - \alpha(|V(C) \cap M| + a) - a \quad (2)$$

In step (1) we are using the fact that $|Color_{max(C)}(C)| \geq |Color_i(C)|$ and in step (2) we use that $|Color_{max(C)}(C)| \geq \frac{|V(C)\backslash M|-a}{l-1}$ and $|Color_i(C)| \leq |V(C) \cap M| + a$.

Rearranging $|Color_{max(C)}(C)| - \alpha|Color_{min(C)}(C)| \geq \frac{|V(C)\backslash M|-a}{l-1} - \alpha(|V(C) \cap M| + a) - a$ gives us the desired inequality. □

We can now give a bound on the number of unmarked vertecies in missing cliques.

Observation 6.5. *There are at most* $a + (6\alpha + 3a\alpha + 3al)lk^2 + (6a\alpha + 4 + 6a)lk$ *unmarked vertecies in missing cliques.*

Proof. Let $\mathcal{C}_m \subseteq \mathcal{C}$ be the set of all missing cliques.

$$\sum_{C \in \mathcal{C}_m} |V(C)\backslash M|$$

$$\leq \sum_{C \in \mathcal{C}_m} (l-1)(\alpha(|V(C) \cap M| + a) + |Color_{max(C)}(C)| - \alpha|Color_{min(C)}(C)| + a) + a \quad (1)$$

$$\leq a(3k^2 + 3k) + l \sum_{C \in \mathcal{C}_m} [\alpha(|V(C) \cap M| + a) + |Color_{max(C)}(C)| - \alpha|Color_{min(C)}(C)| + a] \quad (2)$$

$$\leq a + \alpha l \sum_{C \in \mathcal{C}_m} [|V(C) \cap M| + a] + l \sum_{C \in \mathcal{C}_m} [|Color_{max(C)}(C)| - \alpha|Color_{min(C)}(C)| + a] \quad (3)$$

$$\leq a + \alpha l(6k^2 + a(3k^2 + 6k)) + l(4k + a(3k^2 + 6k)) \quad (4)$$

$$= a + 6\alpha lk^2 + 3a\alpha lk^2 + 6a\alpha lk + 4lk + al3k^2 + al6k$$

$$= a + (6\alpha + 3a\alpha + 3al)lk^2 + (6a\alpha + 4 + 6a)lk$$

Step (1) is due to Observation 6.4, where we simply multiple both sides by $l - 1$ and add a to both sides. In step (2) we bring out the last a from the sum, and multiply it by $3k^2 + 3k$ since $|\mathcal{C}_m| \leq 3k^2 + 3k$ by Observation 6.1. Then we factor out l from the sum. In step (3) we simply split up the sum. And finally in step (4) the first sum is bounded by Observation 6.2 and Observation 6.1, and the seconds sum is bounded by Observation 6.4 and Observation 6.1. □

6.5 Linear cliques

We are now going to look at the case when Reduction rule 6.7 can not be applied due to $|Color_i^*(C)| < \beta k$ for some i. Let $\theta = \alpha\beta + 1$, and denote cliques $C \in \mathcal{C}$ where $|Color_i(C)| \leq \theta k$ for all $i \in [l]$ as *linear cliques*. Our next step is bounding the number of vertecies in such cliques.
Our first observation is that if two colors are more than k deletions away from being balanced ,then the clique can not be balanced. This means that in a *YES*-instance if one color have less than βk vertecies in a clique, then the clique must be linear.

(Rule 6.9) If for any $C \in \mathcal{C}$ we have $|Color_i^*(C)| > \alpha(|Color_j^*(C)|) + k$ for some colors $i, j \in [l]$, then return the trivial *NO*-instance.

Lemma 6.13. *Reduction rule 6.9 is safe and can be applied in time $\mathcal{O}(|V|)$.*

Proof. Assume $|Color_i^*(C)| > \alpha(|Color_j^*(C)|) + k$ for some colors $i, j \in [l]$, and that S^* is a solution to (G, k). Also let $C' = f_{S^*}(C)$. There can be at most k vertecies from $Color_i^*(C)$ in S^* which means that $|Color_i(C')| \geq |Color_i^*(C)| - k > \alpha|Color_j^*(C)| + k - k \geq \alpha|Color_j(C')|$, which means that C' is not balanced. To apply the Reduction rule we simply need to compare to largest and smallest color in each clique which can be done in $\mathcal{O}(|V|)$ time using count sort. □

Observation 6.6. *If for any $C \in \mathcal{C}$ there is a color i where $|Color_i(C)| < k\beta$, then C is a linear clique.*

Proof. Assume that there is some $C \in \mathcal{C}$ where $|Color_i^*(C)| < \beta k$ for some $i \in [l]$ and that there is some color j where $|Color_j^*(C)| > \theta k$. We then get that $|Color_j^*(C)| > \theta k = (\alpha(\beta) + 1)k = \alpha\beta k + k > \alpha|Color_i^*(C)| + k$, which means Reduction rule 6.9 would be applied. □

Since we have l colors, we get a simple bound on the number of vertecies in a linear clique.

Observation 6.7. *Any linear clique $C \in \mathcal{C}$ have $|V(C)| \leq \theta l k$.*

6.6 Connected cliques

We are now gonna look at a sub-class of linear cliques. We give the following definition. We say that a linear clique is *connected* if $N(C - M) \cap S \neq \emptyset$, so the unmarked vertecies of C are connected to S. To bound the number of unmarked vertecies in connected cliques we use the observation that since the clique is linear there is a bounded number of vertecies in the clique. So if some vertex $s \in S$ is connected to a lot of such vertecies then s have a lot of connections to more than one clique.

For some subset $D \subseteq \mathcal{C}$, we define $V(D) = \bigcup_{C \in D} V(C)$.

(Rule 6.10) If for any $s \in S$ we can partition the set $N_{\mathcal{C}}(s)$ into 2 sets C_1 and C_2 such that $|V(C_1)| > k$ and $|V(C_2)| > k$, then return $(G - v, k - 1)$.

We will use an algorithm for the knapsack problem to compute Reduction rule 6.10, which is the following problem.

KNAPSACK WITHOUT REPETITION

Input: A set of n items numbered $1, ..., n$ where each item have value v_i and weight w_i, and capacity W.

Output: A subset of items $I \subseteq [n]$ where $\sum_{i \in I} v_i$ is maximized with the condition that $\sum_{i \in I} w_i \leq W$.

Lemma 6.14. *Reduction rule 6.10 is safe and can be applied in time $\mathcal{O}(|V|^3)$.*

Proof. If any solution S^* does not contain v then for each P_3 on the form xvy where $x \in V(C_1)$ and $y \in V(C_2)$ we must put either $V(C_1)$ or $V(C_2)$ in S^* which is impossible since both of these are greater than k. To apply the Reduction we do the following. Let A be an empty set and for each clique $C \in N_C(s)$ put the number $|V(C)|$ into A. The question then become if we can partition this set where both parts are greater than k. We can view this as a knapsack problem without repetition where our knapsack capacity is $\lceil \frac{\sum_{a \in A} a}{2} \rceil$ where A represent both the weight and value of the items. This problem can be solved in $\mathcal{O}(|S|(k+1)) = \mathcal{O}(|V|^2)$ using a standard dynamic programming algorithm [9]. We can also compute $N_{\mathcal{C}}(s)$ in $\mathcal{O}(|V|)$ time. Doing this for all $s \in S$ we use at most $\mathcal{O}(|V|^3)$ time. □

Observation 6.8. *There can be at most $3\theta lk^2 + 6k^2$ unmarked vertecies in connected cliques.*

Proof. Assume there are more than $3\theta lk^2+6k^2$ unmarked vertecies in connected cliques. Any given linear clique contains at most θlk vertecies by Observation 6.7. Now each unmarked vertex in a connected clique is connected to S since these vertecies form a module by Observation 6.3. So there must be a vertex $v \in S$ which is connected to at least $\frac{3\theta lk^2+6k^2}{3k} = \theta lk+2k$ unmarked vertecies in connected cliques. Now this is a contradiction since we can construct sets $C_1, C_2 \subseteq N_{\mathcal{C}}(s)$ consisting of connected cliques by putting cliques $C \in N_{\mathcal{C}}(s)$ into C_1 until $|V(C_1)| > k$. Then $|V(C_1)| \leq \theta lk + k$ since $|V(C)| \leq \theta lk$ in connected cliques. This means that $|V(C_2)| > \theta lk+2k-(\theta lk+k) = k$, so Reduction rule 6.10 should be applied. □

6.7 Hidden cliques

Next we are going to look at cliques where the number of unmarked vertecies are much greater than the number of marked vertecies. We start by giving a bound on the number of vertecies in cliques where the number of unmarked vertecies are not more than a constant factor bigger than the number of marked vertecies.

Observation 6.9. *There are at most* $(18\alpha^2 3a)lk^2+3akl$ *unmarked vertecies in cliques* $C \in \mathcal{C}$ *where* $|V(C)\backslash M| < 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al$.

Proof. Let $\mathcal{C}' \subseteq \mathcal{C}$ we the cliques C where $|V(C)\backslash M| < 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al$. Then

$$\begin{aligned}
\sum_{C\in\mathcal{C}'} |V(C)\backslash M| &\leq \sum_{C\in\mathcal{C}'} 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al && (1)\\
&= 2\alpha^2 l \sum_{C\in\mathcal{C}'} |V(C) \cap M| + 2\alpha^2 l \sum_{C\in\mathcal{C}'} N(C) + \sum_{C\in\mathcal{C}'} al \\
&\leq 2\alpha^2 l 6k^2 + 2\alpha^2 l \sum_{C\in\mathcal{C}'} N(C) + \sum_{C\in\mathcal{C}'} al && (2)\\
&\leq 2\alpha^2 l 6k^2 + 2\alpha^2 l 3k^2 + \sum_{C\in\mathcal{C}'} al && (3)\\
&= 18\alpha^2 lk^2 + (3k^2 + 3k)al. \\
&= (18\alpha^2 + 3a)lk^2 + 3akl
\end{aligned}$$

Step (1) Is due to our assumption and step (2) is from Observation 6.2. In step (3) we use the fact that any vertex $s \in S$ can be connected to at most k different cliques by Reduction rule 6.5. So s gets counted at most k times in the sum $\sum_{C\in\mathcal{C}} N(C)$. So since $|S| \leq 3k$ the total sum will be at most $3k^2$.

□

Our next next goal is to deal with unmarked vertecies which are isolated from S. To do this we make the following definition. We call a clique $C \in \mathcal{C}$ *hidden* if it satisfies the following conditions.

- (1) $N(C - M) \cap S = \emptyset$,
- (2) C is balanced,
- (3) $|V(C) \backslash M| \geq 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al$.

And we say that a clique $C \in \mathcal{C}$ is *almost hidden* if C satisfies condition 1 and 2, and condition 3 is replaced with the weaker condition $|V(C) \backslash M| \geq 2\alpha^2 l(|V(C) \cap M| + |N(C)|)$. Denote the set of hidden cliques as $\mathcal{C}_I \subseteq \mathcal{C}$ and almost hidden cliques as $\mathcal{C}_A \subseteq \mathcal{C}_I$. We will now try and bound the number of unmarked vertecies in hidden cliques.

The main idea is that there is really no point in deleting a lot unmarked vertecies in almost hidden cliques. This is because there are a lot more unmarked vertecies than marked vertecies in these cliques, so deleting all the unmarked vertecies is a very sub-optimal way to delete any potential P_3 in G.

We first use the fact that almost hidden cliques are balanced, and have a lot more unmarked than marked vertecies, to argue that almost hidden cliques contain a lot of unmarked vertecies with every color.

Lemma 6.15. *For any $C \in \mathcal{C}_A$, $|Color_i(C) \backslash M| \geq \alpha(|V(C) \cap M| + |N(C)|)$ for all $i \in [l]$.*

Proof. Assume that there is some color i where $|Color_i(C) \backslash M| < \alpha(|V(C) \cap M| + |N(C)|)$. This means that $|Color_i(C)| < \alpha(|V(C) \cap M| + |N(C)|) + |V(C) \cap M| \leq 2\alpha(|V(C) \cap M| + |N(C)|)$. There must be some j so that $|Color_i(C) \backslash M| \geq 2\alpha^2(|V(C) \cap M| + |N(C)|)$ for some j since otherwise $|V(C) \backslash M| < 2\alpha^2 l(|V(C) \cap M| + |N(C)|)$, which contradicts condition (3) for an almost hidden clique. Then i and j can not be balanced since

$$\begin{aligned} |Color_j(C)| &\geq 2\alpha^2(|V(C) \cap M| + |N(C)| \\ &> \alpha |Color_i(C)|. \end{aligned}$$

So i and j are not balanced in C which contradicts the fact that C is balanced by condition (2) of almost hidden cliques. □

The next lemma is what allows us to potentially delete unmarked vertecies in almost hidden cliques even if we go below k unmarked vertecies in the clique. There is no reason for a solution to delete all the unmarked vertecies in isolated cliques.

Lemma 6.16. *If (G, k) is a YES-instance, then there is a solution S^* where $(V(C)\backslash M)\backslash S^* \neq \emptyset$, for all $C \in \mathcal{C}_A$.*

Proof. Assume (G, k) is a *YES*-instance with solution S^*, where $(V(C)\backslash M)\backslash S^* = \emptyset$. We know by Lemma 6.15 that $|Color_i(C)\backslash M| \geq \alpha(|V(C) \cap M| + |N(C)|)$ for all $i \in [l]$. So let A be a set with $\alpha(|V(C) \cap M| + |N(C)|)$ vertecies from $Color_i(C)\backslash M$ for each $i \in [l]$. Then let $S' = (S^* \cup f_{S^*}(C))\backslash A$. So we delete the clique $f_{S^*}(C)$ and add a lot of the unmarked vertecies to create a new clique. We know this clique is balanced since all the colors have the same number of vertecies. And we know we are not creating any P_3 since all the vertecies are isolated so they are only connected to each other and potentially some of $f_{S^*}(C)$, but we deleted these vertecies (added them to S'). Finally we need to show that $|S'| \leq |S^*|$. We add $|f_{S^*}(C)|$ vertecies and delete $|V(C)\backslash M|$ vertecies from S^* to create S'. We know that $|f^*_S(C)| \leq |V(C) \cap M| + |N(C)|$ since $(V(C)\backslash M)\backslash S^* = \emptyset$ (So the only vertecies in $f_{S^*}(C)$ are either marked or in the neighborhood of C)), and we know that $|V(C)\backslash M| \geq 2\alpha^2 l(|V(C) \cap M| + |N(C)|)$ since C is an hidden clique. So we clearly delete more vertecies than we add to S'. We can repeat this process for all $C \in \mathcal{C}_A$. □

Not only is there no point in deleting all the unmarked vertecies in a almost hidden clique, but if we have a solution S^* we can construct a solution S', where the number of unmarked vertecies in S' with each color are upper bounded.

For a vertex set N let $min(N)$ be a color with the fewest vertecies in N.

Lemma 6.17. *If (G, k) is a YES-instance, then for any $C \in \mathcal{C}_A$ there is a solution S', where $|S' \cap (Color_i(C)\backslash M)| \leq \alpha|V(C) \cap M| + 1$ for all $i \in [l]$ and $(V(C)\backslash M)\backslash S' \neq \emptyset$.*

Proof. Assume (G, k) is a *YES*-instance with solution S^*. We then construct a new solution S' by first letting S' be the solution from Lemma 6.16. We then apply the following procedure exhaustively.
Let $C'' = f_{S'}(C)$, $V' = V(C'')$ and $i = min(V')$. Now remove vertecies v from S' where $v \in S' \cap (Color_i(C)\backslash M)$ (and update the definition of C'' until we run out of such vertecies or color i will becomes unbalanced in C'' (we don't remove v if i will become unbalanced). If we run out of such vertecies we also declare i marked and let $V' = V'\backslash Color_i(C'')$. We then set $i = min(V')$ again and continue this process until all colors are either marked or will become unbalanced in C'' if any more vertecies are added to C'' (removed from S'). After this, if all colors have the same number of vertecies in C'' we remove one unmarked vertex from each color from S' until at least one color have no unmarked vertecies in S'.

First we observe that S' is still a solution since by the construction of S', C'' will still be balanced. Also all the vertecies we added to C'' are unmarked so have a twin by Lemma 6.16.

For the other part let i be any marked color. In that case there are no vertecies in $S' \cap (Color_i(C) \backslash M)$ which means $|S' \cap Color_i(C)| \leq |V(C) \cap M|$, which is what we wanted to prove. Now let i be an unmarked color. That means removing any vertex in $S' \cap (Color_i(C) \backslash M)$ from S' makes i unbalanced in C'', which must mean that $i = max(C'')$. There are then two cases to consider for $min(C'')$. If $min(C'')$ is marked then $|S' \cap (Color_{min(C'')}(C)| \leq |V(C) \cap M|$. For the case where $min(C'')$ is unmarked the only possibility is that $|Color_{min(C'')}(C'')| = |Color_{max(C'')}(C'')|$ since otherwise we could remove an vertex from $S' \cap Color_{min(C'')}(C)$. This means all colors have the same number of vertecies in C''. In that case one color j must have no unmarked vertecies in S' since we delete one unmarked vertex of each color until this is the case. This means that $|Color_j(C'')| \leq |V(C) \cap M|$ (This color $j = min(C'')$ since all the colors have the same number of vertecies). So both the case when $min(C'')$ is marked and unmarked we have that $|S' \cap Color_{min(C'')}(C)| \leq |V(C) \cap M|$. Then to arrive at a contradiction assume $|S' \cap Color_i(C)| > \alpha |V(C)| \cap M| + 1$.

$$
\begin{aligned}
\alpha |Color_{min(C'')}(C'')| &\geq \alpha(|Color_{min(C'')}(C)| - |V(C) \cap M|) && (1)\\
&= \alpha |Color_{min(C'')}(C)| - \alpha |V(C) \cap M| && (2)\\
&\geq |Color_i(C)| - \alpha |V(C) \cap M| && (3)\\
&\geq |Color_i(C'')| + \alpha(|V(C) \cap M|) + 1 - \alpha |V(C) \cap M| && (4)\\
&= |Color_i(C'')| + 1
\end{aligned}
$$

Step (1) is due to the fact that $|S' \cap (Color_{min(C'')}(C)) \leq |V(C) \cap M|$. In step (2) we simply simply multiply out the α. Then we use the fact that C is balanced in (3) and finally we use the assumption that $|S' \cap Color_i(C)| > \alpha |V(C)| \cap M| + 1$ in step (4) which means that at at least $\alpha |V(C) \cap M| + 1$ vertecies are deleted with color i when comparing C to C''.

The end result is that it is possible to increase $Color_i(C'')$ by one and i will still be balanced in C''. Which contradicts the stopping condition, and the fact that i is unmarked. Finally the fact that $(V(C) \backslash M) \backslash S' \neq \emptyset$ follows from Lemma 6.16, and the fact that we do not add any vertecies to S' after this initial construction.

□

Now we will give 2 reduction rules analogous to reduction rules 6.6 and 6.7 but now only

concerning hidden cliques. Using Lemma 6.17 we have a bound of how many vertecies of a given color a solution will contain, so we do not have to worry about bringing the number of marked vertecies below k.

(Rule 6.11) If any $C \in \mathcal{C}_I$ satisfies the following conditions

- $d_C(max(C), min(C)) > 2\alpha|V(C) \cap M| + 2a + 2$
- $|Color_i(C) \backslash M| > a$ for all $i \in [l]$.

Then let the set $X \subseteq V(C) \backslash M$ be constructed the following way. For each color $i \in [l]$

- If $d_C(i, max(C)) \leq \alpha|V(C) \cap M| + a + 1$ then put a vertecies from $Color_i(C) \backslash M$
- If $d_C(i, min(C)) \leq \alpha|V(C) \cap M| + a + 1$ then put b vertecies from $Color_i(C) \backslash M$ in X

Finally return $(G - X, k)$.

In the following proof we will use Lemma 6.17. We will abuse notation when we use this Lemma on a clique $C_R = C - X$ where $C \in \mathcal{C}_I$, since C_R is not really in $\mathcal{C}$. But this makes no difference as C still satisfies all the requirements of being almost hidden which is something we will argue in the proof.

Lemma 6.18. *Reduction rule 6.11 is safe and can be applied in time* $\mathcal{O}(|V| + |E| \log |V|)$.

Proof. Firstly it should be clear that no color i can satisfy both $d_C(i, max(C)) \leq \alpha|V(C) \cap M'| + a + 1$ and $d_C(i, min(C)) \leq \alpha|V(C) \cap M'| + a + 1$ since that would imply that $d_C(max(C), min(C)) \leq d_C(max(C), i) + d_C(i, min(C)) \leq 2\alpha|V(C) \cap M'| + 2a + 2$.

Let $G' = G - X$. First assume S^* is a solution to (G, k) and start by applying Lemma 6.17 to S^* and C. We first want to modify S^*, such that $S^* \cap X = \emptyset$.

Let $C' = f_{S^*}(C)$ and the following. For each $v \in S^* \cap X$, remove v from S^* and replace it with some vertex in $Color_i(C) \backslash (A \cup S^*)$. Start by assuming there are enough vertecies in $Color_i(C) \backslash (A \cup S^*)$ for all colors i to do this replacement for all $v \in S^* \cap X$, and call this new solution S'. To see that S' is a solution we clearly did not change any color ratios. Let U' be the set of vertecies removed from S^* when we created S', and let $U = V(C) \backslash M$. We see that $U \backslash U' \neq \emptyset$ since by Lemma 6.17 there is some marked vertex of C not in S^*, so it also not in U'. We also observe that $(G - S') - U' \subseteq G - S^*$ which means $(G - S') - U'$ does not have any P_3. Then Lemma 3.3 tells us that $G - S'$ also does not have P_3.

If there are not enough vertecies in $Color_i(C) \backslash (A \cup S^*)$ for some color i to do this replacement, then $|Color_i(C')| \leq 2a$, since otherwise there must be at least a vertecies in

$Color_i(C')\backslash A$, so the replacement would be possible. Then we let $S' = S^*\backslash(Color_i(C) \cup A)$ (For all colors i where the replacement was not possible) and want to show that S' is still a solution. Let $C'' = f_{S'}(C)$. For any P_3 let $U = V(C)\backslash M$ and $U' = S^*\backslash S'$. Then $U\backslash U' \neq \emptyset$ and since $(G - S') - U' \subseteq G - S^*$, $(G - S') - U'$ does not have a P_3 which means that neither does $G - S'$ by Lemma 3.3.

For the colors we increased the size of color i in C'' compared to C' so we only need to verify that $|Color_i(C'')| \leq |Color_{max(C')}(C')|$ to conclude that C'' is balanced, since we know that C' is balanced. This is clearly true since the fact $d_C(max(C), min(C)) > 2\alpha|V(C) \cap M| + 2a + 2$ means that $|Color_{max(C')}(C')| \geq 2a$, since there are at most $\alpha|V(C) \cap M| + 1$ vertecies with color $max(C)$ in S^* by Lemma 6.17. Then finally let $S^* = S'$. We note that S^* still satisfies the specification of Lemma 6.17 since we have not done anything but potentially remove unmarked vertecies from S^*.

We now want to argue that S^* is a solution to (G', k) as well. Since $G' - S^* \subset G - S^*$, $G' - S^*$ can clearly not contain a P_3, since S^* is a solution to (G, k). For the color ratios let $C' = f_{S^*}(C)$, $C_R = C - X$ and $C'_R = f^{G'}_{S^*}(C_R)$. Then by Lemma 6.8 we only need to check that $max(C'_R)$ and $min(C'_R)$ are balanced in C'_R. Since $|S^* \cap Color_i(C)| \leq \alpha|V(C) \cap M'| + 1$ for all i only colors where $d_C(i, max(C)) \leq \alpha|V(C) \cap M'| + 1 + a$ can become $max(C'_R)$ since there are at most $\alpha|V(C) \cap M'| + 1 + a$ vertecies deleted from $max(C'_R)$ when comparing C to C'_R (S^* deletes at most $\alpha|V(C) \cap M'| + 1$ vertecies and X delete at most a vertecies). For similar reasons only colors i where $d_C(i, min(C)) \leq \alpha|V(C) \cap M'| + 1 + a$ can become $min(C'_R)$, since at most $\alpha|V(C) \cap M'| + 1 + a$ vertecies from any specific color gets deleted when comparing C to C'_R. This means that there are a vertecies with color $max(C'_R)$ and b vertecies with color $min(C'_R)$ inn X. Let i and j be $max(C'_R)$ and $min(C'_R)$ respectively. We then want to apply Lemma 6.7. We have establishment that $|Color_i(C')| = |Color_i(C'_R)| + a$ and $|Color_j(C')| = |Color_j(C'_R)| + b$ since $S^* \cap X = \emptyset$. We also have $|Color_i(C')| \geq |Color_j(C')| + a$ since $|S^* \cap Color_i(C)| \leq \alpha|V(C) \cap M'| + 1$ and $d_C(max(C), min(C)) > 2\alpha|V(C) \cap M| + 2a + 2$. So we can apply Lemma 6.7, and conclude that i and j are balanced in C'_R as well.

Now assume S^* is a solution to (G', k) and let $C_R = C - X$. Start by applying Lemma 6.17 to S^* and C_R. The conditions are satisfied (even though C_R is not really in $\mathcal{C}$) because of the fact that C is hidden, which means that $|V(C)\backslash M| \geq 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al$ and since $|X| \leq al$ we get that $|V(C_R)\backslash M| \geq 2\alpha^2 l(|V(C) \cap M| + |N(C)|)$, so C_R is almost hidden. To show that there are no P_3 in $G - S^*$ we apply Lemma 3.3 to $G - S^*$ with $U = V(C)\backslash M$ and $U' = U\backslash S^*$. Then $U\backslash U' \neq \emptyset$ by Lemma 6.17 since there is at least one unmarked vertex in C which is not in S^*. So by Lemma 3.3 $G - S^*$ does not have a P_3 since

$(G - S^*) - U'$ does not.

Now let $C' = f_{S^*}(C)$ and $C'_R = f^{G'}_{S^*}(C_R)$. We need to check that the colors of C' are balanced but we again only have to check that $max(C')$ and $min(C')$ are balanced by Lemma 6.8. We apply a similar argument as above but the only difference is that now at most $|V(C) \cap M'| + 1$ vertecies can be deleted from any color when comparing C with C'. So let i and j be $max(C')$ and $min(C')$ respectively. If we add a vertecies from $Color_i(C)$ and b vertecies from $Color_j(C)$ to C'_R we get C' with respect to these two colors. Also we see that $|Color_i(C'_R)| \geq |Color_j(C'_R)|$ since $|S' \cap Color_i(C)| \leq \alpha|V(C) \cap M'| + 1$. So we can apply Lemma 6.7 which gives us that i and j are balanced in C' as well.
To apply the algorithm it should be clear that all the requirements can be checked in $\mathcal{O}(|V|)$ time using count sort for the distance. Then deleting the vertecies is done in time $\mathcal{O}(|V| + |E| \log |V|)$ in an adjacency matrix.

□

The next rule compliments Reduction rule 6.11, and is for the case when the maximum color distance is small.

(Rule 6.12) If $\alpha > 1$ and for some $C \in \mathcal{C}_I$ the following conditions are satisfied

- $d_C(max(C), min(C)) \leq 2\alpha|V(C) \cap M'| + 2a + 3$,
- $|Color_i(C)| \geq \frac{(|V(C) \cap M| + 2a + 3)a}{a-b} + \alpha|V(C) \cap M| + 2$, for all $i \in [l]$

then let v be any vertex in $V(C) \backslash M$ and return $(G - \{v\}, k)$.

Lemma 6.19. *Reduction rule 6.12 is safe and can be applied in time $\mathcal{O}(|V| + |E|)$.*

Proof. First assume (G, k) is a *YES*-instance with solution S^*. We then start by applying Lemma 6.17 on S^* to get a new solution S^*. Let $C' = f_{S^*}(C)$, $C_R = C - \{v\}$, $C'_R = f^{G'}_{S^*}(C_R)$ and $c = \frac{(3\alpha|V(C) \cap M'| + 2a + 4)a}{a-b}$. Then by Lemma 6.10 if $d_{C'_R}(max(C'_R), min(C'_R)) \leq \frac{c(a-b)}{a} = 3\alpha|V(C) \cap M| + 2a + 4$ and $|Color_i(C')| \geq c$ then C'_R is balanced. We know by Lemma 6.17 that $|S' \cap Color_i(C)| \leq \alpha|V(C) \cap M'| + 1$ for all i. Which means that $|Color_i(C'_R)| \geq |Color_i(C)| - (\alpha|V(C) \cap M'| + 1) \geq c$, so C'_R must be balanced. There is no P_3 in $G' - S^*$ since $G' - S^* \subset G - S^*$.

Now assume (G', k) is a *YES*-instance with solution S^*. Again apply Lemma 6.17 to get a new solution S^*. Then let $C' = f_{S^*}(C)$, $C_R = C - \{v\}$ and $C'_R = f^{G'}_{S^*}(C_R)$. We now want to apply Lemma 6.10 to C' which we can since $d_{C'}(max(C'), min(C')) \leq 3\alpha|V(C) \cap M| + 2a + 4$

by Lemma 6.17. So the argument is the same as above. For any potential P_3 in $G - S^*$ we observe that v has a twin by Lemma 6.17. □

We now want to bound the number of unmarked vertecies in cliques where the second requirement of Reduction rule 6.12 isn't satisfied (One color does not have enough unmarked vertecies).

Observation 6.10. *If for any $C \in \mathcal{C}_I$ $|Color_i(C)| < \frac{(|V(C)\cap M|+2a+3)a}{a-b} + \alpha|V(C)\cap M| + 2$ for some i, then $|V(C)| \leq \alpha l \frac{(|V(C)\cap M'|+2a+3)a}{a-b} + \alpha^2 l|V(C)\cap M| + 2\alpha l$ for all j.*

Proof. If $|Color_i(C)| < \frac{(|V(C)\cap M|+2a+3)a}{a-b} + \alpha|V(C)\cap M| + 2$. Then since C is balanced $|Color_j(C)| \leq \alpha|Color_i(C)| < \alpha\frac{(|V(C)\cap M'|+2a+3)a}{a-b} + \alpha^2|V(C)\cap M| + 2\alpha$. So since we have l colors, $|V(C)| \leq \alpha l \frac{(|V(C)\cap M|+2a+3)a}{a-b} + \alpha^2 l|V(C)\cap M| + 2\alpha l$ □

Observation 6.11. *There are at most $\frac{6\alpha a^2}{a-b}lk^2 + (\frac{6a^2+9a}{a-b} + 6 + 6\alpha)k^2 + (\frac{6a^2+9a}{a-b} + 6)k$ vertecies in cliques $C \in \mathcal{C}_I$, where $|Color_i(C)| < \frac{(|V(C)\cap M|+2a+3)a}{a-b} + \alpha|V(C)\cap M'| + 2$ for some i.*

Proof. If $|Color_i(C)| < \frac{(|V(C)\cap M'|+2a+3)a}{a-b} + \alpha|V(C)\cap M'| + 2$ for some i, then by Observation 6.10 $|V(C)| \leq \alpha l \frac{(|V(C)\cap M|+2a+3)a}{a-b} + \alpha^2 l|V(C)\cap M'| + 2\alpha l$. Then if we let $\mathcal{C}' \subseteq \mathcal{C}_I$ be all such cliques we get.

$$\begin{aligned}
\sum_{C\in\mathcal{C}'} |V(C)| &\leq \sum_{C\in\mathcal{C}'} \alpha l \frac{(|V(C)\cap M| + 2a + 3)a}{a-b} + \alpha^2 l|V(C)\cap M'| + 2\alpha l && (1)\\
&= \alpha l \sum_{C\in\mathcal{C}'} \frac{(|V(C)\cap M| + 2a + 3)a}{a-b} + \alpha|V(C)\cap M'| + 2 && (2)\\
&\leq \alpha l \frac{a}{a-b} \sum_{C\in\mathcal{C}'} a|V(C)\cap M| + \sum_{C\in\mathcal{C}'} \frac{2a^2+3a}{a-b} + 2 + \alpha \sum_{C\in\mathcal{C}'} |V(C)\cap M| \\
&\leq \frac{\alpha a^2 l 6k^2}{a-b} + (3k^2+3k)(\frac{2a^2+3a}{a-b} + 2) + \alpha 6k^2 && (3)\\
&= \frac{6\alpha a^2}{a-b}lk^2 + (\frac{6a^2+9a}{a-b} + 6 + 6\alpha)k^2 + (\frac{6a^2+9a}{a-b} + 6)k
\end{aligned}$$

Step (1) is using Observation 6.10 and step (2) we just factor out αl. Step 3 is using Observations 6.1 and 6.2.

□

Lemma 6.20. *Reduction rule 6.13 is safe and can be applied in time $\mathcal{O}(|V|)$.*

Proof. Assume that there are more than $4k$ unbalanced cliques in $\mathcal{C}$, and that S^* is a solution to (G, k). There must be at least one of these unbalanced cliques $C \in \mathcal{C}$ where $f_{S^*}(C) = C$. This is because there at at most $3k$ vertecies in S and k vertecies in S^*, so at most $4k$ cliques can be changed in some way. So this clique C is not balanced, a contradiction. To apply the Reduction rule we need to go through all the cliques $C \in \mathcal{C}$ and check whether or not C is balanced , which can be done in $\mathcal{O}(|V(C)|)$ time using count sort. □

We now have all we need to bound the number of unmarked vertecies in hidden cliques.

Observation 6.12. *There are $\mathcal{O}(k^2 l)$ unmarked vertecies in hidden cliques when $\alpha > 1$.*

Proof. We look what happens when Reduction rules 6.11 and 6.12 can not be applied. These 2 rules compliment each other. First we deal with the cliques $C \in \mathcal{C}_I$ where $d_C(max(C), min(C)) > 2\alpha|V(C) \cap M| + 2a + 2$. Then $|Color_i(C) \backslash M| \leq a$ for some i if the Reduction 6.11 can not be applied. So Observation 6.5 gives us a $\mathcal{O}(k^2 l)$ bound on the number of unmarked vertecies in such missing cliques.

Then in the case when $d_C(max(C), min(C)) \leq 2\alpha|V(C) \cap M| + 2a + 2$, if Reduction rule 6.12 can not applied to some $C \in \mathcal{C}_I$, then $|Color_i(C)| < \frac{(|V(C) \cap M| + 2a + 3)a}{a-b} + \alpha|V(C) \cap M| + 2$ for some $i \in [l]$ and there are at most $\mathcal{O}(k^2 l)$ unmarked vertecies in such cliques by Observation 6.11. □

Observation 6.13. *There are at most $O(k^2 l)$ unmarked vertecies in linear cliques.*

Proof. Connected cliques are bounded by Observation 6.8. For cliques which are not connected then condition (2) is satisfied for hidden clique.s If condition 2 and 3 also are satisfied then the number of vertecies are bounded by Observation 6.12 since the clique would be hidden. If condition 2 is not satisfied then there are at most $4k$ unbalanced cliques by Reduction 6.13, so there are at most $\theta lk4k = O(k^2 l)$vertecies in such cliques. If condition (3) is not satisfied then the number of vertecies in such cliques are bounded by Observation 6.9. □

The final observation we need is to limit the number of balanced cliques which have few unmarked vertecies and are not linear.

Observation 6.14. *There are at most k balanced cliques where $|V(C) \backslash M| < k + al$ and C is not linear.*

Proof. Assume there are more than k such cliques and denote them as $\mathcal{C}' \subseteq \mathcal{C}$. Then

$$\begin{aligned}
\sum_{C \in \mathcal{C}'} |V(C) \cap M| &= \sum_{\mathcal{C}'} |V(C)| - |V(C) \backslash M| \\
&> \sum_{C \in \mathcal{C}'} \theta lk - (k + al) && (1) \\
&\geq \sum_{C \in \mathcal{C}'} 5alk - k - al && (2) \\
&\geq \sum_{C \in \mathcal{C}'} 3alk \\
&\geq 3alk^2 && (3) \\
&\geq 6ak^2 && (4) \\
&\geq 6k^2
\end{aligned}$$

In step (1) we use our assumption about that the cliques in $\mathcal{C}'$ are not linear and that they have less than $k + al$ unmarked vertecies. In step (2) we simply us the definition of θ and β. In (3) we use the assumption that $|\mathcal{C}'| \geq k$ and in (4) we know that $l \geq 2$.
So we see that there are to many marked vertecies in these cliques which contradicts Observation 6.2. □

(Rule 6.13) If there are more than $4k$ unbalanced cliques in $\mathcal{C}$, then return the trivial *NO*-instance.

6.8 Proof of Theorem 2

We now prove the part of Theorem 2 when $\alpha > 1$, we prove the part where $\alpha = 1$ in chapter 8.

Theorem 2. *There is a kernel with $\mathcal{O}(k^2 l)$ vertecies for $\alpha-$BALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V| + |E|))$.*

Proof. Assume $\alpha > 1$. We first split the cliques $C \in \mathcal{C}$ up into 2 cases. The first is where $d^*_C(min^*(C), max^*(C)) > 2k + 2a$. Then one of the following conditions must be true if Reduction rule 6.6 can not be applied.

- $|Color_i(C) \backslash M| \leq a$. Then C is a missing cliques which means the number of unmarked vertecies are bounded by Observation 6.5.

- $|V(C) \backslash M| < k + al$. Then if C is linear, the number of unmarked vertecies are bounded by Observation 6.13. If C is not linear then there are at most $5k$ such cliques(k balanced and $4k$ unbalanced) by Observation 6.14 and Reduction rule 6.13. So there are at most $5k(k + al) = \mathcal{O}(k^2 l)$ vertecies in such cliques.

Then the other case is when $d^*_C(min^*(C), max^*(C)) \leq 2k + 2a$. Then if Reduction rule 6.7 can not be applied then one of the following conditions must be true

- $|Color^*_i(C)| < \beta k$, then C is a linear clique so bounded by Observation 6.13.
- $|V(C) \backslash M| \leq k + 1$, then if C is not linear the number of vertecies are bounded by Observation 6.14, and if C is linear the number of vertecies are bounded by Observation 6.13.

Each time we apply a reduction rule we delete at least one vertex. Applying algorithm 6.2 is the bottleneck (slowest step, which dominates all the other steps in terms of asymptotic runtime) for each such application. □

Chapter 7

$\mathcal{O}(k^4)$ kernel for $\alpha-BCVD$

We are now going to give a kernel not dependent on l. We do this by presenting a reduction rule to delete colors from the graph. To do this the algorithm first picks a color j which is to be deleted. j is chosen such that S contain no vertex with color j. The algorithm then goes through all the cliques in $C \in \mathcal{C}$ and finds a color i which vertecies can be removed in C. This is possible since we find a color i which have no marked vertecies, and where i are in the "middle" if we sort the colors by the number of vertecies in C. These vertecies then gets deleted and the vertecies of color j gets recolored to color i. The result is that there are no vertecies left with color j in G.

Algorithm 7.1 Delete Color

1: $j \leftarrow$ any color such that $Color_i(S) = \emptyset$
2: **for** $C \in \mathcal{C}$ **do**
3: $\quad L \leftarrow$ List of all colors i such that $Color_i(S) = \emptyset$ and $Color_i(C) \cap M = \emptyset$.
4: $\quad$ Sort L increasingly by the size of $Color_i(C)$ for all $i \in L$.
5: $\quad i \leftarrow L[1]$.
6: $\quad G \leftarrow G - Color_i(C)$.
7: $\quad$ **for** $v \in Color_j(C)$ **do**
8: $\quad\quad c(v) = i$
9: **return** G

(Rule 7.1) If $l > 6k^2 + 4k + 2$, then apply Algorithm 7.1 to get G', and return (G', k).

Lemma 7.1. *Reduction rule 7.1 is safe and can be applied in time* $\mathcal{O}(|V|^2 + |E| \log |V|)$.

Proof. We first observe that for each iteration in the loop in algorithm 7.1, $|L| > k + 2$. This

is because $l > 6k^2 + 10k + 2$, $|M| \leq 6k^2$ (By Observation 6.2) and $|S| \leq 3k$. So we get $|L| > 6k^2 + 4k + 2 - (6k^2 + 3k) = k + 2$. Now let G' be the graph returned by Algorithm 7.1.

First assume that S^* is a solution to (G, k). Then clearly $G' - S^*$ does not have any P_3 since $G - S^*$ does not. Then for each $C \in \mathcal{C}$ let $C' = f_{S^*}(C)$, C_R is the clique C after applying Algorithm 7.1 and $C'_R = f^{G'}_{S^*}(C_R)$. Let i, j be the variables chosen at line 4 and 1 in Algorithm 7.1 respectively. For the color ratios of C'_R we first see that $|Color_i(C'_R)| = |Color_j(C')|$ and that there is no color j in G'. for any other color $k \in [l] \backslash \{i, j\}$ we see that $|Color_k(C')| = |Color_k(C'_R)|$. So since C' is balanced, so is C'_R, since all the colors of C'_R have a corresponding color in C' with the same number of vertecies.

Now assume S^* is a solution to (G', k). We claim S^* is a solution to (G, k) as well. For each $C \in \mathcal{C}$, let $C' = f_{S^*}(C)$, C_R be C after applying Algorithm 7.1 and $C'_R = f^{G'}_{S^*}(C_R)$. Let j, i be the colors chosen at line 1 and 4 respectively in Algorithm 7.1. First for color i in C', there is some color $p = L[0]$ which means that $|Color_p(C)| \leq |Color_i(C)|$. We know that $S^* \cap Color_i(C) = \emptyset$ since the vertecies from $Color_i(C)$ are not in G', and we know that $S \cap Color_p(C) = \emptyset$, we get that $|Color_p(C')| \leq |Color_i(C')|$. Also because $|L| > k + 2$ there must be at least one color $h \in L, h \neq p$ such that $S^* \cap Color_h(C) = \emptyset$, which mean $|Color_h(C')| \geq |Color_i(C')|$, since $|Color_h(C)| \geq |Color_i(C)|$ and $Color_i(C) \cap S = \emptyset$. So in total we have $|Color_r(C')| \leq |Color_i(C')| \leq |Color_h(C')|$, which means that color i is balanced in C' as long as r and h are balanced. For the color j we know that $|Color_j(C')| = |Color_i(C'_R)|$ and for any other color q not i and j we have that $|Color_q(C')| = |Color_q(C'_R)|$, so C' must be balanced as well.

To apply the Reduction rule we need to run Algorithm 7.1. We first need to compute L which we can do in $\mathcal{O}(|V|)$ time by going through all the vertecies in S and $V(C) \cap M$ and deleting all the colors we find from L. Then we can sort L in $\mathcal{O}(V(C))$ time by using count sort. The deletion is then done in $\mathcal{O}(|V| + |E| \log |V|)$ time for all the cliques in total. And the recoloring is done in $\mathcal{O}(|V|)$ time in total. □

We can now prove theorem 3.

Theorem 3. *When $\alpha > 1$ there is a kernel with $\mathcal{O}(k^4)$ vertecies for $\alpha-$BALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V| + |E|))$.*

Proof. $l \leq 6k^2 + 4k + 2$ by Reduction 7.1 so combining this with theorem 2 we get a $\mathcal{O}(k^4)$ vertex kernel. □

Chapter 8

Kernel for $\alpha-BCVD$ when $\alpha = 1$

We deal with the special case when $\alpha = 1$. In this case we are able to find a better kernel than in the general case because we are able to give a better bound on the number of colors. Assume that $\alpha = 1$ in this section. We first want to obtain a $\mathcal{O}(k^2 l)$ kernel for the case when $\alpha = 1$.

8.1 $\mathcal{O}(k^2 l)$ Kernel

(Rule 8.1) If for some $C \in \mathcal{C}$ we have

- $|Color_i(C) \backslash M| > 0$ for all $i \in [l]$,
- $|V(C) \backslash M| > k + l$,

then let $X \subseteq V(C) \backslash M$ contain one vertex of each color, and return $(G - X, k)$.

Lemma 8.1. *Reduction rule 8.1 is safe and can be applied in time* $\mathcal{O}(|V| + |E| \log |V|)$.

Proof. Let $G' = G - X$. First assume (G, k) is a *YES*-instance with solution S^*. Then for any $C \in \mathcal{C}$, let $C' = f_{S^*}(C)$. We first need to modify S^* such that $S^* \cap X = \emptyset$. We do the following. For each $v \in X \cap S^*$, where $c(v) = i$, delete v from S^* and replace it with any vertex in $Color_i(C')$. Denote this new solution by S', and let $C'' = f_{S^*}(C)$. We did not change the color ratios of C'' compared with C'. Then let $U = V(C) \backslash M$ and $U' = S^* \backslash S'$. We see that $U \backslash U' \neq \emptyset$. We have that $(G - S') - U' \subseteq G - S^*$ which means $(G - S') - U'$ does not have a P_3 which means that $G - S'$ does not have a P_3 by Lemma 3.3.

Now let $S^* = S'$ and let $C' = f_{S^*}(C), C_R = C - X$ and $C'_R = f^{G'}_{S^*}(C_R)$.

Each color have one less vertex in C'_R than in C'. So the colors will be balanced in C'_R as well. Also $G' - S^*$ will clearly not have a P_3 since $G - S^*$ does not.

Now assume (G', k) is a *YES*-instance with solution S^*. Then for any $C \in \mathcal{C}$ let $C' = f_{S^*}(C), C_R = C - X$ and $C'_R = f_{S^*}(C_R)$. Each color in C' has one more vertex than in C'_R, so all the colors will be balanced in C' as well. For any potential P_3 we apply Lemma 3.4 with $U = (V(C) \backslash M$ and $U' = X$, then since $G' = (G - U')$, $G - S^*$ does not have a P_3 since $G' - S^*$ does not.
We can check the condition for all cliques in $O(|V|)$ time and deleting vertecies is done in $O(|E| \log |V|)$ time. □

We will use a similar rule but this one is only for hidden cliques.

(Rule 8.2) If for any $C \in \mathcal{C}_I$, $|Color_i(C) \backslash M| > 0$ for all $i \in [l]$ then let $X \subseteq V(C) \backslash M$ contain one vertex of each color and return $(G - X, k)$.

Lemma 8.2. *Reduction rule 8.2 is safe and can be applied in time $O(|V| + |E| \log |V|)$.*

Proof. Let C be the clique the Reduction was applied to.

First assume (G, k) is a *YES*-instance with solution S^*. Start with applying Lemma 6.17 to S^* and C. We first construct a solution S^* where $S^* \cap X = \emptyset$. Simply replace all vertecies in $S^* \cap A$ with vertecies in C' with the same color. This must be possible since otherwise C' is missing a color, so is not balanced. The color ratios stay the same and since $(V(C) \backslash M) \backslash S^*$ by Lemma 6.17 all the vertecies in X have a twin in C', so no new P_3 is created by Lemma 3.2. Let $C_R = C - X$. S^* is a solution to (G', k) as well since $f_{S^*}(C_R)$ have one less of each color, and clearly $G' - S^*$ also does not contain a P_3.
Now assume (G', k) is a *YES*-instance. We apply Lemma 6.17 to get a solution S^*. S^* is a solution to (G, k) as well since for any P_3 we let $U = V(C) \backslash M$ and $U' = X$. Then the conditions of Lemma 3.4 are satisfied which means that $G - S^*$ does not have a P_3 since $(G - S^*) - U'$ does not.Also all the vertecies in X (one of each color) will be in the same clique so that clique will still be balanced. □

We can now finish the proof for Theorem 2 for the case when $\alpha = 1$.

Theorem 2. *There is a kernel with $\mathcal{O}(k^2 l)$ vertecies for $\alpha-$BALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V| + |E|))$.*

Proof. Assume $\alpha = 1$. If for some $C \in \mathcal{C}$ Reduction rule 8.2 can not be applied then C is either a missing clique so bounded by Observation 6.5 or C is not hidden, which means that one of the following condition must be true.

- $N(C - M) \cap S \neq \emptyset$ Then C is a connected clique so bounded by Observation 6.8.
- C not balanced. Then there are at most $4k$ such cliques by Reduction 6.13. So if Reduction rule 8.1 can not be applied and C is not a missing clique then $|V(C) \backslash M| \leq k + l$. So there are at most $4k(k + l) = \mathcal{O}(k^2 l)$ unmarked vertecies in such cliques. If C is a missing clique the number of vertecies are bounded by Observation 6.5.
- $|V(C) \backslash M| < 2\alpha^2 l(|V(C) \cap M| + |N(C)|) + al$ then the number of unmarked vertecies are bounded by Observation 6.9.

□

8.2 $\mathcal{O}(k^3)$ Kernel

Our next goal is to give a linear bound on the number of colors. Our strategy here is to argue that if there are a lot of colors (more than 5k), then most of them have to have the exact same number of vertecies and we can not delete any vertecies with these colors since that would mean we would have to delete vertecies from all the other colors as well, which becomes to many deletions. This allows us to delete the vertecies of one of these colors, since they must form a module.

For any clique $C \in \mathcal{C}$ let $[i]_C$ be the set of colors j such that $|Color_j(C)| = i$, and let $[i]^*_C$ be the set of colors j such that $|Color^*_j(C)| = i$. Let $[max]^*_C = [a]^*_C$ be the largest sets of all $[i]^*_C$.

If one color is completely missing from the the some clique C and it's neighborhood then there is no way to balance C.

(Rule 8.3) If for any clique $C \in \mathcal{C}$, $|Color^*_i(C)| = 0$ then return $(G - V(C), k - |C|)$

Lemma 8.3. *Reduction rule 8.3 is safe and can be applied in time $O(|V| + |E|)$.*

Proof. Clearly if a color is missing from the closed neighborhood of a clique then there is no way to balanced C, so we must delete it. Calculating $N[C]$ for all $C \in \mathcal{C}$ in total can be done in $O(|V| + |E|)$ since we already know the vertecies in $V(C)$ and including the vertecies in S for each $v \in S$ we check all it's edges and add it to the respective neighborhoods. In total we add these vertecies to at most $|E|$ neighborhoods so this is done in $O(|V| + ||E|)$. □

(Rule 8.4) If $l > 5k$ and for some $C \in \mathcal{C}$, $|[max]^*_C| \leq 4k$, then return the trivial *NO*-instance.

Lemma 8.4. *Reduction rule 8.4 is safe and can be applied in time $\mathcal{O}(|V|)$.*

Proof. For any solution S^*, let $C' = f_{S^*}(C)$. First of all $V(C') \neq \emptyset$ because $l > 5k$ and Reduction rule 8.3. C' has to have to have the same number of verteceies with each color. Assume this number is a. We know that $|[a]^*_C| \leq 4k$. which means there are more than k colors i such that $|Color^*_i(C)| \neq a$. Since $|S^*| \leq k$ there is at least one of these colors j such that $Color^*_j(C) \cap S^* = \emptyset$ which contradicts our choice of a.
To apply the algorithm simply count the number of colors in each clique and count how many colors have the same number of vertecies.

□

Observation 8.1. *If $l > 5k$ and S^* is a solution to (G, k), then for all $C \in \mathcal{C}$, $S^* \cap Color^*_i(C) = \emptyset$ for all $i \in [max]^*_C$.*

Proof. For any $C \in \mathcal{C}$, we know that $[max]^*_C > 4k$ by Reduction rule 8.4. So if $Color^*_i(C) \cap S^* \neq \emptyset$ for some $i \in [max]^*_C$, then color i can not be balanced in $f_{S^*}(C)$ because there will be at least one color $j \in [max]^*_C$ where $Color^*_j(C) \cap S^* = \emptyset$ and $Color^*_j(C) \cap S = \emptyset$. □

(Rule 8.5) If $l > 5k$ and for some $C \in \mathcal{C}$ there is a $s \in S$ such that there exist verteceies $u, v \in V(C)$, where $c(u), c(v) \in [max]^*_C$, $su \in E(G)$ and $sv \notin E(G)$, then return $(G - s, k - 1)$.

Lemma 8.5. *Reduction rule 8.5 is sound and can be applied in time $\mathcal{O}(|V| + |E|)$.*

Proof. Since $c(u), c(v) \in [max]^*_C$ we know that for any potential solution S^* to (G, k), $u, v \notin S^*$ by Observation 8.1. Now since there exist a P_3 suv that means s must be in S^*. To apply the Reduction rule simply need to go through all verteceies in S and check if it forms a P_3 with 2 verteceies with color in $[max]^*_C$. For each vertex $s \in S$ check the neighborhood of s and see if it's connected to either all or none of the verteceies in C with color inn $[max]^*_C$. □

Observation 8.2. *If $l > 5k$, then for any clique $C \in \mathcal{C}$ the verteceies of C with color in $[max]^*_C$ form a module.*

Proof. For any verteceies $u, v \in C$ where $c(u), c(v) \in [max]^*_C$. If there is some $s \in S$ where $su \in E(G), sv \notin E(G)$ then Reduction rule 8.5 would remove s so this is impossible which means u and v are twins. □

We now present an algorithm to bound the number of colors. The algorithm goes through all cliques $C \in \mathcal{C}$ one at the time, and finds a color which is in $[max]^*_C$ and have no vertecies in S. It then deletes all the vertecies with this color which are in C. After having done this for all the cliques the algorithm pick some color j which have no vertecies inn S and relabels all the vertecies with color j to the deleted color in each clique. The result is that no vertex in G have color j.

Algorithm 8.1 Delete Color

1: $j \leftarrow$ any color such that $Color_j(S) = \emptyset$
2: **for** $C \in \mathcal{C}$ **do**
3: $\quad i \leftarrow$ any color $i \in [max]^*_C$ where $S \cap Color_i(C) = \emptyset$
4: $\quad G \leftarrow G - Color_i(C)$
5: $\quad$ **for** $v \in Color_j(C)$ **do**
6: $\quad\quad c(v) = i$
7: **return** G

(Rule 8.6) If $|[max]^*_C| > 4k$, then apply algorithm 8.1 to get G' and return (G', k).

Lemma 8.6. *Reduction rule 8.6 is safe and can be applied in time* $\mathcal{O}(|V|^2 + |E| \log |V|)$.

Proof. First we see that there is a color j on line 1 such that $Color_j(S) = \emptyset$ since $|[max]^*_C| > 4k$ and $|S| \leq 3k$. Also there is an color i on line 3 where $i \in |[max]^*_C|$ and $Color_i(C) = \emptyset$ for the same reason as above.
First assume S^* is a solution to (G, k). Let $C' = f_{S^*}(C)$, C_R is the clique C after applying algorithm 8.1 and $C'_R = f_{S^*}(C_R)$. Let i and j be the colors chosen at line 3 and 1 respectively. The colors in C'_R are clearly balanced since it simply has 1 less color and 1 color relabeled. Since $G' - S^* \subset G - S^*$ there are no P_3 in $G' - S^*$.
Now assume S^* is a solution to (G', k). Let $C' = f_{S^*}(C)$, C_R is the clique C after applying algorithm 8.1 and $C'_R = f_{S^*}(C_R)$. Let i and j be the colors chosen at line 3 and 1 respectively. It is clear than all colors except for i will be balanced with each other since they have the same number of vertecies in C' and C'_R, j is simply relabeled. For color i since no vertecies with color inn $[max]^*_C$ can be in S^* by Observation 8.1. Color i will be balanced as well. Then for any potential P_3 in $G - S^*$, let $C \in \mathcal{C}$ and i be the color chosen at line 3 in Algorithm 6.13 for that clique. Then the colors of $[max]^*_C$ form a module by Observation 8.2. Then applying Lemma 3.4 with $U' = Color_i(C)$ and U be all of vertecies in C with color in $[max]^*_C$. We apply Lemma 3.4 repeatedly for all cliques $C \in \mathcal{C}$ this way and get that

there is no P_3 in $G - S^*$ as well.

To apply the Reduction we first need to run algorithm 8.1. To do this we need to compute $[max]^*_C$ which can be done in $\mathcal{O}(|V|)$ time for all the cliques. Then to find a color which is not in S can also be done in $\mathcal{O}(|V|)$ time. finally deleting the vertecies in the end is done in $\mathcal{O}(|V| + |E| \log |V|)$ time. □

Observation 8.3. $l \leq 5k$

Proof. If $l > 5k$ then $|[max]^*_C| > 4k$ by reduction rule 8.4 which is not possible by Reduction rule 8.6, so $l \leq 5k$. □

We can now prove the final theorem, Theorem 4.

Theorem 4. *When $\alpha = 1$ there is a kernel with $\mathcal{O}(k^3)$ vertecies for αBALANCED CLUSTER VERTEX DELETION, which can be computed in time $\mathcal{O}(|V|^3(|V| + |E|))$.*

Proof. By Observation 8.3 and Theorem 2. □

Chapter 9

Open Problems

We have in this thesis studied the vertex deletion variant of $\alpha-$BCVD. But there are also natural edge modification variants of this problem. We can allow only edge deletions which lead to the problem called *CLUSTER EDGE DELETION* this problem should have a $\mathcal{O}(k^2)$ vertex kernel following the methods given for example by Steinvik [10]. A more interesting problem is the problem of *CLUSTER EDITING* where we are allowed to both add and delete edges. It is not clear whether this problem even admits a polynomial kernel since we can not simply delete isolate cliques.

α-Balanced Cluster Editing $(\alpha-BCE)$	
Input:	A colored graph (G, c) and integer k.
Output:	Does there exist a set $X \subseteq \binom{V(G)}{2}$ where $\lvert X\rvert \leq k$, so that $G \triangle X$ is an α-balanced cluster graph.

Open Problem 1. *Does α-BCE admit a polynomial kernel?*

Even in the easier problem where we are only allowed to add edges it's not obvious how do deal with isolated cliques.

α-Balanced Cluster Completion $(\alpha-BCC)$	
Input:	A colored graph (G, c) and integer k.
Output:	Does there exist a set $X \subseteq \binom{V(G)}{2}$ where $\lvert X\rvert \leq k$, so that $G + X$ is an α-balanced cluster graph.

Open Problem 2. *Does $\alpha-BCC$ admit a polynomial kernel?*

The best know kernel for *CVD* have $\mathcal{O}(k^{\frac{5}{3}})$ vertecies. It is probably possible to follow the same method used in [8] to obtain this kernel to obtain an $\mathcal{O}(k^{\frac{5}{3}}l^2)$ vertex kernel. A natural question is then whether we can remove one of the factors l.

Open Problem 3. *Does $\alpha-BCVD$ admit a kernel with $\mathcal{O}(k^{\frac{5}{3}}l)$ vertecies.*

Bibliography

[1] Suman Bera, Deeparnab Chakrabarty, Nicolas Flores, and Maryam Negahbani. Fair algorithms for clustering. *Advances in Neural Information Processing Systems*, 32, 2019.

[2] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. *Advances in Neural Information Processing Systems*, 30, 2017.

[3] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5. Springer.

[4] Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

[5] Michael R Garey and David S Johnson. Computers and intractability. *A Guide to the*, 1979.

[6] Falk Hüffner, Christian Komusiewicz, Hannes Moser, and Rolf Niedermeier. Fixed-parameter algorithms for cluster vertex deletion. *Theory of Computing Systems*, 47(1): 196–217, 2010.

[7] Tomohiro Koana and André Nichterlein. Detecting and enumerating small induced subgraphs in c-closed graphs. *Discrete Applied Mathematics*, 302:198–207, 2021.

[8] Tien-Nam Le, Daniel Lokshtanov, Saket Saurabh, Stéphan Thomassé, and Meirav Zehavi. Subquadratic kernels for implicit 3-hitting set and 3-set packing problems. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 331–342. SIAM, 2018.

[9] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[10] Andreas Steinvik. Kernelization for balanced graph clustering. 2020.

[11] Dekel Tsur. Faster parameterized algorithm for cluster vertex deletion. *Theory of Computing Systems*, 65(2):323–343, 2021.