Fachbereich Mathematik & Informatik
AG Algorithmik
Prof. Dr. C. Komusiewicz

Sommersemester 21

Master-Arbeit zum Thema

Complexity Analysis of Graph-Based Orthology Assignment

Betreuer: Prof. Dr. C. Komusiewicz

Name: Jaroslav Garvardt
Matrikelnummer: 2486741
Studiengang: Master Data Science (7. Fachsemester)
Email: jaroslav.garvardt@gmail.com
Datum der Abgabe: 09. Juni 2021

Hiermit versichere ich, Jaroslav Garvardt, dass ich die vorliegende Arbeit mit dem Titel *Complexity Analysis of Graph-Based Orthology Assignment* selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Master-Arbeit wurde in der jetzigen oder einer ähnlichen Form noch bei keiner anderen Hochschule eingereicht und hat noch keinen sonstigen Prüfungszwecken gedient.

Marburg, den 09.06.2021

Abstract

The assignment of orthologous genes is an important concept in comparative genomics that is also computationally challenging. There is a great variety of methods and algorithms proposed to infer orthology relations. One possible approach is to use graph-based models, where the genes of two or more species are represented as vertices of different colors in a graph and the goal is to find a clustering of the vertices such that genes inside the same cluster are very likely to be orthologs. In this work we propose several variants of graph problems in the context of orthology assingment and analyze their computational complexity. We show the NP-hardness for each variant and provide FPT-algorithms and problem kernels for the canonical parameter. We then conduct experiments on graphs obtained from biological data to provide a first assessment of the practical use of the models.

Zusammenfassung

Die Zuordnung von orthologen Genen ist ein wichtiges Konzept in der vergleichenden Genomik, das sich als anspruchsvolles Berechnungsproblem erweist. Es existiert eine große Vielfalt an Methoden und Algorithmen, die verfolgt werden um Orthologie-Beziehungen abzuleiten. Ein möglicher Ansatz ist die Verwendung von Graph-basierten Modellen, bei denen die Gene von zwei oder mehr Spezies als verschiedenfarbige Knoten in einem Graphen dargestellt werden und das Ziel darin besteht, eine Zuordnung der Knoten zu finden, so dass Gene innerhalb derselben Gruppe sehr wahrscheinlich ortholog zueinander sind. In dieser Arbeit führen wir mehrere Varianten von Graphenproblemen im Kontext der Zuordnung von orthologen Genen ein und analysieren ihre rechnerische Komplexität. Wir zeigen die NP-Härte für jede Variante und formulieren FPT-Algorithmen und Problemkerne für den kanonischen Parameter. Anschließend führen wir Experimente an Graphen durch, die aus biologischen Daten gewonnen wurden, um eine erste Einschätzung der praktischen Anwendung der Modelle zu geben.

Contents

1 Introduction

Orthology is a central concept in the field of evolutionary and comparative genomics [9, 17]. Two genes in different species are orthologs if they arose via a speciation event from a common ancestor in the gene tree. The identification of orthologous genes is of great importance for phylogenetic tree interference, genome annotation and the prediction of gene functions, but is also a difficult computational task [16].

Various models and methods have been proposed for the study of orthologous relations, which can be broadly divided into tree-based [3, 13] and graph-based approaches [14, 4].

A possible graph-based approach used by Zheng et al. [20] is to construct a graph, where the vertices of the graph represent the genes of the different species. Edges between two genes of distinct species are present if the genes are likely to be orthologs, based on sequence similarity scores between the genes of the different species. Then the goal is to find a clustering of the graph in groups of orthologous genes that contain at most one gene from each species, where, roughly speaking, there are many edges between genes inside the same cluster and few edges between genes from different clusters.

Paralogs, two genes from the same species that originate from a gene duplication event, pose an additional difficulty for orthology assignments. Zhen et al. [20] and Fertin et al. [10] proposed orthology assignment models that also consider paralogous genes.

In this work we introduce graph problems on bicolored undirected graphs, in which the common goal is to achieve a cluster graph by a minimal amount of edge modifications, where every cluster contains at most one vertex from one of the two colors. This can be seen as a generalization of the model analyzed by Fertin et al. [10], where we now also consider multiple gene duplication events that occurred for both species, resulting in orthology clusters that can contain more than one gene from one of the two species. Each cluster can be seen as a group of paralogs in one species that is orthologous to a gene of the other species.

The graph problems considered in this work have also close ties to and can be seen as an extension of the well-studied CLUSTER DELETION and CLUSTER EDITING problems.

The structure of this work is as follows. In Section 2 we establish the notation and definitions used throughout this work and give a brief overview of the most important concepts of computational complexity theory. In Section 3 we define the graph problems we analyze in this work. In Section 4 we prove the **NP**-hardness for all variants. More precisely, we show that each variant is **NP**-complete, even when restricted to graphs with maximum degree six. In Section 5 we then study the parameterized complexity of the variants, where we only allow edge deletions. For the solution size parameter k we formulate branching algorithms that also show the fixed-parameter-tractability of the problem. We then formulate for both problems reduction rules that lead to a vertex-kernel for the respective problem. In Section 6 we then conduct our study for the variants, where we also allow edge insertions. We again postulate **FPT**-algorithms for both variants and present a vertex-kernel for the parameter k. In Section 7 we then give an integer linear program formulation and run experiments for two of the considered problems, evaluating the general practicality of our model.

2 Preliminaries

In this section we first provide the notation and definitions we use throughout this work. We then give a brief overview about the most important concepts of parameterized computational complexity theory and give some further definitions.

2.1 Notation and Definitions

For a finite set X and an integer k, $0 \leq k \leq |X|$, we define $\binom{X}{k} = \{Y \subseteq X \mid |Y| = k\}$. In this work we consider undirected graphs $G = (V, E)$ consisting of a finite set of vertices $V(G) := V$ and a set of edges $E(G) := E \subseteq \binom{V}{2}$. We usually denote the sizes of those sets with $n := |V|$ and $m := |E|$.

For a subset of vertices $V' \subseteq V$ we denote with $G[V'] := (V', E \cap \binom{V'}{2})$ the *induced subgraph* of G which is induced by V' and with $G - V' := G[V \setminus V']$ the induced subgraph of G without the vertices in V' and their incident edges. For two subsets of vertices $V_1, V_2 \subseteq V$ we denote with $E_G(V_1, V_2) := \{\{v_1, v_2\} \in E(G) | v_1 \in V_1, v_2 \in V_2\}$ the set of edges between a vertex from V_1 and a vertex from V_2.

A *path* of length k, with $k \geq 0$, in a graph $G = (V, E)$ is a sequence $P = (v_1, v_2, \dots, v_{k+1}) \in V^{k+1}$ of pairwise distinct vertices in V such that $\{v_i, v_{i+1}\} \in E$ for all i, $1 \leq i \leq k$. Moreover, we call P a (v_1, v_{k+1})-path in G. A sequence $C = (v_1, v_2, \dots, v_{k+1}) \in V^{k+1}$, where $(v_1, v_2, \dots, v_k)$ is a (v_1, v_k)-path and $v_{k+1} = v_1$, $\{v_k, v_1\} \in E$, is called a *cycle* of length k.

For an integer k we refer to a graph that consists of a path of length $k-1$ as a P_k. We refer to a set of vertices $P = \{v_1, \dots, v_k\} \subseteq V$ as an *induced* P_k in G, if the induced subgraph $G[P]$ is a P_k. By abuse of notation we also refer to a path $P = (v_1, \dots, v_k)$ in G as an induced P_k, if $\{v_1, \dots, v_k\}$ is an induced P_k in G. If there is no induced P_k in G, we say that G is *P_k-free*.

For two vertices $u, v \in V$ the *distance* function $\operatorname{dist}(u, v)$ returns the length of the shortest (u, v)-path. If there is no path between u and v, we define $\operatorname{dist}(u, v) := \infty$. For an integer $i \geq 1$ and a vertex $v \in V$ we define the *(open) i-neighborhood* $N_G^i(v)$ of v as the set of all vertices $u \in V$ with $\operatorname{dist}(u, v) = i$ and $N_G^\infty(v) := \{u \in V \mid \operatorname{dist}(u, v) = \infty\}$. If the graph is clear from the context we usually omit the identifier in the index.

The *(open) neighborhood* of a vertex $v \in V$ is denoted by $N_G^1(v) := \{u \in V \mid \{v, u\} \in E\}$ or just $N_G(v)$ and the *closed neighborhood* by $N_G^1[v] := N_G(v) \cup \{v\}$ or just $N_G[v]$. The (open) neighborhood of a set of vertices $V' \subseteq V$ is defined by $N_G(V') = (\bigcup_{v \in V'} N_G(v)) \setminus V'$ and the closed neighborhood by $N_G[V'] := N_G(V') \cup V'$. The *degree* of a vertex v is the number of adjacent vertices $\deg_G(v) := |N_G(v)|$. The minimum and maximum degree of G are denoted by $\delta(G) \coloneqq \min\{\deg(v) \mid v \in V\}$ and $\Delta(G) \coloneqq \max\{\deg(v) \mid v \in V\}$, respectively.

We say a graph $G = (V, E)$ is *connected*, if for every pair of vertices $\{u, v\} \subseteq V$ there is an (u, v)-path in G, that is, if $\operatorname{dist}(u, v) < \infty$.

A subset of vertices $K \subseteq V$ is called a *connected component* of G, if $G[K]$ is connected and there is no $V' \subseteq V$ with $K \subset V'$ such that $G[V']$ is connected. A *singleton* is a connected component consisting of a single vertex, or in other words, a vertex without incident edges.

A subset of vertices $C \subseteq V$ such that $G[C]$ is complete in the sense that $E(G[C]) = \binom{C}{2}$ is called a *clique*. A *triangle* is a clique of size three.

A *critical clique* C is a clique, such that every vertex $v \in C$ has the same neighborhood and C is maximal under this property. A critical clique C is a *closed critical clique* if $C \cup N(C)$ is also a clique.

We call a connected component, which is a clique, a *cluster* and a graph, in which every connected component is a cluster, a *cluster graph*.

A *bicoloring* of a graph $G = (V, E)$ is a function $g : V \to \{black, white\}$ that labels every vertex of the graph as either black or white. When the context is clear we usually abbreviate the colors with b for black and w for white. For a set of vertices $V' \subseteq V$ we denote with $B(V') \coloneqq \{v \in V' \mid g(v) = b\}$ and $W(V') \coloneqq \{v \in V' \mid g(v) = w\}$ the black and white vertices in V', respectively. If for a given cluster C and a bicoloring g we have that at least half of the vertices in C have the same color x, we say C is *x-dominated* and if $g(c) = g(c')$ for any two $c, c' \in C$, we call C a *$g(c)$ monochromatic cluster*.

We denote a graph that consists of a clique that contains exactly two black and exactly two white vertices with $K_{(2,2)}$. We refer to a set of vertices $K \subseteq V$ as an *induced* $K_{(2,2)}$ in G, if the induced subgraph $G[K]$ is a $K_{(2,2)}$. If there is no induced $K_{(2,2)}$ in G, we say that G is *$K_{(2,2)}$-free*.

A *graph property* Π is defined by a family of graphs $\mathcal{G}_\Pi$ and we say a graph G satisfies the property Π if and only if $G \in \mathcal{G}_\Pi$.

Let Π be a graph property. We say Π is *quasi-hereditary*, if for every graph $G = (V, E)$ that satisfies Π the deletion of a certain vertex $v \in V$ yields a graph $G' = (V \setminus \{v\}, E')$ that also satisfies Π. If for every graph $G = (V, E)$ satisfying Π the deletion of any vertex $v \in V$ yields a graph $G' = (V \setminus \{v\}, E')$ that also satisfies Π, we call Π *hereditary*. Equivalently, a graph property is hereditary if it is preserved by induced subgraphs. Note that a hereditary graph property is also quasi-hereditary.

We say a family of graphs $\mathcal{G}$ has a *forbidden induced subgraph characterization*, if there is a set of graphs $\mathcal{F}$, such that a graph g belongs to $\mathcal{G}$ if and only if it does not contain any graph in $\mathcal{F}$ as an induced subgraph. A family of graphs $\mathcal{G}$ has a forbidden induced subgraph characterization if and only if for some hereditary graph property Π every graph in $\mathcal{G}$ satisfies Π [5].

2.2 Computational Complexity Theory

Formally, a decision problem is a language $L \subseteq \{0,1\}^*$. An instance $I \in \{0,1\}^*$ is a yes-instance of L if $I \in L$ and a no-instance otherwise. We say an algorithm solves a problem L, if it decides for every instance I, whether $I \in L$ or not. A *polynomial reduction* from a decision problem A to a decision problem B is an algorithm that has a polynomial running time and transforms an instance I_A of A into an instance I_B of B such that I_A is a yes-instance of A if and only if I_B is a yes-instance of B. If there is such an algorithm we say that A can be reduced to B and write $A \leq_P B$. Note that the reduction relation $\leq_P$ is transitive. The class of problems, which can be solved in polynomial time, is denoted by $\mathbf{P}$. A verifier $\mathcal{V}$ for a decision problem L is an algorithm that can verify an instance $I \in L$ in the sense that for every $x \in \{0,1\}^*$ there is a certificate $c \in \{0,1\}^{p(|x|)}$ for some polynomial function p such that $\mathcal{V}$ accepts the input (x, c) if and only if x is a yes-instance of L. The class of problems, for which there is a verifier running in polynomial time, is denoted by $\mathbf{NP}$. Note that $\mathbf{P} \subset \mathbf{NP}$ and this is widely assumed to be a proper inclusion. We say a problem A is $\mathbf{NP}$-hard, if for every problem B in $\mathbf{NP}$ we have $B \leq_P A$. If A is $\mathbf{NP}$-hard and $A \in \mathbf{NP}$ we say A is $\mathbf{NP}$-complete. An example for a $\mathbf{NP}$-hard problem is the Exact-3-SAT problem [11].

> Exact-3-SAT
> **Input**: A boolean formula ϕ in conjunctive normal form with exactly three literals per clause.
> **Question:** Is there an assignment to the variables of ϕ that satisfies all clauses of ϕ?.

For more details on computational complexity theory we refer to [11].

2.3 Parameterized Complexity

In this section we want to give an overview over the most important concepts of parameterized complexity theory. For a comprehensive read on the topic we refer to [8]. Parameterized complexity is a two-dimensional framework for describing the computational complexity of a decision problem. An instance (x, k) of a parameterized problem $L \subseteq \{0,1\}^* \times \mathbb{N}$ consists of the input x of a decision problem and a parameter k. A parameterized complexity class $\mathcal{L}$ is a set of parameterized problems. A parameterized problem L is called *fixed-parameter tractable*, if there is a computable function f such that for every instance $(x, k) \subseteq \{0,1\}^* \times \mathbb{N}$ it can be decided in $f(k) \cdot |x|^{O(1)}$

time whether (x, k) is a yes-instance of L. The class of problems that contains exactly those that are fixed-parameter tractable is called **FPT**. The class **XP** contains exactly all the parameterized problems, for which there is a computable function f such that for every instance $(x, k) \subseteq \{0, 1\}^* \times \mathbb{N}$ it can be decided in $|x|^{f(k)}$ time wether (x, k) is a yes-instance of L. A *parameterized reduction* is an algorithm that takes an instance $I_1 = (x_1, k_1)$ of a problem L_1 and transforms it into an instance $I_2 = (x_2, k_2)$ of a problem L_2 such that $I_1 \in L_1$ if and only if $I_2 \in L_2$ and $k_2 \leq g(k_1)$ with running time $f(k_1) \cdot |x_1|^{O(1)}$ for some computable functions f and g.

A parameterized problem L admits a *problem kernel*, if there is a parameterized reduction running in polynomial time that transforms an instance (x_1, k_1) of L into an instance (x_2, k_2) of L such that $k_2 \leq k_1$ and $|x_2| \leq h(k_1)$ for some computable function h, so we get an equivalent new instance, where the input size is upper-bounded by a computable function that only depends on the old parameter k_1. The function h is called the size of the kernel. We refer to such a parameterized reduction as a *kernelization algorithm* for the parameterized problem L. A parameterized problem is in **FPT** if and only if it admits a problem kernel [8].

A *data reduction rule*, or just reduction rule, for a parameterized problem L is an algorithm that runs in polynomial time and transforms an instance (x_1, k_1) of L into an instance (x_2, k_2) of L, such that $(x_1, k_1) \in L$ if and only if $(x_2, k_2) \in L$. A kernelization algorithm usually consists of exhaustively applying a set of reduction rules.

In this work an *edge-modification problem with property* Π is a decision problem that gets an undirected graph G and an integer k as inputs and asks, whether it is possible to transform G by at most k deletions or insertions of edges into a graph G' that satisfies the property Π. Analogously, a bicolored edge-modification problem is an edge-modification problem that also gets a bicoloring g as input.

An *edge-modification set* for an instance $(G = (V, E), k)$ of a parameterized edge-modification problem is a subset $S = S^- \dot\cup S^+ \subseteq \binom{V}{2}$ with $S^- \subseteq E$ corresponding to edge deletions and $S^+ \subseteq \binom{V}{2} \setminus E$ to edge insertions. *Applying* an edge-modification set S to the graph $G = (V, E)$ yields a new graph $G_S = (V, (E \setminus S^-) \cup S^+)$. In the case that for a given edge-modification problem only edge-deletions are allowed we have $E_2 = \emptyset$. We say a vertex $u \in V$ is *affected* by an edge-modification set S if $\{u, v\} \in S$ for any $v \in V$ and *unaffected*, otherwise. Analogously, we say a set of vertices V' is affected by an edge-modification set S, if any vertex $v \in V'$ is affected, or unaffected, if every vertex $v \in V'$ is unaffected.

A *solution set* or just *solution* for an instance $(G = (V, E), k)$ of a parameterized edge-modification problem with property Π is an edge-modification set S, such that applying S to G yields a solution graph G_S that satisfies the property Π. We say a solution S^* is *optimal*, if for every other solution S' we have $|S^*| \leq |S'|$. We say a solution S is *valid* for an instance (G, k) of a parameterized edge-modification problem, if $|S| \leq k$.

We analogously define an edge-modification set and a solution for an instance $(G = (V, E), g, k)$ of a bicolored edge-modification problem.

3 Problem Definitions

In this section we formulate the decision problems which we analyze in this work.

3.1 Definitions

In this work we consider several parameterized bicolored edge-modification problems on undirected graphs. For every variant the common goal is to transform the input graph into a cluster graph, that also satisfies a certain property Π depending on the bicoloring, by at most k edge modifications. Depending on the variant this can either be achieved by only edge deletions or a combination of edge deletions and insertions.

For a given bicoloring $g : V \to \{b, w\}$ we say a graph $G = (V, E)$ is a *bicolored cluster graph* or, in other words, satisfies the *bicolored cluster* property, if G is a cluster graph such that every cluster contains at most one black or at most one white vertex. If G is a cluster graph such that every cluster contains exactly one black or exactly one white vertex, we call it a *strictly bicolored cluster graph.*

First, we formulate the deletion variants.

> Bicolored Cluster Deletion (BCD)
> **Input**: An undirected graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$ and an integer k.
> **Question:** Can G be transformed into a bicolored cluster graph by at most k edge deletions?

Observe that in Bicolored Cluster Deletion monochromatic clusters in the resulting graph are allowed. Considering the biological interpretation of assigning orthology clusters it makes sense to only accept clusters of size two or more if they contain at least one vertex of both colors. This is expressed in the problem Strict Bicolored Cluster Deletion.

> Strict Bicolored Cluster Deletion (SBCD)
> **Input**: An undirected graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$ and an integer k.
> **Question:** Can G be transformed into a strictly bicolored cluster graph by at most k edge deletions?

For our next problem Bicolored Cluster Editing we also allow edge insertions.

Bicolored Cluster Editing (BCE)
Input: An undirected graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$ and an integer k.
Question: Can G be transformed into a bicolored cluster graph by at most k edge deletions or insertions?

Similar to the deletion variants, in the editing case we also want to differentiate between a non-strict and a strict variant.

Strict Bicolored Cluster Editing (SBCE)
Input: An undirected graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$ and an integer k.
Question: Can G be transformed into a strictly bicolored cluster graph by at most k edge deletions or insertions?

3.2 Simple Observations

We now proceed to state some useful observations about the aforementioned properties.

Lemma 3.1. *The bicolored cluster property is hereditary.*

Proof. Let G be a bicolored cluster graph. Recall that every connected component of G is a cluster with at most one black or at most one white vertex. Deleting a vertex v of any cluster K does not affect other clusters and $K \setminus \{v\}$ is still a cluster with at most one black or at most one white vertex, so $G[V \setminus \{v\}]$ is also a bicolored cluster graph. □

The property to be a strictly bicolored cluster graph is only quasi-hereditary, however.

Lemma 3.2. *The strictly bicolored cluster property is quasi-hereditary, but not hereditary.*

Proof. Let G be a strictly bicolored cluster graph, so every connected component of G is a cluster with exactly one black or exactly one white vertex. Consider a cluster K in G. If K consists of a single vertex v, then the deletion of v does not impact any other cluster and $G[V \setminus \{v\}]$ is still a strictly bicolored cluster graph. Now, assume that $|K| \geq 2$ and without loss of generality let K contain exactly one black vertex. Deleting any white vertex u in K results in a new cluster $K \setminus \{u\}$ that still contains exactly one black vertex and does not impact any other cluster, so $G[V \setminus \{u\}]$ is still a strictly bicolored cluster graph. This shows that for every strictly bicolored cluster graph there is a vertex v such that deleting v results in a strictly bicolored cluster graph, so the strictly bicolored cluster property is quasi-hereditary. Now, consider for example a strictly bicolored cluster graph that only consists of a single black-dominated cluster K of size $|K| \geq 3$ with exactly one white vertex. Deleting any black vertex in K again yields a black-dominated cluster with exactly one white vertex. However, deleting the white vertex in K creates a monochromatic black cluster, so the resulting graph is no longer strictly bicolored. Therefore the strictly bicolored cluster property is not hereditary. $\square$

A common way to describe a family of graphs is to use a forbidden induced subgraph characterization. For example, cluster graphs can be characterized as graphs that do not contain an induced P_3. This is because every cluster is per definition a clique and therefore does not contain any induced P_3s and if a graph G contains a set of vertices $V' := \{v_1, v_2, v_3\}$, such that V' is an induced P_3 in G, then V' cannot be part of a cluster and G is not a cluster graph.

Next we show that bicolored cluster graphs also have a forbidden induced subgraph characterization.

Lemma 3.3. *Let $G = (V, E)$ be a graph with a bicoloring $g : V \to \{b, w\}$ on G. Then the graph G is a bicolored cluster graph if and only if G is P_3-free and $K_{(2,2)}$-free.*

Proof. ($\Rightarrow$) Let G be a cluster graph, such that every cluster contains at most one black or at most one white vertex. Since G is a cluster graph, G is P_3-free. Furthermore, no cluster in G contains two black and two white vertices, so G is also $K_{(2,2)}$-free.

($\Leftarrow$) Let G be a graph that is P_3-free and $K_{(2,2)}$-free. Since G does not contain an induced P_3 it is a cluster graph. Now assume that G contains a cluster K with at least two black vertices b_1, b_2 and at least two white vertices w_1, w_2. Then, since K is a cluster, the vertices b_1, b_2, w_1 and w_2 form an induced $K_{(2,2)}$ in G, which contradicts that G is $K_{(2,2)}$-free. Therefore G only contains clusters with at most one black or at most one white vertex. $\square$

For the non-strict variants we call a cluster *valid*, if it is $K_{(2,2)}$-free. For the strict variants we call a cluster of size at least two *valid*, if it is $K_{(2,2)}$-free and is not a monochromatic cluster. Singletons are also considered as a valid cluster.

Let $P := (v_1, v_2, v_3)$ be an induced P_3 in a graph G and S an edge modification set. We say that P is *resolved* by S, if the edge $\{v_1, v_3\}$ is inserted by S or at least one of the edges $\{v_1, v_2\}$ or $\{v_2, v_3\}$ is deleted by S. Let K

be an induced $K_{(2,2)}$ in a graph G and S an edge modification set. We say that K is *resolved* by S, if at least one of the edges between vertices in K is deleted by S.

4 NP-hardness results

To show the **NP**-hardness of the several problem variants we use a reduction from the **NP**-hard problem Exact-3-SAT very similar to the one used by Komusiewicz and Uhlmann [15] for Cluster Editing. They showed that Cluster Editing is **NP**-hard, even when restricted to graphs with maximum degree six. We show **NP**-hardness SBCE, the same reduction and similar argumentation then also implies **NP**-hardness for the other variants.

First we want to recap the reduction presented by Komusiewicz and Uhlmann [15] and then explain the adjustments made for the problem variants we consider in this work. The basic idea of the reduction is as follows. For a given 3-CNF formula ϕ with n variables and m clauses there is a variable cycle of length $4m_i$ for every variable x_i, with m_i being the number of clauses containing x_i. For a cycle with even length such as $4m_i$, deleting every second edge yields a minimum-cardinality edge modification set to transform the cycle into a cluster graph. The two possibilities of either deleting every edge that is labeled odd or every edge that is labeled even in the cycle represent a `true` or `false` assignment to x_i. Furthermore, for every clause C_j we have a clause gadget, consisting of a vertex a_j representing the clause, which is connected to the variable cycles of the variables in C_j in such a way, that each of those variable cycles has four designated vertices for C_j. The idea is to ensure that only four edge modifications per clause are needed, if there is a satisfying assignment for ϕ, and at least five edge modifications otherwise. We now present the details of the reduction.

Construction. For a given 3-CNF formula ϕ with clauses $C_0, \ldots, C_{m-1}$ over the variables $\{x_0, \ldots, x_{n-1}\}$ we construct the graph of a Cluster Editing instance $(G = (V, E), k)$ with $k = 10m$ as follows. For each variable x_i, $0 \leq i < n$, the graph G contains a variable cycle Z_i that consists of the vertices $V_i := \{v_0^i, \ldots, v_{4m_i-1}^i\}$ and the edges $E_i^v := \{\{v_k^i, v_{k+1}^i\} \mid 0 \leq k < 4_{m_i}\}$ with $v_{4m_i}^i = v_0^i$. For ease of presentation we always interpret v_y^i as $v_{y \bmod 4m_i}^i$. We call an edge $\{v_y^i, v_{y+1}^i\}$ *even*, if y is even, and *odd* otherwise. Now, for every variable x_i we have an arbitrary, but fixed ordering of the clauses that contain x_i given by a bijective function π_i, with $\pi_i(j) \in \{0, \ldots, m_i - 1\}$ being the position of a clause C_j containing x_i in this ordering. For every clause C_j with variables x_p, x_q, x_r we construct a clause gadget by adding a new vertex a_j and connecting a_j to the variable cycles of x_p, x_q and x_r. For each $i \in \{p, q, r\}$ we add the edges $\{a_j, v_{4\pi_i(j)}^i\}$ and $\{a_j, v_{4\pi_i(j)+1}^i\}$ if x_i occurs nonnegated in C_j or the edges $\{a_j, v_{4\pi_i(j)+1}^i\}$ and $\{a_j, v_{4\pi_i(j)+2}^i\}$ otherwise. We denote the set of edges in the clause gadget of C_j by E_j^c. This completes the construction of the graph $G = (V, E)$ with $V := \bigcup_{i=0}^{n-1} V_i \cup \bigcup_{j=0}^{m-1} \{a_j\}$

and $E \coloneqq \bigcup_{i=0}^{n-1} E_i^v \cup \bigcup_{j=0}^{m-1} E_j^c$.

Now we make adjustments to the constructed graph G. We specify a bicoloring g of V in order to obtain an instance $(G = (V, E), g, 10m)$ of SBCE. Let $g : V \to \{b, w\}$ be a bicoloring of the vertices in V, such that $g(a_j) = w$ for every clause vertex a_j. Furthermore, for every vertex v_s^i in the variable cycle Z_i of a variable x_i we have $g(v_s^i) = w$ if s is even and $g(v_s^i) = b$ otherwise. We set $W \coloneqq \{v \in V \mid g(v) = w\}$ and $B \coloneqq \{v \in V \mid g(v) = b\}$.

In the following we show that ϕ has a satisfying assignment if and only if $(G, g, 10m)$ is a yes-instance of SBCE.

Lemma 4.1. *If ϕ has a satisfying assignment β, then there is a valid solution $S \subseteq E$ for $I \coloneqq (G, g, 10m)$ only consisting of edge deletions, such that G_S is a cluster graph where every cluster contains exactly one black and either one or two white vertices.*

Proof. For each variable x_i, if $\beta(x_i) = \texttt{true}$, then delete all edges in E_i^v of the variable cycle Z_i in G that are odd and if $\beta(x_i) = \texttt{false}$, then delete all of the even edges in Z_i, resulting in

$$\sum_{0 \le i < n} 4m_i/2 = 2 \sum_{0 \le i < n} m_i = 6m$$

edge deletions in total. This resolves any induced P_3s containing only vertices of the same variable cycle. Now, for each clause C_j of ϕ consider the clause gadget in G and let x_p, x_q, x_r be the variables of that clause. Without loss of generality assume that the literal corresponding to x_p fulfills C_j under the assignment β. Deleting the four edges between a_j and a vertex in $V_q \cup V_r$ resolves all induced P_3s containing a_j, since, by construction, the two endpoints of the remaining edges between a_j and V_p are connected (by an odd edge, if x_p appears negated in C_j, or an even edge, otherwise) and the literal containing x_p in C_j is $\texttt{true}$ under the assignment β and therefore the connecting edge was not deleted. This procedure deletes four edges per clause gadget in G, so we apply another $4m$ deletions and therefore need $10m$ edge deletions in total. Since there are no edges between different variable cycles and therefore no induced P_3s involving vertices from different variable cycles, this destroys every induced P_3 in G, yielding a cluster graph G'. Every connected component of G' is either an edge of a variable cycle with a black and a white vertex or a triangle consisting of an edge of a variable cycle and a clause vertex a_j, containing one black and two white vertices. Therefore applying the set S of $10m$ edge deletions described above to G yields a cluster graph $G_S = G'$, where every cluster contains exactly one black and either one or two white vertices. Moreover G' is a cluster graph such that every cluster contains at most one black vertex. Therefore I is a yes-instance and S a valid solution. $\square$

Now, we show the opposite direction of the equivalence.

Lemma 4.2. *If $I := (G, g, 10m)$ is a yes-instance of* SBCE*, then I has an optimal solution $S \subseteq E$ only consisting of edge deletions with $|S| = 10m$, such that G_S is a cluster graph where every cluster contains exactly one black and either one or two white vertices and ϕ has a satisfying assignment β.*

Proof. Let $I := (G, g, 10m)$ be a yes-instance for SBCE. Let S be a valid solution for G with $|S| \leq 10m$. We show that S also has to contain at least, and therefore exactly, $10m$ edge modifications, which are all edge deletions. Per construction in every variable cycle Z_i of length $4m_i$ there are $2m_i$ edge-disjoint induced P_3s with all three involved vertices on the variable cycle that each require an edge modification to be resolved. Clearly, either deleting all even or all odd edges resolves all of the induced P_3s with $2m_i$ deletions. Consider one of those P_3s, say $P := (v^i_{j-1}, v^i_j, v^i_{j+1})$. Note that, by construction, v^i_{j-1} and v^i_{j+1} have the same color. Adding the edge $\{v^i_{j-1}, v^i_{j+1}\}$ to resolve P implies that the edges $\{v^i_{j-2}, v^i_{j-1}\}$ and $\{v^i_{j+1}, v^i_{j+2}\}$ must be deleted, since $g(v^i_{j-2}) = g(v^i_{j+2}) \neq g(v^i_{j-1}) = g(v^i_{j+1})$. This amounts to three edge-modifications and still leaves $2m_i - 2$ edge-disjoint induced P_3s in Z_i, each requiring at least one edge-modification. When one odd and one even edge is deleted there are also still $2m_i - 1$ edge-disjoint induced P_3s left. The minimum amount $2m_i$ of edge-modifications to resolve all P_3s in Z_i can therefore only be achieved by deleting either all even or all odd edges. Overall at least $\sum_{0 \leq i < n} 4m_i/2 = 6m$ edge deletions are necessary to destroy all induced P_3s of the variable cycles.

Now, consider an arbitrary clause vertex a_j of a clause C_j containing the variables x_p, x_q, x_r. At least four edge modifications are needed to resolve all induced P_3s containing edges incident to the vertex a_j, since every edge connecting a_j to the variable cycle of one of the variables in C_j forms an induced P_3 with any edge connecting a_j to another variable cycle. We now proceed to show that the minimum of four edge modifications can only be achieved by deleting all edges connecting a_j to two of the adjacent variable cycles.

Let K_j denote the cluster containing a_j in G_S and $W_j = N_G(a_j) \cap K_j$ denote the set of neighbors of a_j in G that are part of the same cluster K_j in G_S with $\rho := |W_j|$. Note that, since S is a solution and a_j is white, W_j can not contain two black vertices and a white vertex at the same time, thus $\rho \leq 4$. Furthermore, K_j can contain at most one edge between two neighbors of a_j on the same variable cycle, since all three of those edges each connect a black and a white vertex. This means that K_j contributes to S with $6-\rho$ deletions and if $\rho > 1$ at least $\binom{\rho}{2} - 1$ insertions, one for each pair of vertices in W_j minus one for the edge that may already exist, only involving vertices in $W_j \cup \{a_j\}$.

We now consider all possible values for ρ:

- $\rho = 4$: S deletes two edges between a_j and $N_G(a_j)$ and inserts at least five edges between vertices in W_j.
- $\rho = 3$: S deletes three edges between a_j and $N_G(a_j)$ and inserts at least two edges between vertices in W_j.
- $\rho = 2$: S deletes four edges between a_j and $N_G(a_j)$ and inserts one edge if the two vertices in W_j were not already connected.
- $\rho \leq 1$: S deletes at least five edges between a_j and $N_G(a_j)$.

In any case, at least four edge modifications are needed. The only possibility where S contains exactly four edge modifications involving only vertices in $W_j \cup \{a_j\}$ is when $\rho = 2$ and the two remaining neighbors of a_j are from the same variable cycle and are still connected by an edge. At least $4m$ edge modifications are needed to resolve all induced P_3s containing clause vertices a_j.

Since at least $6m$ edge modifications are already needed to resolve all induced P_3s in the variable cycles, the total amount of edge modifications is at least $10m$, showing that $|S| \geq 10m$ and therefore $|S| = 10m$. Now, since S is a solution, this also implies that indeed in every variable cycle either all odd or all even edges are deleted. Furthermore, for every clause vertex a_j the induced P_3s involving a_j are resolved by exactly four edge deletions, such that the two remaining neighbors of a_j in G_S are from the same variable cycle.

Let β be the assignment for ϕ that sets a variable x_i, $0 \leq i < n$, to $\beta(x_i) \coloneqq$ `true` if all odd edges of its variable cycle are deleted and $\beta(x_i) \coloneqq$ `false` if all even edges of its variable cycle are deleted. We show that β is a satisfying assignment for ϕ. Let C_j be an arbitrary clause of ϕ containing the variables x_p, x_q, x_r. In the final cluster graph G_S resulting by application of S to G the vertex a_j can not be the center of an induced P_3, so only edges to at most one variable cycle can be present. Without loss of generality let Z_p be that cycle. Since exactly four of the six edges incident to a_j are deleted, both edges connecting a_j to the variable cycle Z_p of x_p are not deleted by S as shown above. Without loss of generality, assume that x_p appears nonnegated in C_j. Then the two vertices adjacent to a_j are $v^j_{4\pi_p(j)}$

and $v^j_{4\pi_p(j)+1}$. Since S is a solution, the even edge $\{v^j_{4\pi_p(j)}, v^j_{4\pi_p(j)+1}\}$ is also not deleted by S and therefore no even edge of Z_p is deleted. As x_p appears nonnegated in C_j and the remaining edges of Z_p are all the even edges, C_j is fulfilled by $\beta(x_p) \coloneqq \texttt{true}$. Every resulting cluster in G_S is either an edge of a variable cycle or a triangle consisting of the two vertices of an edge of a variable cycle and a clause vertex a_j, so every cluster consists of exactly one black and either one or two white vertices. □

Using the reduction above we can now proof the **NP**-hardness for the several bicolored edge modification problems.

Theorem 4.3. BCD*,* SBCD*,* BCE *and* SBCE *are* ***NP****-complete, even when restricted to graphs with maximum degree six.*

Proof. It is easy to see that BCD, SBCD, BCE and SBCE and are all in **NP**. A possible verifier $\mathcal{V}$ would verify a yes-instance (G, g, k) by taking a solution S as a certificate, applying S to G and checking in polynomial time that G_S fulfills the desired property.

Since Exact-3-SAT is **NP**-hard, Lemma 4.1 and Lemma 4.2 directly imply the **NP**-hardness of SBCE. Furthermore, if (G, g, k) is a yes-instance for SBCE with a valid solution S that only performs edge deletions, then (G, g, k) clearly is also a yes-instance with valid solution S for every other bicolored edge modification problem we consider. This implies that the reduction above together with Lemma 4.1 and Lemma 4.2 also shows the **NP**-hardness for the other variants. Since the constructed graph in the reduction has a maximum degree six, this implies that the bicolored edge modifications problems are **NP**-complete, even when restricted to graphs with maximum degree six. □

5 Deletion Variants

In this section we study the parameterized complexity of the deletion variants BCD and SBCD, parameterized by the solution size k. First, we present simple **FPT**-algorithms for both variants. Next, we formulate reduction rules that lead to a linear-vertex problem kernel for BCD and a non-linear but subquadratic-vertex problem kernel for SBCD.

We often compare the number of edge deletions needed to achieve a certain valid clustering with the number of edge deletions needed for another valid clustering. For this we will make use of the following observation.

Observation 1. *Let C be a clique. Splitting C into two cliques C_1 and C_2 with $|C_1| \leq |C_2|$ requires $|C_1| \cdot |C_2|$ edge deletions. The number of edge deletions needed is at least $|C| - 1$ and is monotonously increasing with the size of C_1.*

5.1 FPT-Algorithms

5.1.1 Bicolored Cluster Deletion

One way to get an **FPT**-algorithm for BCD is to use the forbidden subgraph characterization of bicolored cluster graphs. Let $I \coloneqq (G, g, k)$ be an instance of BCD. If I is a yes-instance, then there must be a valid solution S such that G_S does not contain an induced P_3 and no induced $K_{(2,2)}$. Therefore, we can solve an instance I by resolving all induced P_3s and $K_{(2,2)}$s. This can be achieved by branching over the possible edge deletions that resolve a given induced P_3 or induced $K_{(2,2)}$. We formulate two branching rules that are applied by our **FPT**-algorithm for BCD. If the application of a branching rule results in k being negative, the corresponding branching case is a no-instance and the algorithm returns to the parent node in the search tree. The first rule resolves an induced P_3.

Branching Rule 1. *Let $G = (V, E)$ be a graph with a bicoloring $g : V \to \{b, w\}$ and let $k \geq 0$. If G contains an induced P_3, denoted by $P \coloneqq (v_1, v_2, v_3)$, branch into the cases:*

1. *remove the edge $\{v_1, v_2\}$ from G and decrease k by one;*

2. *remove the edge $\{v_2, v_3\}$ from G and decrease k by one.*

Since both branching cases of the rule reduce the parameter k by one, Branching Rule 1 admits the branching vector $(1, 1)$, which has a branching number of $\beta(1, 1) = 2$.

Exhaustively applying Branching Rule 1 branches over all possibilities to transform G into a cluster graph with at most k edge deletions. In order to achieve a bicolored cluster graph, additionally every induced $K_{(2,2)}$ must be resolved.

Let K be an induced $K_{(2,2)}$ in G consisting of two black vertices b_1, b_2 and two white vertices w_1, w_2. A trivial branching would in each case remove one of the $\binom{4}{2} = 6$ edges between vertices in K and reduce k by one. This would lead to a branching rule with branching vector $(1, 1, 1, 1, 1, 1)$ and branching number $\beta(1, 1, 1, 1, 1, 1) = 6$. However, we can achieve a much better branching by considering which of the vertices in K can stay together in a cluster in a solution graph. Let S be a solution. Let C be a cluster in G_S that contains at least one vertex from K. Clearly, C can not contain all four vertices from K, otherwise it would contain an induced $K_{(2,2)}$. We thus have three cases:

- If C contains exactly one vertex from K, then S deletes all three edges to the other three vertices.
- If C contains exactly two vertices from K, then S deletes all four edges to the other two vertices.
- If C contains exactly three vertices from K, then S again deletes all three edges between the last vertex from K and the vertices in $C \cap K$ as in the first case.

In each case K is no longer an induced $K_{(2,2)}$ in G_S and every solution S has to delete at least all three or four edges according to one of the cases, so that K is no longer an induced $K_{(2,2)}$ in G_S. Therefore, the following branching rule resolves an induced $K_{(2,2)}$ in G.

Branching Rule 2. *Let $G = (V, E)$ be a graph with a bicoloring $g : V \to \{b, w\}$ and let $k \geq 0$. If G contains an induced $K_{(2,2)}$, consisting of two black vertices b_1, b_2 and two white vertices w_1, w_2, branch into the cases:*

1. *remove all edges between b_1 and $\{b_2, w_1, w_2\}$ from G and decrease k by 3.*
2. *remove all edges between b_2 and $\{b_1, w_1, w_2\}$ from G and decrease k by 3.*
3. *remove all edges between w_1 and $\{b_1, b_2, w_2\}$ from G and decrease k by 3.*
4. *remove all edges between w_2 and $\{b_1, b_2, w_1\}$ from G and decrease k by 3.*
5. *remove all edges between $\{b_1, b_2\}$ and $\{w_1, w_2\}$ from G and decrease k by 4.*

6. *remove all edges between* $\{b_1, w_1\}$ *and* $\{b_2, w_2\}$ *from* G *and decrease* k *by 4.*
7. *remove all edges between* $\{b_1, w_2\}$ *and* $\{b_2, w_1\}$ *from* G *and decrease* k *by 4.*

Branching Rule 2 admits the branching vector $(3,3,3,3,4,4,4)$ with branching number $\beta(3,3,3,3,4,4,4) \approx 1.78$. We can now formulate the **FPT**-algorithm for BCD that gets initially called with an edge modification set $S := \emptyset$.

Algorithm 1:

Input: A graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$, an integer k and an edge modification set S.
Output: A valid solution S^*, if one exists.

```
if k < 0 then
    Return to the parent node in the search tree;
else
    Search for an induced P_3 and an induced K_(2,2) in G;
    if an induced P_3 in G was found then
        Apply Branching Rule 1;
    else if an induced K_(2,2) in G was found then
        Apply Branching Rule 2;
    else
        Return the solution set S;
    end
end
```

Since Branching Rule 1 has a branching number of 2 and Branching Rule 2 has a lower branching number, the size of the search tree can be upper bounded by 2^k and we get the following proposition.

Proposition 1. BCD *is in* **FPT** *and can be solved in* $O(2^k \cdot n^{O(1)})$ *time.*

In practice, Branching Rule 2 is not needed. Since edge insertions are not allowed and therefore two vertices from different connected components can never be part of the same cluster in the resulting cluster graph, we have the following observation.

Observation 2. *Let* (G, g, k) *be an instance of* BCD. *Each connected component of* G *can be solved independently.*

Let $I := (G, g, k)$ be an instance of BCD. Using Observation 2, we show that if the input graph G is already P_3-free, or in other words a cluster graph, we can solve I in polynomial time. For this we will make use of the following lemma.

Lemma 5.1. *Let $I := (G, g, k)$ be an instance of* BCD*. Let K be a cluster in G with $b_K > 1$ black and $w_K > 1$ white vertices. Then every optimal solution S^* for I splits K into exactly two clusters.*

Proof. Let S be a solution for I. Since K is not valid, there are at least two clusters in G_S containing vertices of K. Let $\mathcal{C}_K = \{C_1, C_2, \ldots, C_r\}$, with $r > 1$, denote the valid clusters in G_S that each contain at least one vertex of K. Since S is a solution, every cluster in $\mathcal{C}_K$ is a monochromatic black cluster, a monochromatic white cluster, a valid black-dominated cluster or a valid white-dominated cluster.

Suppose that $r > 2$. We show that we can always find a better solution S^* with $|S^*| < |S|$ that only creates two clusters instead of r many. To this end we step-wise merge some clusters that contain vertices from K, so that in the end we only have one valid black-dominated cluster and one valid white-dominated cluster left. We describe the merging-steps for black-dominated clusters, the merging-steps for white-dominated clusters are analogous.

1. Let C_i and C_j be two different monochromatic black clusters. We can merge C_i and C_j into a single monochromatic black cluster $C_i \cup C_j$, which does not require the deletion of edges between C_i and C_j and gives us a better solution $S' := S \setminus E(C_i, C_j)$ with one less cluster in $\mathcal{C}_K$.

   After step 1 we can assume that there is at most one black monochromatic cluster in $\mathcal{C}_K$.

2. Let C_i be a valid black-dominated cluster and C_j be a black monochromatic cluster. We can merge them into a single valid black-dominated cluster $C_i \cup C_j$ which does not require the deletion of edges between C_i and C_j and gives us a better solution $S' := S \setminus E(C_i, C_j)$ with one less cluster in $\mathcal{C}_K$.

   After step 2 we can assume that if there is a monochromatic black-cluster in $\mathcal{C}_K$ that it is the only black-dominated cluster in $\mathcal{C}_K$.

3. Let C_i and C_j be two different valid black-dominated cluster with $W(C_j) = \{w_j\}$. We can separate w_j from C_j and merge the monochromatic black cluster $C'_j := C_j \setminus \{w_j\}$ with C_i. This requires $|C_j| - 1$ additional edge deletions to separate w_j from the other vertices in C_j, but also saves $(|C_j| - 1) \cdot |C_i|$ deletions due to the merging. Thus, we obtain us a better solution $S' := \big(S \setminus E(C_i, C'_j)\big) \cup E(C_i, w_j)$. The solution S' merges two black-dominated clusters in $\mathcal{C}_K$ into one, but also creates the new white singleton w_j.

Case 1: $\mathcal{C}_K$ contains a white-dominated cluster C_ℓ. Then we can merge w_j with C_ℓ and get a better solution $S'' := S' \setminus E(w_j, C_\ell)$.

Case 2: $\mathcal{C}_K$ does not contain a white-dominated cluster. Then, after splitting w_j from the other vertices in C_j, we have that $C_\ell := \{w_j\}$ constitutes a new white-dominated cluster.

In both cases we can find a better solution that contains one less black-dominated cluster and at least one white-dominated cluster in $\mathcal{C}_K$.

After step 3 we can assume that $\mathcal{C}_K$ contains exactly one valid black-dominated and exactly one valid white-dominated cluster, otherwise we could merge some clusters according to step 1, 2 or 3.

□

Lemma 5.2. *Let $I := (G, g, k)$ be an instance of* BCD*, such that G is a cluster graph. Then, I can be solved in $O(n + m)$ time.*

Proof. According to Observation 2 every cluster of G can be solved independently. A cluster which is either a monochromatic cluster or contains exactly one black or exactly one white vertex is already valid. Hence, we only have to consider components that contain at least two black and two white vertices.

Let K be a connected component of G with $b_K := |B(K)| > 1$ black and $w_K := |W(K)| > 1$ white vertices.

According to Lemma 5.1, we can assume that a minimum-cardinality solution S^* splits K into exactly two clusters C_1 and C_2 in G_{S^*}. It remains to show that the minimal set of edge deletions that splits K into C_1 and C_2 can be determined in polynomial time.

According to Observation 1 splitting K into C_1 and C_2 needs $|C_1| \cdot |C_2|$ edge deletions and this number is minimal if, without loss of generality, $|C_1|$ is as small as possible or in other words $|C_2|$ is as big as possible. Since C_1 and C_2 both need to be a valid cluster, the maximal size of C_2 depends on the numbers b_K and w_K.

Case 1: $b_K \leq w_K$. The cluster C_2 can at most contain all w_K white and a black vertex $b \in B(K)$. Thus, splitting K into $C_1 := B(K) \setminus \{b\}$ and $C_2 := K \setminus C_1$ is optimal.

Case 2: $b_K > w_K$. The cluster C_2 can at most contain all b_K black and a white vertex $w \in W(K)$. Thus, splitting K into $C_1 := W(K) \setminus \{w\}$ and $C_2 := K \setminus C_1$ is optimal.

Given b_K and w_K we can therefore determine the minimal number of edge deletions involving vertices in K in constant time.

Finding every connected component K and determining the corresponding values b_K and w_K can be done in $O(n + m)$ time by using a modified breadth-first-search. Constructing the optimal set of edge deletions according to the values of b_K and w_K can then also be done in $O(n+m)$ time. □

Using Lemma 5.2 we get a faster **FPT**-algorithm for BCD by resolving all induced P_3s by branching via Branching Rule 1 and then solving the resulting cluster graph as described in Lemma 5.2. The following algorithm is again initially called with an edge modification set $S := \emptyset$.

Algorithm 2:

Input: A graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$, an integer k and an edge modification set S.
Output: A valid solution S^*, if one exists.

```
if k < 0 then
|   Return to the parent node in the search tree;
else
|   Search for an induced P3 in G;
|   if an induced P3 in G was found then
|   |   Apply Branching Rule 1;
|   else
|   |   Solve each connected component K according to Lemma 5.2;
|   |   if k ≥ 0 then
|   |   |   Return the solution set S;
|   |   end
|   end
end
```

Theorem 5.3. BCD *can be solved in* $O(2^k \cdot (n+m))$ *time.*

Proof. Algorithm 2 first branches over all possibilities to resolve a given induced P_3. This is correct, since, if P is an induced P_3 in G, every optimal solution S^* must resolve P in order to yield a bicolored cluster graph. Resolving all induced P_3s results in a cluster graph that then can be solved according to Lemma 5.2.

In each node of the search tree finding an induced P_3 can be done in $O(n+m)$ time. If no induced P_3 can be found, according to Lemma 5.2 the given instance in that node of the search tree can be solved in $O(n+m)$ time. Since Branching Rule 1 admits the branching number 2, the size of the search tree is bounded by 2^k. The total worst-case running-time of the algorithm is therefore $O(2^k \cdot (n+m))$. □

5.1.2 Strict Bicolored Cluster Deletion

Now we present a very similar **FPT**-algorithm for SBCD. Note that, since the strict bicolored cluster property is not hereditary, strict bicolored cluster graphs have no forbidden subgraph characterization. However, we can again first resolve all induced P_3s and then determine an optimal solution in polynomial time.

Since only edge deletions are allowed we can again make the following observation.

Observation 3. *Let* (G, g, k) *be an instance of* SBCD*. Each connected component of* G *can be solved independently.*

Let $I := (G, g, k)$ be an instance of SBCD. Using Observation 3 we again show that if the input graph G is already P_3-free, or in other words if G is a cluster graph, we can solve I in polynomial time.

Lemma 5.4. *Let* $I := (G, g, k)$ *be an instance of* SBCD*. Let* K *be a cluster in* G *with* $b_K > 1$ *black and* $w_K > 1$ *white vertices. Every optimal solution* S^* *for* I *splits* K *into exactly two clusters, one of which is black-dominated and one of which is white-dominated.*

Proof. Let S be a solution for I. Since K is not valid, there are at least two clusters in G_S containing vertices of K. Let $\mathcal{C}_K = \{C_1, C_2, \ldots, C_r\}$, with $r > 1$, denote the valid clusters in G_S that each contain at least one vertex of K. Since S is a solution, every cluster in $\mathcal{C}_K$ is a valid black-dominated cluster or a valid white-dominated cluster.

Suppose that $r > 2$. We show that we can always find a better solution S^* with $|S^*| < |S|$ that only creates two clusters instead of r many. To this end we step-wise rearrange and merge some clusters that contain vertices from K, so that in the end we only have one valid black-dominated cluster and one valid white-dominated cluster left.

Case 1: $\mathcal{C}_K$ contains a white-dominated cluster C_W and a black-dominated cluster C_B.

Let C_i be another black-dominated cluster in $\mathcal{C}_K$ with $W(C_i) = \{w_i\}$. We define three sets of edges. $E_1 := E(\{w_i\}, C_W)$, $E_2 := E(C_i \setminus \{w_i\}, C_B)$ and $E_3 := E(\{w_i\}, C_i \setminus \{w_i\})$. Consider the edge-modification set $S' := \big(S \setminus (E_1 \cup E_2)\big) \cup E_3$ that splits w_i from the rest of C_i and merges w_i with C_W into a new cluster C'_W and $C_i \setminus \{w_i\}$ with C_B into a new cluster C'_B. Since w_i is a white vertex and every vertex in $C_i \setminus \{w_i\}$ is black, C'_W is still a valid white-dominated and C'_B still a valid black-dominated cluster. Clearly $|E_3| < |E_1| + |E_2|$, so we have $|S'| < |S|$.

This shows that if $\mathcal{C}_K$ contains a black-dominated cluster C_i with $C_i \neq C_B$, we can always find a better solution where C_B is the only black-dominated cluster in $\mathcal{C}_K$. Analogously the same is true for a white-dominated C_j cluster and C_W. Thus, we can assume that $\mathcal{C}_K$ contains a single white-dominated cluster C_W and a single black-dominated cluster C_B.

Case 2: $\mathcal{C}_K$ contains two white-dominated clusters C_{W_1} and C_{W_2} and no black-dominated cluster.

In this case we can rearrange C_{W_1} and C_{W_2} as follows, so that we have a white-dominated cluster and a black-dominated cluster in $\mathcal{C}_K$ and Case 1 applies. Let b_2 denote the single black vertex in C_{W_2} and let w_2 denote a white vertex in C_{W_2}. Consider the edge modification set

$$S' \coloneqq \big(S \setminus E(C_{W_2} \setminus \{b_2, w_2\}, C_{W_1})\big) \cup E(\{b_2, w_2\}, C_{W_2} \setminus \{b_2, w_2\})$$

that leaves b_2 and w_2 as the cluster $C'_{W_2} \coloneqq \{b_2, w_2\}$ and merges all other (white) vertices in C_{W_2} together with C_{W_1} into the valid white-dominated cluster $C'_{W_1} \coloneqq C_{W_1} \cup (C_{W_2} \setminus \{b_2, w_2\})$. Compared to S, the solution S' adds $2 \cdot (|C_{W_2}| - 2)$ edge deletions, but also saves $|C_{W_1}| \cdot (|C_{W_2}| - 2)$ edge deletions. Since C_{W_1} is a valid cluster and thus $|C_{W_1}| \geq 2$, we have $|S'| \leq |S|$. As C'_{W_2} only consists of a single black and a single white vertex, C'_{W_2} constitutes a valid black-dominated cluster and Case 1 applies.

Case 3: $\mathcal{C}_K$ contains two black-dominated clusters C_{B_1} and C_{B_2} and no white-dominated cluster.

Analogously to Case 2 we can rearrange C_{B_1} and C_{B_2} so that we again have a white-dominated cluster and a black-dominated cluster in $\mathcal{C}_K$ and Case 1 applies. □

With Lemma 5.4, we can now show that if the input graph G is already P_3-free, we can solve the instance in polynomial time.

Lemma 5.5. *Let $I \coloneqq (G, g, k)$ be an instance of* SBCD*, such that G is a cluster graph. Then, I can be solved in $O(n + m)$ time.*

Proof. According to Observation 3, every cluster of G can be solved independently. A cluster which contains exactly one black or exactly one white vertex is already valid. Hence, we only have to consider components that contain at least two black and two white vertices.

Let K be a connected component of G with $b_K \coloneqq |B(K)| > 1$ black and $w_K \coloneqq |W(K)| > 1$ white vertices.

According to Lemma 5.4, we can assume that a minimum-cardinality solution S^* splits K into exactly two clusters C_B and C_W in G_{S^*}, such that C_B is a valid black-dominated cluster and C_W is a valid white-dominated cluster.

It remains to show that the minimal set of edge deletions that splits K into C_B and C_W can be determined in polynomial time.

According to Observation 1 splitting K into two clusters C_1 and C_2 needs $|C_1| \cdot |C_2|$ edge deletions and this number is minimal if, without loss of generality, $|C_1|$ is as small as possible or in other words $|C_2|$ is as big as possible. The optimal partition of K into C_B and C_W depends on the values of b_K and w_K.

Case 1: $b_K > 2, w_K > 2$. Let b_1 be a black and w_1 be a white vertex in K. Since C_B must be a valid black-dominated cluster, it must contain exactly one white vertex and C_W must contain all the other white vertices. Also, since C_W must be a valid white-dominated cluster, it must contain exactly one black vertex and C_B must contain all the other black vertices. Therefore, splitting K into $C_B \coloneqq (B(K) \setminus \{b_1\}) \cup \{w_1\}$ and $C_W \coloneqq (W(K) \setminus \{w_1\}) \cup \{b_1\}$ is optimal.

Case 2: $b_K = 2$. Let $B(K) \coloneqq \{b_1, b_2\}$. Since a single black vertex is also a valid black-dominated cluster, splitting K into $C_B \coloneqq \{b_1\}$ and $C_W \coloneqq W(K) \cup \{b_2\}$ is optimal.

Case 3: $w_K = 2$. Let $W(K) \coloneqq \{w_1, w_2\}$. Since a single white vertex is also a valid white-dominated cluster, splitting K into $C_W \coloneqq \{w_1\}$ and $C_B \coloneqq B(K) \cup \{w_2\}$ is optimal.

Given b_K and w_K we can therefore determine the minimal amount of edge deletion involving vertices in K in constant time.

Finding every connected component K and determining the corresponding values b_K and w_K can be done in $O(n + m)$ time by using a modified breadth-first-search. Constructing the optimal set of edge deletions according to the values of b_K and w_K can then also be done in $O(n + m)$ time. $\square$

Using Lemma 5.5, we get a **FPT**-algorithm for SBCD by resolving all induced P_3s by branching via Branching Rule 1 and then solving the resulting cluster graph as described in Lemma 5.5. The following algorithm is initially called with an edge modification set $S := \emptyset$.

Algorithm 3:

Input: A graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$, an integer k and an edge modification set S.
Output: A valid solution S^*, if one exists.
if $k < 0$ **then**
    Return to the parent node in the search tree;
else
    Search for an induced P_3 in G;
    if *an induced* P_3 *in* G *was found* **then**
        Apply Branching Rule 1;
    else
        Solve each connected component K according to Lemma 5.5;
        if $k \geq 0$ **then**
            Return the solution set S;
        end
    end
end

Theorem 5.6. SBCD *is in* **FPT** *and can be solved in* $O(2^k \cdot (n+m))$ *time.*

Proof. Algorithm 3 first branches over all possibilities to resolve a given induced P_3. This is correct, since, if P is an induced P_3 in G, every optimal solution S^* must resolve P in order to yield a strict bicolored cluster graph. Resolving all induced P_3s results in a cluster graph that then can be solved according to Lemma 5.5.

In each node of the search tree finding an induced P_3 can be done in $O(n + m)$ time. If no induced P_3 can be found, according to Lemma 5.5 the given instance in that node of the search tree can be solved in $O(n + m)$ time. Since Branching Rule 1 admits the branching number 2, the size of the search tree is bounded by 2^k. The total worst-case running-time of the algorithm is therefore $O(2^k \cdot (n + m))$. □

5.2 Problem Kernels

5.2.1 Bicolored Cluster Deletion

Now we present a linear-vertex kernel for BCD that makes use of the following observation regarding critical cliques in a resulting bicolored cluster graph. Recall that a critical clique C is a clique, such that every vertex $v \in C$ has the same neighborhood and C is maximal under this property. A closed critical clique is a critical clique C such that $C \cup N(C)$ is also a clique.

Lemma 5.7. *Let* (G, g, k) *be an instance of* BCD *or* SBCD *and let* S *be an optimal solution. Let* v *be an unaffected vertex and let* K *be the critical clique in* G *containing* v*. Then* K *is a closed critical clique, the valid cluster containing* v *in* G_S *is* $K \cup N(K)$*, and every vertex in* K *is unaffected.*

Proof. First, we show that K is closed. Assume that for $u, w \in N(K)$ we have $\{u, w\} \notin E$. Since only edge deletions are allowed, this implies that u and w can not be in the same resulting cluster in G_S and therefore at least one of the edges $\{v, u\}$ and $\{v, w\}$ is deleted by S, which would contradict that v is unaffected.

Now consider the valid cluster C in G_S containing v. Since v is unaffected, no edge adjacent to v is deleted by S, so $N[v] = K \cup N(K) \subseteq C$. Because edge insertions are not allowed, no vertex $w \notin N[v] = K \cup N(K)$ can be in the same resulting cluster as v. Hence $C \subseteq K \cup N(K)$ and therefore $C = K \cup N(K)$.

Let u be a vertex in K. Since K is a critical clique we have $N[u] = K \cup N(K) = C$. As u belongs to the valid cluster C in G_S, no edge modification is incident to u, hence u is unaffected. □

We now formulate reduction rules that give rise to a linear-vertex kernel for BCD. Similar to previous kernels obtained for Cluster Editing [12, 7] the idea is to bound the number of affected and unaffected vertices in the input graph of a yes-instance using the concept of critical cliques.

The first reduction rule removes already valid clusters, since those are not affected by any edge deletions.

Reduction Rule 1. *Remove a valid cluster in* G*.*

Lemma 5.8. *Reduction Rule 1 is correct and can be exhaustively applied in* $O(n + m)$ *time.*

Proof. Let (G, g, k) be an instance of BCD and C a valid cluster in G. Let S be a solution such that there are two clusters $C_1 \subseteq C$, $C_2 \subseteq C$ with $C_1 \neq C_2$ in G_S. Consider the edge modification set $S' := S \setminus \{\{u, v\} \mid u \in C_1, v \in C_2\}$ that does not separate C_1 and C_2. Clearly, $|S'| < |S|$. Since C was already a valid cluster, S' is also a solution. Therefore an optimal solution S^* does not delete any edges between vertices in C and C can be safely removed from G.

Using a modified breadth-first search every connected component can be computed and checked for validity in $O(n + m)$ time. □

The next reduction rule bounds the number of edges between critical cliques in the resulting input graph.

Reduction Rule 2. *Let K be a closed critical clique in G such that $K \cup N(K)$ forms a valid cluster and $|K| > |E(N(K), N^2(K))|$. Delete every edge in $E(N(K), N^2(K))$, delete $K \cup N(K)$ and reduce k by $|E(N(K), N^2(K))|$.*

To prove the correctness of Reduction Rule 2 we make use of the following claim.

Claim 1. *Let (G, g, k) be an instance of* BCD*. Let K be a closed critical clique in G such that $K \cup N(K)$ forms a valid cluster. Then, for every optimal solution S^* every vertex in K is part of the same resulting cluster in G_{S^*}.*

Proof. Let S be a solution such that there are two clusters C_1, C_2 with $C_1 \subseteq K \cup N(K)$, $C_2 \subseteq K \cup N(K)$ and $C_1 \neq C_2$ in G_S. Consider the edge modification set $S' := S \setminus \{\{u, v\} \mid u \in C_1, v \in C_2\}$ that does not separate C_1 and C_2. Clearly, $|S'| < |S|$. Since K is a closed critical clique and $K \cup N(K)$ is already a valid cluster, $C_1 \cup C_2$ is also a valid cluster and S' is a solution. Therefore an optimal solution S^* does not delete any edges between vertices in K and every vertex in K is part of the same resulting cluster in G_{S^*}. □

Lemma 5.9. *Reduction Rule 2 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of BCD. Let K be a closed critical clique that meets the requirements in Reduction Rule 2. First, note that for any optimal solution S^* every vertex in K is part of the same cluster $C_K \subseteq K \cup N(K)$ in G_{S^*} according to Claim 1.

Let S be a solution. Assume that $C_K \subsetneq K \cup N(K)$ in G_S. This implies that G_S contains a cluster $C' \neq C_K$ with $C' \cap N(K) \neq \emptyset$. Let $V' := C' \cap N(K)$.

Consider the edge modification set $S' := S \setminus E(V', C_K) \cup E(V', C' \setminus V')$ that leaves V' in the same cluster as K. We proceed to show that S' deletes less edges than S and is also a solution.

As $K \subseteq C_K$, we have $|E(V', C_K)| \geq |K|$. Since $V' \subseteq N(K)$, clearly $E(V', C' \setminus V') \subseteq E(N(K), N^2(K))$ and thus $|E(V', C' \setminus V')| \leq |E(N(K), N^2(K))|$. With $|K| > |E(N(K), N^2(K))|$, we get $|S'| < |S|$.

Since the bicolored cluster property is hereditary by Lemma 3.1, $C_{V'} \setminus V'$ is still a valid cluster and $C_K \cup V' \subseteq K \cup N(K)$ is also a valid cluster, so S' is a solution.

Overall, this implies that for every optimal solution S^* in the resulting cluster graph $K \cup N(K)$ forms a cluster and every edge in $E(N(K), N^2(K))$ is deleted by S^*.

For a given graph G all critical cliques can be determined in $O(n+m)$ time. Using a modified breadth-first search the critical cliques can be checked for validity and the edges between critical cliques can be determined in $O(n+m)$ time. Every application of Reduction Rule 2 deletes at least one vertex, hence the rule must be applied at most n times. Therefore, Reduction Rule 2 can be exhaustively applied in $O(n^2 + n \cdot m)$ time. □

Lemma 5.10. *Let $I \coloneqq (G, g, k)$ be exhaustively reduced with respect to Reduction Rules 1 and 2. If G has more than $4k$ vertices, then I is a no-instance of* BCD*.*

Proof. Let I be a yes-instance of BCD and let S be a valid solution for I. We prove the lemma by giving an upper bound on the number of vertices in G that are affected and unaffected by S. Let V_α denote the vertices affected by S and V_β denote the vertices that are unaffected by S, $V_\alpha \dot\cup V_\beta = V$. Clearly, $|V_\alpha| \leq 2k$, since S is valid and every edge in S is incident with at most two unique vertices. According to Lemma 5.7 every unaffected vertex $v \in V_\beta$ is part of a closed critical clique K in G, such that every vertex in K is unaffected and $K \cup N(K)$ is a valid cluster in G_S. Let $K_1, K_2, \dots K_r, r \geq 0$ denote the closed critical cliques in G that contain the unaffected vertices. Since $K_i \cup N(K_i)$ is a valid cluster in G_S, every edge in $E(N(K_i), N^2(K_i))$ must have been deleted by S. For two indices i and j, $i \neq j$, an edge in $E(N(K_i), N^2(K_i))$ can have an endpoint in $N(K_j)$ and thus also be included in $E(N(K_j), N^2(K_j))$. Since S is valid we therefore have

$$\sum_{i=1}^{r} |E(N(K_i), N^2(K_i))| \leq 2k.$$

Since G is reduced with respect to Reduction Rule 1 and 2, for every remaining closed critical clique K_i we also have that $|K_i| \leq |E(N(K_i), N^2(K_i))|$. Thus, in total we get

$$|V_\beta| = \sum_{i=1}^{r} |K_i| \leq \sum_{i=1}^{r} |E(N(K_i), N^2(K_i))| \leq 2k$$

and finally

$$|V| = |V_\alpha| + |V_\beta| \leq 2k + 2k = 4k.$$

□

Theorem 5.11. BCD *admits a $4k$-vertex kernel that can be computed in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of BCD. The kernelization algorithm for BCD first exhaustively applies Reduction Rules 1 and 2. Then, if for the resulting graph G' we have $|V(G')| > 4k$, the algorithm returns a trivial no-instance.

Exhaustively applying Reduction Rules 1 and 2 takes $O(n^2 + n \cdot m)$ time. The correctness of the kernelization algorithm follows from Lemma 5.10. $\square$

5.2.2 Strict Bicolored Cluster Deletion

Now we proceed to present a problem kernel for SBCD, again using the notion of critical cliques to bound the number of affected and unaffected vertices in the input graph of a yes-instance. Recall that for SBCD a valid cluster of size at least two is a cluster that is $K_{(2,2)}$-free and is not a monochromatic cluster. Singletons are also considered a valid cluster. Recall that Lemma 5.7 also holds for SBCD, only the notion of a valid cluster is slightly different.

The first reduction rule again removes already valid clusters.

Reduction Rule 3. *Remove a valid cluster in G.*

Lemma 5.12. *Reduction Rule 3 is correct and can be exhaustively applied in $O(n + m)$ time.*

Proof. Let (G, g, k) be an instance of SBCD and C a valid cluster in G. Note that, since C is a valid cluster, C contains exactly one black or exactly one white vertex. Without loss of generality, assume that $|B(C)| = 1$ and let b denote the black vertex in C. This also implies that every vertex in $C \setminus \{b\}$ is white. Let S be a solution such that there are two valid clusters $C_1 \subseteq C$, $C_2 \subseteq C$ with $C_1 \neq C_2$ and, without loss of generality, assume $b \in C_1$ in G_S. Note that this also implies that C_2 consists of a single white vertex $w \in W(C)$. Consider the edge modification set $S' := S \setminus \{\{u, w\} \mid u \in C_1\}$ that does not separate C_1 and C_2. Clearly, $|S'| < |S|$. Since C is already a valid cluster and C_1 contains b, $C_1 \cup C_2 \subseteq C$ is also a valid cluster and S' is a solution. Therefore, an optimal solution S^* does not delete any edges between vertices in C. Thus, C can be safely removed from G.

Using a modified breadth-first search, every connected component can be computed and checked for validity in $O(n + m)$ time. $\square$

Reduction Rule 4 upper-bounds the size of closed critical cliques. This later contributes to bounding the number of unaffected vertices in a reduced instance.

Reduction Rule 4. *Let K be a closed critical clique in G with $|K| \geq k + 2$. Then, if $K \cup N(K)$ forms a valid cluster, delete $K \cup N(K)$ and reduce k by $|E(N(K), N^2(K))|$. Otherwise, return a trivial no-instance.*

Lemma 5.13. *Reduction Rule 4 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Since $|K| \geq k+2$, according to Observation 1 splitting off any clique $K' \subset K \cup N(K)$ from the other vertices in $K \cup N(K)$ would require at least $|K| - 1 \geq k+1$ edge deletions.

Therefore, if $K \cup N(K)$ forms a valid cluster, every valid solution S must contain the cluster $K \cup N(K)$ in G_S and $K \cup N(K)$ can be safely deleted. Otherwise, at least one vertex in $K \cup N(K)$ must be separated from the others. Since this requires at least $k+1$ edge deletions, (G, g, k) is a no-instance.

For a given graph G all critical cliques can be determined in $O(n+m)$ time. Using a modified breadth-first search the critical cliques can be checked for validity and the edges between critical cliques can be determined in $O(n+m)$ time.

Since every application of Reduction Rule 4 either returns a trivial no-instance or deletes at least one vertex, the rule can be applied at most n times. Therefore, Reduction Rule 4 can be exhaustively applied in $O(n^2 + n \cdot m)$ time. $\square$

For SBCD we can also formulate a claim similar to Claim 1.

Claim 2. *Let (G, g, k) be an instance of* SBCD. *Let K be a closed critical clique in G such that $K \cup N(K)$ forms a valid cluster. Then, for every optimal solution S^* either every vertex in K is part of the same resulting cluster in G_{S^*} or every vertex in K is a singleton in G_{S^*}.*

Proof. Since $K \cup N(K)$ is a valid cluster, we have $|B(K \cup N(K))| = 1$ or $|W(K \cup N(K))| = 1$. Without loss of generality, assume that $|B(K \cup N(K))| = 1$ and let b denote the black vertex in $K \cup N(K)$. This also implies that every vertex in $(K \cup N(K)) \setminus \{b\}$ is white. We consider two cases, depending on whether $b \in K$ or $b \in N(K)$.

Case 1: First, let $b \in K$. Let S be a solution such that there are two valid clusters C_1 with $C_1 \subseteq K \cup N(K), C_1 \cap K \neq \emptyset$, and C_2 with $C_2 \subseteq K \cup N(K), C_2 \cap K \neq \emptyset$, such that $C_1 \neq C_2$ in G_S and, without loss of generality, $b \in C_1$. Note that this also implies that C_2 consists of a single white vertex w, because C_2 does not contain any black vertices and must therefore be a singleton in G_S. Consider the edge modification set $S' := S \setminus \{\{u, w\} \mid u \in C_1\}$ that does not separate C_1 and C_2. Clearly, $|S'| < |S|$. Since $K \cup N(K)$ is already a valid cluster and C_1 contains b, $C_1 \cup C_2 \subseteq K \cup N(K)$ is also a valid cluster and S' is a solution. Therefore an optimal solution S^* does not delete any edges between vertices in K and every vertex in K is part of the same resulting cluster in G_{S^*}.

Case 2: Now, let $b \in N(K)$. Note that in a resulting cluster graph the cluster containing b cannot contain a vertex $v \in K$ and a vertex $u \in N^2(K)$ at the same time, since per definition $\{v, u\} \notin E$ and edge insertions are not allowed. Therefore, for every solution S the cluster C_b containing b is a subset of $K \cup N(K)$ or a subset of $N(K) \cup N^2(K)$.

Case 2.1: Let S_1 be a solution such that the cluster containing b in G_{S_1} is $C_b \subseteq N(K) \cup N^2(K)$. Then, since $(K \cup N(K)) \setminus \{b\}$ does not contain any black vertex, every vertex $v \in K$ cannot be contained in a valid cluster with another vertex $u \in K \cup N(K)$ and is therefore a singleton in G_{S_1}.

Case 2.2: Let S_2 be a solution, such that the cluster containing b in G_{S_2} is $C_b \subseteq K \cup N(K)$. Let C' with $C' \subseteq K \cup N(K)$ be another cluster in G_{S_2}. Note that C' consists of a single white vertex w, because C' does not contain any black vertices.

Consider the edge modification set $S_2' \coloneqq S_2 \setminus \{\{u, w\} \mid u \in C_b\}$ that does not separate C_b and C'. Clearly, $|S_2'| < |S_2|$. Since $K \cup N(K)$ is already a valid cluster and C_b contains b, $C_b \cup C' \subseteq K \cup N(K)$ is also a valid cluster and S_2' is a solution.

In any case, for an optimal solution S^* either every vertex in K is part of the same resulting cluster in G_{S^*} or every vertex in K is a singleton in G_{S^*}. $\square$

Similar to Reduction Rule 2, Reduction Rule 5 aims to bound the number of edges between critical cliques in the resulting input graph, using that the size of critical cliques is already bounded according to Reduction Rule 4.

Reduction Rule 5. *Let K be a closed critical clique in G such that $K \cup N(K)$ forms a valid cluster and $|K| > |E(N(K), N^2(K))| \cdot k^{\frac{1}{2}}$. Delete $K \cup N(K)$ and reduce k by $|E(N(K), N^2(K))|$.*

Lemma 5.14. *Let $I \coloneqq (G, g, k)$ be an instance of* SBCD *such that I is exhaustively reduced with respect to Reduction Rule 4 and $k \geq 16$. Then Reduction Rule 5 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Let K be a closed critical clique in G that meets the requirements of Reduction Rule 5. First, note that for any optimal solution S^* every vertex in K is part of the same cluster $C_K \subseteq K \cup N(K)$, or every vertex in K is a singleton in G_{S^*} according to Claim 2.

Let S be a solution, such that $K \cup N(K)$ is not a cluster in G_S. Since edge insertions are not allowed, every cluster in G_S containing a vertex in $K \cup N(K)$ is a subset of $K \cup N(K)$ or a subset of $N(K) \cup N^2(K)$. Let $C_1, C_2, \ldots, C_r$, $r > 0$, denote all the clusters in G_S that are subsets of $K \cup$

$N(K)$, including singletons. Let $C'_1, C'_2, \dots, C'_\ell$, $\ell \geq 0$, denote all the clusters in G_S that are subsets of $N(K) \cup N^2(K)$ and contain at least one vertex of $N(K)$ and at least one vertex of $N^2(K)$.

We define four sets of edges in order to obtain a better solution. The sets

$$E_1 \coloneqq \bigcup_{1 \leq i < j \leq r} E(C_i, C_j),$$

$$E_2 \coloneqq \bigcup_{\substack{1 \leq i \leq r \\ 1 \leq p \leq \ell}} E(C_i, C'_p \cap N(K)),$$

and

$$E_3 \coloneqq \bigcup_{1 \leq p < q \leq \ell} E(C'_p \cap N(K), C'_q \cap N(K))$$

contain all the edges between vertices in $K \cup N(K)$ that are deleted by S. The set

$$E_4 \coloneqq \bigcup_{1 \leq p \leq \ell} E(C'_p, C'_p \cap N^2(K))$$

contains all edges that have to be additionally deleted in order to make every vertex in $C'_p \cap N^2(K)$, $1 \leq p \leq \ell$ a singleton.

Case 1: Let $\ell = 0$. This implies that every vertex in $K \cup N(K)$ is contained in one of the clusters C_i, $1 \leq i \leq r$, in G_S. Consider the edge modification set

$$S' \coloneqq S \setminus E_1$$

that instead preserves $K \cup N(K)$ as a cluster. Clearly, $|S'| < |S|$. In $G_{S'}$ every vertex from C_i, $1 \leq i \leq r$, is now part of the cluster $K \cup N(K)$ and every other cluster in G_S remains the same in $G_{S'}$. Since $K \cup N(K)$ is a valid cluster and S is a solution, thus S' is also a solution.

Case 2: Let $\ell \geq 1$. Consider the edge modification set

$$S' \coloneqq \big(S \setminus \big(E_1 \cup E_2 \cup E_3\big)\big) \cup E_4$$

that instead preserves $K \cup N(K)$ as a cluster and leaves every vertex in $C'_p \cap N^2(K)$, $1 \leq p \leq \ell$, as a singleton. We proceed to show that S' deletes fewer edges than S and is also a solution.

Observe that, since $|K| > |E(N(K), N^2(K))| \cdot k^{\frac{1}{2}}$ and I is exhaustively reduced with respect to Reduction Rule 4, we have

$$k + 1 \geq |K| > |E(N(K), N^2(K))| \cdot k^{\frac{1}{2}}$$

which implies

$$|E(N(K), N^2(K))| < k^{\frac{1}{2}} + 1 \tag{1}$$

Let v be a vertex in $C'_1 \cap N(K)$. Since $K \cup N(K)$ is a clique in G, the vertex v is adjacent to every vertex $u \in K$ in G and thus $\bigcup_{1 \le i \le r} E(C_i, C'_1 \cap N(K))$ contains an edge between v and every vertex u in K. Hence, for the number of edges in $E_2 = \bigcup_{\substack{1 \le i \le r \\ 1 \le p \le \ell}} E(C_i, C'_p \cap N(K))$ we have

$$|E_2| \ge |K|. \tag{2}$$

Furthermore, we have

$$E_4 = \bigcup_{1 \le p \le \ell} E(C'_p, C'_p \cap N^2(K)) \subseteq E(N(K), N^2(K)) \cup N^2(K). \tag{3}$$

Since every vertex in $N^2(K)$ has at least one neighbor in $N(K)$, we have $|N^2(K)| \le |E(N(K), N^2(K))|$ and therefore the edges between two vertices in $N^2(K)$ can be bound by $\binom{|E(N(K),N^2(K))|}{2}$. Setting $x := |E(N(K), N^2(K))|$ we get the following inequalities that hold for every $k \ge 16$:

$$\begin{aligned}
|E_4| &= |\bigcup_{1 \le p \le \ell} E(C'_p, C'_p \cap N^2(K))| \\
&\overset{(3)}{\le} x + \binom{x}{2} \le x \cdot \left(1 + \frac{x}{2}\right) \\
&\overset{(1)}{<} x \cdot \left(1 + \frac{k^{1/2}+1}{2}\right) < x \cdot \left(\frac{k^{1/2}}{2} + 2\right) \\
&\le x \cdot k^{1/2} && \text{since } k \ge 16 \\
&< |K| \\
&\overset{(2)}{\le} |E_2|.
\end{aligned}$$

This gives us $|E_4| < |E_2|$ which implies that $|S'| < |S|$.

Now we show that S' is a solution. Note that every cluster in G_S except C_i, C'_p, $1 \le i \le r, 1 \le p \le \ell$, is also a cluster in $G_{S'}$. By construction of S' every vertex from C_i, C'_p, $1 \le i \le r, 1 \le p \le \ell$ is either a singleton or part of the valid cluster $K \cup N(K)$ in $G_{S'}$. Therefore every vertex in $G_{S'}$ is part of a valid cluster and S' is a solution.

Overall, this implies that for every optimal solution S^* in the resulting cluster graph G_{S^*} the vertex set $K \cup N(K)$ forms a cluster and every edge in $E(N(K), N^2(K))$ is deleted by S^*.

For a given graph G all critical cliques can be determined in $O(n+m)$ time. Using a modified breadth-first search the critical cliques can be checked for validity and the edges between critical cliques can be determined in $O(n+m)$

time. Every application of Reduction Rule 5 deletes at least one vertex, hence the rule must be applied at most n times. Therefore, Reduction Rule 5 can be exhaustively applied in $O(n^2 + n \cdot m)$ time. $\square$

Lemma 5.15. *Let $I \coloneqq (G, g, k)$ be exhaustively reduced with respect to Reduction Rules 3, 4 and 5 and $k \geq 16$. If G has more than $2k^{\frac{3}{2}} + 2k$ vertices, then I is a no-instance.*

Proof. Let I be a yes-instance of SBCD and let S be a valid solution for I. We prove the lemma by giving an upper bound on the number of vertices in G that are affected and unaffected by S. Let V_α denote the vertices affected by S and V_β denote the vertices that are unaffected by S, recall that $V_\alpha \dot\cup V_\beta = V$. Clearly, $|V_\alpha| \leq 2k$, since S is valid and every edge in S is incident to at most two unique vertices. According to Lemma 5.7 every unaffected vertex $v \in V_\beta$ is part of a closed critical clique K in G, such that every vertex in K is unaffected and $K \cup N(K)$ is a valid cluster in G_S. Let $K_1, K_2, \ldots, K_r, r \geq 0$ denote the closed critical cliques in G that contain the unaffected vertices. Since $K_i \cup N(K_i)$ is a valid cluster in G_S, every edge in $E(N(K_i), N^2(K_i))$ is deleted by S. For two indices i and j, $i \neq j$, an edge in $E(N(K_i), N^2(K_i))$ can have an endpoint in $N(K_j)$ and thus also be included in $E(N(K_j), N^2(K_j))$.

Since S is valid we therefore have

$$\sum_{i=1}^{r} |E(N(K_i), N^2(K_i))| \leq 2k.$$

Since G is reduced with respect to Reduction Rule 3, 4, and 5 and $k \geq 16$, for every remaining closed critical clique K_i we also have that

$$|K_i| \leq |E(N(K_i), N^2(K_i))| \cdot k^{\frac{1}{2}}.$$

Thus, in total we get

$$|V_\beta| = \sum_{i=1}^{r} |K_i| \leq \sum_{i=1}^{r} |E(N(K_i), N^2(K_i))| \cdot k^{\frac{1}{2}} \leq 2k \cdot k^{\frac{1}{2}} = 2k^{\frac{3}{2}}$$

and finally

$$|V| = |V_\alpha| + |V_\beta| \leq 2k^{\frac{3}{2}} + 2k$$

$\square$

Theorem 5.16. SBCD *admits a $2k^{\frac{3}{2}} + 2k$-vertex kernel that can be computed in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of SBCD. The kernelization algorithm for SBCD first exhaustively applies Reduction Rule 3 and 4 and then checks whether $k \geq 16$. As long as $k \geq 16$, the algorithm tries to apply Reduction Rule 5. If $k \geq 16$ and Reduction Rule 5 has been exhaustively applied, if for the resulting graph G' we have $|V(G')| > 2k^{\frac{3}{2}} + 2k$, the algorithm returns a trivial no-instance. If at any point $k < 16$, the algorithm solves the current instance $I \coloneqq (G', g, k)$ using Algorithm 3 in $O(n + m)$ time. If I is a yes-instance, the kernelization algorithm returns a trivial yes-instance. Otherwise, the algorithm returns a trivial no-instance.

Clearly, the kernelization algorithm is correct for $k < 16$. The correctness of the kernelization algorithm for $k \geq 16$ follows from Lemma 5.15. Exhaustively applying Reduction Rule 3, 4 and 5 can be done in $O(n^2 + n \cdot m)$ time. For $k < 16$ using the **FPT**-algorithm for SBCD takes $O(n + m)$ time. $\square$

6 Editing Variants

In this chapter we proceed to study the parameterized complexity of the editing variants BCE and SBCE for the parameter solution size k. First we again propose an **FPT**-algorithm for both variants and then formulate reduction rules that lead to quadratic-vertex kernels.

6.1 FPT-Algorithms

6.1.1 Bicolored Cluster Editing

As for BCD, for BCE we can also make the observation that we can solve each connected component individually.

Observation 4. *Let (G, g, k) be an instance of* BCE*. Each connected component of G can be solved independently.*

Proof. Let S be a solution and let C be a cluster in G_S that contains vertices from two distinct connected components K_1 and K_2 in G. All edges between vertices in $K_1 \cap C$ and $K_2 \cap C$ are inserted by S. Let $S^* := S \setminus E_{G_S}(K_1 \cap C, K_2 \cap C)$ be a solution that leaves the vertices in $K_1 \cap C$ and $K_2 \cap C$ separated, but is otherwise identical to S. Clearly, we have $|S^*| < |S|$. Since S is a solution and the bicolored cluster property is hereditary, S^* is also a solution.

This shows that every optimal solution does not insert edges between vertices from different connected components and therefore every connected component can be solved independently. □

For BCE we have an **FPT**-algorithm very similar to the one proposed for BCD. Again, we branch over all possibilities to resolve any induced P_3s and then solve the resulting instance in polynomial time.

The following branching rule resolves an induced P_3. Note that in the editing case we can resolve an induced P_3 by either deleting one of the present edges or inserting the missing edge.

Branching Rule 3. *Let $G = (V, E)$ be a graph with a bicoloring $g : V \to \{b, w\}$ and let $k \geq 0$. If G contains an induced P_3, denoted by $P := (v_1, v_2, v_3)$, branch into the cases:*

1. *remove the edge $\{v_1, v_2\}$ from G and decrease k by one;*
2. *remove the edge $\{v_2, v_3\}$ from G and decrease k by one;*
3. *insert the edge $\{v_1, v_3\}$ in G and decrease k by one.*

Since all three branching cases of the rule reduce the parameter k by one, Branching Rule 3 admits the branching vector $(1, 1, 1)$, which has a branching number of $\beta(1, 1, 1) = 3$.

Similar to BCD we can again solve an instance of BCE if the input graph is already a cluster graph.

Lemma 6.1. *Let $I := (G, g, k)$ be an instance of* BCE*, such that G is a cluster graph. Then, I can be solved in $O(n + m)$ time.*

Proof. According to Observation 4 every cluster of G can be solved independently. Since every cluster is already a complete subgraph, this also implies that an optimal solution for I only consists of edge deletions. Therefore, an optimal solution for an instance $I' := (G, g, k)$ of BCD is also an optimal solution for I and we can solve I in $O(n + m)$ time according to Lemma 5.2. □

Using Branching Rule 3 and Lemma 6.1 we can now formulate the **FPT**-algorithm for BCE.

Algorithm 4:

Input: A graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$, an integer k and an edge modification set S.
Output: A valid solution S^*, if one exists.
if $k < 0$ **then**
  Return to the parent node in the search tree;
else
  Search for an induced P_3 in G;
  if *an induced P_3 in G was found* **then**
    Apply Branching Rule 3;
  else
    Solve each connected component K according to Lemma 6.1;
    if $k \geq 0$ **then**
      Return the solution set S;
    end
  end
end

Theorem 6.2. BCE *is in* FPT *and can be solved in $O(3^k \cdot O(n+m))$ time.*

Proof. Algorithm 4 first branches over all possibilities to resolve a given induced P_3. This is correct, since, if P is an induced P_3 in G, every optimal solution S^* must resolve P in order to yield a bicolored cluster graph. In each node of the search tree finding an induced P_3 can be done in $O(n + m)$

time. If no induced P_3 can be found, according to Lemma 6.1 the given instance in that node of the search tree can be solved in $O(n+m)$ time. Since Branching Rule 3 admits the branching number 3, the size of the search tree is bounded by 3^k. The total worst-case running-time of the algorithm is therefore $O(3^k \cdot (n+m))$. $\square$

6.1.2 Strict Bicolored Cluster Editing

Before we proceed to propose an **FPT**-algorithm for SBCE, we first formulate a series of lemmas that we will use to design both the **FPT**-algorithm and the problem kernel for SBCE. The first two lemmas describe what happens to the vertices of a monochromatic cluster in G when an optimal solution is applied.

Lemma 6.3. *Let C be a monochromatic black cluster in G. Let S^* be an optimal solution. Then, there is at most one cluster C_B in G_{S^*} that contains two or more vertices from C. Furthermore, every vertex $v \in C$ is in G_{S^*} either*

- *the single black vertex of a white-dominated cluster C_W,*
- *part of the black-dominated cluster C_B with $B(C_B) \subseteq C$, or*
- *a singleton.*

Proof. First, assume that there is a black-dominated cluster C_1 in G_{S^*} such that C_1 contains at least one vertex from C and $B(C_1) \nsubseteq C$. Let w_{C_1} denote the single white vertex in C_1 and $\overline{C_1} := C_1 \setminus C$, $C_1' := C_1 \cap C$.

Case 1: $|C_1'| \leq |\overline{C_1}|$.

Every edge between C_1' and $\overline{C_1}$ is inserted by S^*. Therefore we can get a better solution S' by instead leaving $\overline{C_1}$ as a valid black-dominated cluster and breaking up C_1' into singletons. This needs $\binom{|C_1'|}{2}$ additional edge deletions but also saves $|C_1'| \cdot |\overline{C_1}|$ edge insertions.

Case 2: $|C_1'| > |\overline{C_1}|$.

Every edge between $\overline{C_1} \setminus \{w_{C_1}\}$ and C_1' is inserted by S^*. Therefore we can get a better solution S' by instead leaving $C_1' \cup \{w_{C_1}\}$ as a valid black-dominated cluster and breaking up $\overline{C_1} \setminus \{w_{C_1}\}$ into singletons. This needs at most $\binom{|\overline{C_1}|}{2}$ additional edge deletions but also saves $|C_1'| \cdot (|\overline{C_1}| - 1)$ edge insertions.

We can now assume that for every black-dominated cluster C_1 in G_{S^*} either C_1 contains no vertices from C or $B(C_1) \subseteq C$. Now let C_1, C_2 be two black-dominated clusters with $|C_1 \cap C| \geq 2, |C_2 \cap C| \geq 2$ and let w_{C_2} denote the single white vertex in C_2. We can then get a better solution S' that merges $C_2 \setminus \{w_{C_2}\}$ with C_1 and leaves w_{C_2} as a singleton. This saves the edge deletions applied by S^* to separate $C_1 \cap C$ and $C_2 \cap C$ as well as the edges inserted between $C_2 \cap C$ and w_{C_2}.

Therefore G_{S^*} contains at most one cluster with two or more vertices from C and the lemma holds, since otherwise we can find a better solution S' with $|S'| < |S^*|$, which contradicts the optimality of S^*. □

For monochromatic white clusters we have a lemma analogous to Lemma 6.3.

Lemma 6.4. *Let C be a monochromatic white cluster in G. Let S^* be an optimal solution. Then there is at most one cluster C_W in G_{S^*} that contains two or more vertices from C. Furthermore, every vertex $v \in C$ is in G_{S^*} either*

- *the single white vertex of a black-dominated cluster C_B,*
- *part of the white-dominated cluster C_W with $W(C_W) \subseteq C$, or*
- *a singleton.*

Proof. The proof is analogous to that of Lemma 6.3. □

The next two lemmas handle valid black-dominated and valid white dominated clusters in G.

Lemma 6.5. *Let C be a valid black-dominated cluster in G and let w be the single white vertex in C. Let S^* be an optimal solution. Then there is at most one cluster C^* in G_{S^*} that contains two or more vertices from C and C^* is a black-dominated cluster with $C^* \subseteq C$. Furthermore, in G_{S^*} the vertex w is*

- *the single white vertex of a black-dominated cluster C_B with $C_B \cap C = \{w\}$,*
- *part of the black-dominated cluster $C^* \subseteq C$, or*
- *a singleton*

and for every vertex $v \in C \setminus \{w\}$ in G_{S^*} we have that v is

- *the single black vertex of a white-dominated cluster C_W,*
- *part of the black-dominated cluster $C^* \subseteq C$, or*
- *a singleton.*

Proof. We denote with $\mathcal{C}^B$ the set of black-dominated clusters in G_{S^*} and with $\mathcal{C}^W$ the set of white-dominated clusters in G_{S^*}.

Let C_w be the cluster in G_{S^*} that contains the vertex w. First we show that either $C_w \subseteq C$ or $C_w \in \mathcal{C}^B$ with $C_w \cap C = \{w\}$.

Case 1: $|C_w \cap C| \geq 2$ and $C_w \nsubseteq C$.

We have to distinguish whether $C_w \in \mathcal{C}^B$ or $C_w \in \mathcal{C}^W$.

Case 1.1: $C_w \in \mathcal{C}^B$.

Let $C'_w := C_w \setminus C$ and $\overline{C_w} := C_w \cap C$.

Case 1.1a: $|\overline{C_w}| \leq |C'_w|$.

Every edge between C'_w and $\overline{C_w} \setminus \{w\}$ is inserted by S^*. Therefore, we can find a better solution S' that instead leaves $C'_w \cup \{w\}$ as a cluster and splits up $\overline{C_w} \setminus \{w\}$ into singletons. This requires $\binom{|\overline{C_w}|}{2}$ additional edge deletions but also saves $(|\overline{C_w}| - 1) \cdot |C'_w|$ edge insertions. In $G_{S'}$ then the cluster containing w is $C'_w \cup \{w\} \in \mathcal{C}^B$ with $(C'_w \cup \{w\}) \cap C = \{w\}$.

Case 1.1b: $|\overline{C_w}| > |C'_w|$.

Every edge between C'_w and $\overline{C_w}$ is inserted by S^*. Therefore, we can find a better solution S' that instead leaves $\overline{C_w}$ as a cluster and splits up C'_w into singletons. This requires at most $\binom{|C'_w|}{2}$ additional edge deletions but also saves $|\overline{C_w}| \cdot |C'_w|$ edge insertions. In $G_{S'}$ then the cluster containing w is $\overline{C_w} \subseteq C$.

Case 1.2: $C_w \in \mathcal{C}^W$.

Let b be the single black vertex in C_w. Since C is a black-dominated cluster and $|C_w \cap C| \geq 2$, we have that $C_w \cap C = \{b, w\}$. Let $C'_w := C_w \setminus C$. If $|C'_w| = 1$ we can get a better solution S' by leaving the single vertex in C'_w as a singleton and not inserting the edges to b and w. If $|C'_w| \geq 2$, we can get a better solution S' that leaves $C'_w \cup \{b\}$ as a cluster and w as a singleton. This requires the additional deletion of the edge $\{b, w\}$, but also saves the $|C'_w|$ edge insertions between C'_w and w.

Case 2: $|C_w \cap C| = 1$.

We have to again distinguish whether $C_w \in \mathcal{C}^B$ or $C_w \in \mathcal{C}^W$.

Case 2.1: $C_w \in \mathcal{C}^B$.

In this case we already have that $C_w \in \mathcal{C}^B$ with $C_w \cap C = \{w\}$.

Case 2.2: $C_w \in \mathcal{C}^W$.

In this case we can find a better solution S' by leaving $C_w \setminus \{w\}$ as a valid cluster and $\{w\}$ as a singleton. Since every edge between w and the other vertices in C_w is inserted by S^*, this saves $|C_w| - 1$ edge insertions.

From now on, we may assume that in the solution graph G_{S^*} for the cluster C_w, that contains the white vertex $w \in C$, we have either $C_w \subseteq C$ or $C_w \in \mathcal{C}^B$ with $C_w \cap C = \{w\}$. It remains to show that in G_{S^*} every vertex in $C \setminus \{w\}$ is the single black vertex of a white-dominated cluster, part of C_w, or a singleton.

Case 1: $C_w \subseteq C$.

Let $C' \in \mathcal{C}^B$ be another cluster that contains some (black) vertices from C. Let $C_1 \coloneqq C' \cap C$ and $C_2 \coloneqq C' \setminus C$. Note that C_2 contains the single white vertex in C'. We then can get a better solution S' by merging C_1 with C_w and leaving C_2 as a valid black-dominated cluster. This saves $|C_w| \cdot |C_1|$ edge deletions and $|C_1| \cdot |C_2|$ edge insertions.

Case 2: $C_w \in \mathcal{C}^B$ with $C_w \cap C = \{w\}$.

Let $C' \in \mathcal{C}^B$ be another cluster that contains some (black) vertices from C. Let the single white vertex in C' be w'. Let $C_1 \coloneqq C' \cap C$ and $C_2 \coloneqq C' \setminus C$.

Case 2.1: $|C_1| \leq |C_2|$.

We can get a better solution S' by leaving C_2 as a valid cluster and splitting C_1 into singletons. This requires $\binom{|C_1|}{2}$ additional edge deletions, but also saves $|C_1| \cdot |C_2|$ edge insertions.

Case 2.2: $|C_1| \geq |C_w| - 1$.

We can get a better solution S' by leaving $C_1 \cup \{w\}$ as a cluster and merging C_2 with $C_w \setminus \{w\}$. This requires $(|C_w| - 1) \cdot |C_2|$ additional edge insertions, but also saves $|C_1|$ edge deletions and $(|C_w| - 1) + |C_1| \cdot |C_2|$ edge insertions.

Case 2.3: $|C_2| < |C_1| < |C_w| - 1$.

In this case we can get a better solution S' by leaving $C_1 \cup \{w\}$ as a cluster, splitting up C_2 and merging $C_w \setminus \{w\}$ with w'. This requires $\binom{|C_2|}{2}$ additional edge deletions and $|C_w| - 1$ additional edge insertions, but also saves $|C_1|$ edge deletions and $(|C_w| - 1) + |C_1| \cdot |C_2|$ edge insertions.

This shows that a black vertex $v \in C \setminus \{w\}$ is in G_{S^*} either the single black vertex of a white-dominated cluster C_W, in the same cluster $C_w \subseteq C$ as w, or a singleton. □

For valid white-dominated clusters we have a lemma analogous to Lemma 6.5.

Lemma 6.6. *Let C be a valid white-dominated cluster in G and let b be the single black vertex in C. Let S^* be an optimal solution. Then there is at most one cluster C^* in G_{S^*} that contains two or more vertices from C and C^* is a white-dominated cluster with $C^* \subseteq C$. Furthermore, in G_{S^*} the vertex b is either*

- *the single black vertex of a white-dominated cluster C_W with $C_W \cap C = \{b\}$,*
- *part of the white-dominated cluster $C^* \subseteq C$, or*
- *a singleton*

and for every vertex $v \in C \setminus \{b\}$ in G_{S^*} we have that v is either

- *the single white vertex of a black-dominated cluster C_B,*
- *part of the white-dominated cluster $C^* \subseteq C$, or*
- *a singleton.*

Proof. The proof is analogous to that of Lemma 6.5. $\square$

For the **FPT**-algorithm for SBCE we can again first branch over all possibilities to resolve a given induced P_3 using Branching Rule 3 and branch over all possibilities to resolve a given induced $K_{(2,2)}$ using Branching Rule 2. After Branching Rules 3 and 2 have been exhaustively applied, the resulting graph is a cluster graph that can contain singletons, valid black- or white-dominated clusters, monochromatic black and monochromatic white clusters. Since for SBCE monochromatic clusters of size at least two are not allowed, they must also be handled.

Let C be a monochromatic black cluster in G. According to Lemma 6.3 for an optimal solution S^* every vertex in C is in G_{S^*} the only vertex from C in its cluster or is part of a valid black-dominated cluster C_B that only contains vertices from C and a white vertex that makes the cluster valid. For monochromatic white clusters in G we have an analogous statement with Lemma 6.4.

The idea of the next branching rule is to branch for a given monochromatic cluster C whether we separate a vertex from C, and potentially more later on, or leave it in its current state and just add a vertex of the opposite color to it. For this we introduce two counters, c_b for black vertices and c_w for white vertices, that tell us how many singletons of the respective color are needed to "fix" the monochromatic clusters from which no more vertices are separated according to the branching rule.

Branching Rule 4. *Let $G = (V, E)$ be a cluster graph with a bicoloring $g : V \to \{b, w\}$ and let $k \geq 0$, $c_b \geq 0$, $c_w \geq 0$.*

If G contains a monochromatic black cluster C_B with $|C_B| \geq 2$, branch into the cases:

1. *separate a vertex from C_B and decrease k by $|C_B| - 1$;*
2. *remove C_B from G, decrease k by $|C_B|$ and increase c_w by one.*

Otherwise, if G contains a monochromatic white cluster C_W with $|C_W| \geq 2$, branch into the cases:

1. *separate a vertex from C_W and decrease k by $|C_W| - 1$;*

2. remove C_W from G, decrease k by $|C_W|$ and increase c_b by one.

If the rule can be applied, it always branches into two cases. In Case 1 at least one edge is deleted and in Case 2 at least two edges are inserted. Thus, Branching Rule 4 admits the branching vector $(1, 2)$, which has a branching number of $\beta(1, 2) \approx 1.62$.

Exhaustively applying Branching Rules 3, 2, and 4 yields a cluster graph G that only contains singletons and valid clusters of size at least two. However, for each monochromatic cluster removed by Branching Rule 4, a singleton of the opposite color is needed. The exact amount of black and white singletons needed is given by the values of c_b and c_w. If there are already at least c_b black and c_w white singletons present in G, we are done. Otherwise, we can first delete all singletons in G and adjust c_b and c_w accordingly. The singletons that are then still needed must be separated from valid clusters in G.

The following lemma motivates the last branching rule. It shows that for each singleton that we need to separate from valid clusters we only have to consider the currently smallest black-dominated and currently smallest white-dominated cluster.

Lemma 6.7. *Let $G = (V, E)$ be a cluster graph with a bicoloring $g : V \to \{b, w\}$ that only contains valid clusters of size at least two. Let $\mathcal{C}^B := \{C_1^B, C_2^B, \dots, C_r^B\}$ be the set of black-dominated and $\mathcal{C}^W := \{C_1^W, C_2^W, \dots, C_q^W\}$ be the set of white-dominated clusters in G. Let $|C_1^B| \leq |C_i^B|$ for $1 \leq i \leq r$ and $|C_1^W| \leq |C_j^W|$ for $1 \leq j \leq q$. Let $c_b \geq 0$, $c_w \geq 0$ be integers.*

Then there is a minimal-cardinality edge-modification set S that creates at least c_b black and c_w white singletons in G_S such that:

- *If $c_b > 0$, then S completely splits up C_1^W or separates a black vertex from C_1^B.*
- *If $c_w > 0$, then S completely splits up C_1^B or separates a white vertex from C_1^W.*

Proof. Let S be a minimal-cardinality edge-modification set that creates at least c_b black and c_w white singletons in G_S. Let $c_b > 0$. First, suppose that S creates a black singleton by completely splitting up a cluster $C_j^W \in \mathcal{C}^W$, $j \neq 1$. This requires

$$\binom{|C_j^W|}{2} = \sum_{\ell=1}^{|C_j^W|-1} \left(|C_j^W| - \ell\right)$$

edge deletions. Since $|C_1^W| \leq |C_j^W|$, we can get another edge-modification set S' by instead splitting up C_1^W and separating $d := |C_j^W| - |C_1^W|$ white vertices from C_j^W. This also requires

$$\binom{|C_1^W|}{2} + \sum_{p=1}^{d} \left(|C_j^W| - p\right) = \sum_{\ell=1}^{|C_1^W|-1} \ell + \sum_{p=1}^{d} \left(|C_j^W| - p\right) = \sum_{\ell=1}^{|C_j^W|-1} \ell = \binom{|C_j^W|}{2}$$

edge deletions and creates the same number of white singletons.

Now, suppose that S creates some black singletons by separating i black vertices from a cluster $C_j^B \in \mathcal{C}^B$ and not separating any vertices from C_1^B. Separating i black vertices from a C_j^B requires $\sum_{p=1}^{i} \left(|C_j^B| - p\right)$ edge deletions.

Case 1: $|C_1^B| \geq i+1$. We can get another edge-modification set S' by instead separating i black vertices from C_1^B. This requires $\sum_{p=1}^{i} \left(|C_1^B| - p\right)$ edge deletions. Since $|C_1^B| \leq |C_j^B|$ we have $|S'| \leq |S|$.

Case 2: $|C_1^B| < i+1$. We can get another edge-modification set S' by instead completely splitting up C_1^B and separating the remaining $d \coloneqq i - (|C_1^B| - 1)$ black vertices from C_j^B. This requires

$$x \coloneqq \sum_{\ell=1}^{|C_1^B|-1} \ell + \sum_{p=1}^{d} \left(|C_j^B| - p\right) \leq \sum_{p=1}^{i} \left(|C_j^B| - p\right)$$

edge deletions and we thus have $|S'| \leq |S|$.

If $c_w > 0$ we can analogously show that there is always a minimum-cardinality edge-modification set S that completely splits up C_1^B or separates a white vertex from C_1^W. □

We can now formulate the last branching rule that leads to an **FPT**-algorithm for SBCE.

Branching Rule 5. *Let $G = (V, E)$ be a cluster graph with a bicoloring $g : V \to \{b, w\}$ that only contains valid clusters and no singletons and let $k \geq 0$ and c_b, c_w be integers. Let C_B^1 be the smallest black-dominated and C_W^1 be the smallest white-dominated cluster in G.*

If $c_b > 0$, branch into the cases:

1. *separate a black vertex from C_B^1, decrease k by $|C_B^1| - 1$ and decrease c_b by one;*
2. *remove C_W^1 from G, decrease k by $\binom{|C_W^1|}{2}$, decrease c_b by one and decrease c_w by $|C_W^1| - 1$.*

Otherwise, if $c_w > 0$, branch into the cases:

1. *separate a white vertex from C_W^1, decrease k by $|C_W^1| - 1$ and decrease c_w by one;*

2. *remove C_B^1 from G, decrease k by $\binom{|C_B^1|}{2}$, decrease c_w by one and decrease c_b by $|C_B^1| - 1$.*

The rule always branches into two cases, each of which deletes at least one edge. Thus, Branching Rule 5 admits the branching vector $(1, 1)$ with branching number $\beta(1, 1) = 2$.

Lemma 6.7 shows the correctness of Branching Rule 5. We now propose an **FPT**-algorithm for SBCE using Branching Rules 3, 2, 4, and 5.

Algorithm 5:

Input: A graph $G = (V, E)$, a bicoloring $g : V \to \{b, w\}$, an integer k, integers c_b and c_w, and an edge modification set S.
Output: A valid solution S^*, if one exists.

```
if k < 0 then
|   Return to the parent node in the search tree;
else
|   Search for an induced P_3 and an induced K_(2,2) in G;
|   if an induced P_3 in G was found then
|   |   Apply Branching Rule 3;
|   else if an induced K_(2,2) in G was found then
|   |   Apply Branching Rule 2;
|   else
|   |   Compute all connected components in G and add them to a
|   |    set C;
|   |   if C contains a monochromatic black or a monochromatic
|   |    white cluster of size ≥ 2 then
|   |   |   Apply Branching Rule 4;
|   |   else
|   |   |   Remove all singletons from G and decrease c_b and c_w
|   |   |    accordingly;
|   |   |   if c_b > 0 or c_w > 0 then
|   |   |   |   Apply Branching Rule 5;
|   |   |   else
|   |   |   |   Return the solution set S;
|   |   |   end
|   |   end
|   end
end
```

Theorem 6.8. SBCE *is in* ***FPT*** *and can be solved in* $O(3^k \cdot (n+m))$ *time.*

Proof. Algorithm 5 first branches over all possibilities to resolve a given induced P_3 or a given induced $K_{(2,2)}$. This results in a cluster graph that contains singletons as well as monochromatic and valid clusters of size at least two. Using Branching Rules 4 and 5 the algorithm then branches over all possibilities to handle a monochromatic cluster according to Lemma 6.3 and get the required amount of singletons from valid clusters.

In each node of the search tree, finding an induced P_3 or an induced $K_{(2,2)}$ can be done in $O(n+m)$ time. If neither can be found, the connected components are computed, which can also be done in $O(n+m)$ time. Checking the set of connected components for a monochromatic cluster, checking the values of c_b and c_w and applying the Branching Rule 4 or 5 can again be done in $O(n+m)$ time. Since Branching Rule 3 has the branching number 3 and the other branching rules have a lesser branching number, the size of the search tree is bounded by 3^k. The total worst-case running-time of the algorithm is therefore $O(3^k \cdot (n+m))$. $\square$

6.2 Problem Kernels

For the editing variants we again make use of critical cliques in order to obtain a problem kernel.

Lemma 6.9. *Let* (G, g, k) *be an instance of* BCE *or* SBCE *and let* S *be an optimal solution. Let* v *be an unaffected vertex and let* K *be the critical clique in* G *containing* v*. Then the valid cluster containing* v *in* G_S *is* $K \cup N(K)$*, and every vertex in* K *is unaffected.*

Proof. Let C be the valid cluster in G_S containing v. Since v is unaffected, no edge incident with v is deleted by S, so $N[v] = K \cup N(K) \subseteq C$. Furthermore, no vertex $w \in V \setminus (K \cup N(K))$ can be in the same cluster C with v. Otherwise, S would insert the edge $\{v, w\} \notin E$ and v would be affected. Hence, $C \subseteq K \cup N(K)$ and therefore $C = K \cup N(K)$.

Let u be a vertex in K. Since K is a critical clique we have $N[u] = K \cup N(K) = C$. As u belongs to the valid cluster C in G_S, no edge modification is incident to u, hence u is unaffected. $\square$

6.2.1 Bicolored Cluster Editing

Similar to BCD, for BCE we can again remove all clusters that are already valid.

Reduction Rule 6. *Remove a valid cluster in* G*.*

Lemma 6.10. *Reduction Rule 6 is correct and can be exhaustively applied in $O(n + m)$ time.*

Proof. Let (G, g, k) be an instance of BCE and C a valid cluster in G. Let $S = S^- \dot\cup S^+$ be a solution such that there are two distinct clusters C_1 and C_2 with $C_1 \cap C \neq \emptyset, C_2 \cap C \neq \emptyset$ in G_S, both containing vertices from C.

Consider a new set of edge deletions $S'^- = S^- \setminus \{\{u, v\} \mid u \in C_1 \cap C, v \in C_2 \cap C\}$ and a new set of edge insertions $S'^+ = S^+ \setminus (\{\{u, v\} \mid u \in C_1 \cap C, v \in C_1 \setminus C\} \cup \{\{u, v\} \mid u \in C_2 \cap C, v \in C_2 \setminus C\})$. Now consider the edge modification set $S' := S'^- \cup S'^+$ that does not separate the vertices in C_1 and C_2 that were originally part of the cluster C and does not insert the edges between vertices in C and other vertices in C_1 and C_2, respectively. Since S' applies fewer deletions and fewer insertions than S, we have $|S'| < |S|$. Because the bicolored cluster property is hereditary, $C_1 \setminus C$ and $C_2 \setminus C$ are still valid clusters. Since C already was a valid cluster, S' is therefore also a solution. Hence, an optimal solution S^* does not delete any edges between vertices in C and does not insert edges between C and other connected components, so C can be safely removed from G.

Using a modified breadth-first search every connected component can be computed and checked for validity in $O(n + m)$ time. □

The next reduction rule bounds the number of vertices in a critical clique, which will help to bound the number of unaffected vertices in a reduced instance.

Reduction Rule 7. *Let K be a critical clique in G with $|K| > k + 1$. Insert every missing edge between vertices in $N(K)$, delete every edge in $E(N(K), N^2(K))$, reduce k accordingly, and delete $K \cup N(K)$ from G.*

Lemma 6.11. *Reduction Rule 7 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of BCE. Let S be a solution such that $K \cup N(K)$ is not a cluster in G_S. Let C with $C \cap (K \cup N(K)) \neq \emptyset$ be a cluster in G_S and let K_C denote the vertices from $K \cup N(K)$ in C. This means that S deletes all edges between K_C and $K \setminus K_C$. According to Observation 1 this requires at least $|K| - 1 > k$ edge deletions, so S is not valid. Hence, for every valid solution S^* the resulting cluster graph G_{S^*} must contain $K \cup N(K)$ as a cluster and the reduction rule is correct.

For a given graph G all critical cliques can be determined in $O(n+m)$ time. Using a modified breadth-first search the critical cliques can be checked for validity and the edges between critical cliques can be determined in $O(n+m)$ time.

Since every application of Reduction Rule 7 deletes at least one vertex, the rule can be applied at most n times. Therefore, Reduction Rule 7 can be exhaustively applied in $O(n^2 + n \cdot m)$ time. □

Lemma 6.12. *Let $I := (G, g, k)$ be exhaustively reduced with respect to Reduction Rules 6 and 7. If G has more than $2k^2 + 4k$ vertices, then I is a no-instance of* BCE*.*

Proof. Let I be a yes-instance of BCE and let S be a valid solution for I. We prove the lemma by giving an upper bound on the number of vertices in G that are affected and unaffected by S. Let V_α denote the vertices affected by S and V_β denote the vertices that are unaffected by S, recall that $V_\alpha \dot\cup V_\beta = V$. Clearly, $|V_\alpha| \leq 2k$, since S is valid and every edge in S is incident to at most two unique vertices. According to Lemma 6.9 every unaffected vertex $v \in V_\beta$ is part of a critical clique K in G, such that every vertex in K is unaffected and $K \cup N(K)$ is a valid cluster in G_S. Let $K_1, K_2, \ldots, K_r, r \geq 0$ denote the critical cliques in G that contain the unaffected vertices.

Since I is exhaustively reduced with respect to Reduction Rule 6, every cluster $K_i \cup N(K_i), 1 \leq i \leq r$, is not an isolated cluster in G and therefore contains at least one vertex $v_i \in N(K_i)$ that is incident with an edge $\{v_i, u\} \in S$. Note that the other vertex u incident with that edge can be in $N(K_j)$ for another critical clique K_j with $j \in \{1, \ldots r\}, j \neq i$. Since S is valid, this gives us $r \leq 2|S| \leq 2k$.

Since G is reduced with respect to Reduction Rule 7, for every remaining critical clique K_i we also have that $|K_i| \leq k + 1$. Thus, in total we get

$$|V_\beta| = \sum_{i=1}^{r} |K_i| \leq \sum_{i=1}^{r} k + 1 \leq 2k \cdot (k + 1) = 2k^2 + 2k$$

and finally

$$|V| = |V_\alpha| + |V_\beta| \leq 2k + (2k^2 + 2k) = 2k^2 + 4k.$$

□

Theorem 6.13. BCE *admits a $2k^2 + 4k$-vertex kernel that can be computed in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of BCE. The kernelization algorithm for BCE first exhaustively applies Reduction Rules 6 and 7. Then, if for the resulting graph G' we have $|V(G')| > 2k^2 + 4k$, the algorithm returns a trivial no-instance.

Exhaustively applying Reduction Rules 6 and 7 takes $O(n^2 + n \cdot m)$ time. The correctness of the kernelization algorithm follows from Lemma 6.12. □

6.2.2 Strict Bicolored Cluster Editing

Now we proceed to present a problem kernel for SBCE, again using the notion of critical cliques to bound the number of affected and unaffected vertices in the input graph of a yes-instance. Recall that for SBCE a valid cluster of size at least two is a cluster that is induced $K_{(2,2)}$-free and is not a monochromatic cluster. Singletons are also considered a valid cluster. Note that Lemma 6.9 also holds for SBCE, only the notion of a valid cluster is slightly different.

Unlike BCE, for SBCE in general we cannot remove all valid clusters from the input graph. This is, because in order to achieve a strict bicolored cluster graph, it can be necessary to separate a vertex from a valid cluster C and include it into another cluster C', so that C' has at least one vertex for both colors.

However, we can bound the number of valid clusters in the input graph using the following reduction rules.

Reduction Rule 8. *Let $C_1, \ldots, C_{k+1}$ be valid black-dominated clusters in G with $|C_1| \leq \cdots \leq |C_{k+1}|$. Remove C_{k+1} from G.*

Reduction Rule 9. *Let $C_1, \ldots, C_{k+1}$ be valid white-dominated clusters in G with $|C_1| \leq \cdots \leq |C_{k+1}|$. Remove C_{k+1} from G.*

In order to prove the correctness of Reduction Rule 8 and 9 we make use of Lemma 6.5 and Lemma 6.6.

Lemma 6.14. *Reduction Rule 8 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Let (G, g, k) be an instance of SBCE. Let S^* be an optimal solution that deletes or inserts at least one edge incident to a vertex in C_{k+1}. We show that we can always find a solution S' that applies at most the same number of edge modifications as S^*, but leaves every vertex in C_{k+1} unaffected. This then implies that there is an optimal solution that does not affect any vertex in C_{k+1} and the valid cluster C_{k+1} can therefore be safely removed.

Let $\mathcal{C} \coloneqq \{C_1, \ldots, C_{k+1}\}$. First, note that according to Lemma 6.5 for every cluster $C_i \in \mathcal{C}$ in G we have that in G_{S^*} every vertex $v \in C_i$ is either in a cluster $C'_i \subseteq C_i$ of size at least two or is in a cluster C_v with $C_v \cap C_i = \{v\}$ and there is no other vertex in C_v with the same color as v.

We can therefore assume that in G_{S^*} every vertex in C_{k+1} is either part of a cluster $C'_{k+1} \subseteq C_{k+1}$ or gets separated from every other vertex in C_{k+1}.

Let w_{k+1} be the single white vertex in C_{k+1}. Let D_{k+1} contain every black vertex $v \neq w_{k+1}$ from C_{k+1} that gets separated from w_{k+1}, and therefore also separated from every other black vertex $u \in C_{k+1}$, and let $r \coloneqq |D_{k+1}|$. We show that there is another solution S' that instead separates r black vertices from other clusters in $\mathcal{C}$. We can assume that $r < k$, since otherwise S^* applies more than k edge modifications and is not valid. Furthermore, since $\mathcal{C}$ contains $k+1$ clusters, we can also assume that at least r many of the clusters in $\mathcal{C} \setminus \{C_{k+1}\}$ are unaffected, otherwise S^* would be not valid. Let $\mathcal{C}^\beta \coloneqq \{C_1^\beta, C_2^\beta, \dots, C_q^\beta\} \subset \mathcal{C}$, $q \geq r$, with $|C_1^\beta| \leq |C_2^\beta| \leq \cdots \leq |C_q^\beta|$ denote the unaffected (with respect to S^*) clusters from $\mathcal{C}$ and let $r_i \coloneqq |C_i^\beta| - 1$ denote the number of black vertices in C_i^β.

We can then get a solution S' that leaves C_{k+1} as a cluster and instead creates r black singletons from clusters in $\mathcal{C}^\beta$. This can be done by first completely splitting up the clusters $C_1^\beta, C_2^\beta, \dots, C_{\ell-1}^\beta$ for some $\ell \in \{1, \dots, q\}$, thus getting $r' \coloneqq r_1 + r_2 + \cdots + r_{\ell-1}$ black singletons. Then the remaining $\tilde{r}_\ell \coloneqq r - r'$ black vertices are separated from the cluster C_ℓ^β. This is always possible since $q \geq r$ and every cluster in $\mathcal{C}^\beta$ contains at least one black vertex. It remains to show that S' applies at most the same number of edge modifications as S^*.

Separating the r black vertices from C_{k+1} requires $|C_{k+1}|-1$ edge deletions to separate the first vertex, $|C_{k+1}| - 2$ for the second, and so on, resulting in

$$x^{S^*} \coloneqq \sum_{i=1}^{r} |C_{k+1}| - i$$

edge deletions applied by S^*. Instead completely splitting up C_j^β, $1 \leq j \leq \ell - 1$, requires

$$x_j^{S'} \coloneqq \sum_{i=1}^{r_j} |C_j^\beta| - i = \sum_{i=1}^{r_j} (r_j + 1) - i = \sum_{i=1}^{r_j} i$$

edge deletions, respectively. Additionally, separating the $\tilde{r}_\ell$ remaining vertices from C_ℓ^β requires another

$$x_\ell^{S'} \coloneqq \sum_{i=1}^{\tilde{r}_\ell} |C_\ell^\beta| - i$$

edge deletions. Since $|C_{k+1}| \geq |C_j|$ for every $|C_j| \in \mathcal{C}$, we have

$$\sum_{i=1}^{\tilde{r}_\ell} |C_{k+1}| - i \geq \sum_{i=1}^{\tilde{r}_\ell} |C_\ell^\beta| - i. \tag{4}$$

Furthermore, we have

$$\sum_{i=\tilde{r}_\ell+1}^{r} |C_{k+1}| - i \geq \sum_{j=1}^{\ell-1} \sum_{i=1}^{r_j} i, \tag{5}$$

since the sums on both sides of the equation contain exactly r' terms and the smallest term in the sum $\sum_{i=\tilde{r}_\ell+1}^{r} |C_{k+1}| - i$ is $|C_{k+1}| - r \geq 1$. In total we get

$$x^{S^*} = \sum_{i=1}^{r} |C_{k+1}| - i \geq \big(\sum_{i=1}^{\tilde{r}_\ell} |C_\ell^\beta| - i\big) + \sum_{j=1}^{\ell-1} \sum_{i=1}^{r_j} i = x_\ell^{S'} + \sum_{j=1}^{\ell-1} x_j^{S'},$$

so S' applies at most as many edge deletions as S^*.

Determining all valid black-dominated clusters in G and computing their sizes can be done in $O(n+m)$ time. Deleting the cluster C_{k+1} can then also be done in $O(n+m)$ time.

Since every application of Reduction Rule 8 deletes at least one vertex, the rule can be applied at most n times. Therefore, Reduction Rule 8 can be exhaustively applied in $O(n^2 + n \cdot m)$ time. □

Lemma 6.15. *Reduction Rule 9 is correct and can be exhaustively applied in $O(n^2 + n \cdot m)$ time.*

Proof. Lemma 6.15 can be proven analogously to Lemma 6.14. □

Note that Lemma 6.9 and Lemma 6.11 are also correct for SBCE. Using Reduction Rule 7, 8 and 9 we now get a problem kernel for SBCE.

Lemma 6.16. *Let $I \coloneqq (G, g, k)$ be exhaustively reduced with respect to Reduction Rules 7, 8 and 9. If G has more than $4k^2 + 6k$ vertices, then I is a no-instance of* SBCE*.*

Proof. Let I be a yes-instance of SBCE and let S be a valid solution for I. We prove the lemma by giving an upper bound on the number of vertices in G that are affected and unaffected by S. Let V_α denote the vertices affected by S and V_β denote the vertices that are unaffected by S and recall that $V_\alpha \dot\cup V_\beta = V$. Clearly, $|V_\alpha| \leq 2k$, since S is valid and every edge in S is incident to at most two unique vertices. According to Lemma 6.9 every unaffected vertex $v \in V_\beta$ is part of a critical clique K in G, such that every vertex in K is unaffected and $K \cup N(K)$ is a valid cluster in G_S.

Let $\mathcal{C}_b$ denote the valid black-dominated clusters in G and $\mathcal{C}_w$ denote the valid white-dominated clusters in G. Since I is exhaustively reduced with respect to Reduction Rule 8 and 9, we have $|\mathcal{C}_b| \leq k$ and $|\mathcal{C}_w| \leq k$.

Let $K_1, K_2, \ldots, K_r, r \geq 0$, denote the critical cliques in G that contain the unaffected vertices. Every cluster in G_S contains at most one critical clique K_i. Since every cluster in G_S except for at most $2k$ clusters from $\mathcal{C}_b$ and $\mathcal{C}_w$ contains an affected vertex, there are at most $2k + 2k = 4k$ clusters in G_S and we therefore also have $r \leq 4k$.

Since G is reduced with respect to Reduction Rule 7, for every remaining critical clique K_i we also have that $|K_i| \leq k+1$. Thus, in total we get

$$|V_\beta| = \sum_{i=1}^{r} |K_i| \leq \sum_{i=1}^{r} (k+1) \leq 4k \cdot (k+1) = 4k^2 + 4k$$

and finally

$$|V| = |V_\alpha| + |V_\beta| \leq 2k + (4k^2 + 4k) = 4k^2 + 6k.$$

□

Theorem 6.17. SBCE *admits a* $4k^2+6k$*-vertex kernel that can be computed in* $O(n^2 + n \cdot m)$ *time.*

Proof. Let (G, g, k) be an instance of SBCE. The kernelization algorithm for SBCE first exhaustively applies Reduction Rules 7, 8 and 9. Then, if for the resulting graph G' we have $|V(G')| > 4k^2 + 6k$, the algorithm returns a trivial no-instance.

Exhaustively applying Reduction Rules 7, 8 and 9 takes $O(n^2+n\cdot m)$ time. The correctness of the kernelization algorithm follows from Lemma 6.16. □

7 ILP-Formulation and Experimental Results

In this section we describe the experiments we ran on graphs obtained from biological data sets [10]. We took each graph as input of an instance of the optimization version of BCD and BCE and tried to solve the corresponding Integer Linear Program (ILP) using the Gurobi solver.[1] We first describe our ILP-formulation and the details of the experiments. Then we analyze and compare the results for both variants.

7.1 ILP-Formulation

For a formal definition of and general information about Integer Linear Programs (ILPs) we refer to [18].

We consider the ILP for both of our problems as minimization problems. We first formulate the ILP for BCD. Let $G = (V, E)$ be the input graph. For each edge $e \in E$, we introduce a binary variable $x_e \in \{0, 1\}$. Setting $x_e = 0$ represents that e is deleted by the solution set S, while $x_e = 1$ represents that e is still present in G_S. Since the goal is to delete as few edges as possible in order to transform G into a bicolored cluster graph, the objective function is given by the total number of edges in G minus the sum of all edge variables. As shown by Lemma 3.3 a bicolored cluster graph can be characterized as a graph that is P_3-free and $K_{(2,2)}$-free. We make use of this property for the construction of the constraints of our ILP.

We consider three types of constraints. Let $\mathcal{P}_G$ denote the set of induced P_3s in G, let $\mathcal{T}_G$ denote the set of triangles in G and let $\mathcal{K}_G$ denote the set of induced $K_{(2,2)}$s in G. For each $P \in \mathcal{P}_G$ we introduce a constraint that only allows one of the two edges in P to be present in G_S. We call constraints of this type P_3-*constraints*. Note that the deletion of edges can result in new induced P_3s being created. We therefore also add a set of three constraints for each triangle $T \in \mathcal{T}_G$ in G that make sure that two of the three edges in T can only be present in G_S if the third was also not deleted. These constraints are referred to as *triangle-constraints*. Finally, for each $K \in \mathcal{K}_G$ we add a constraint that only allows three of the six edges between vertices in K to be present in G_S. This guarantees that not all four vertices from K can end up in the same cluster. We denote these constraints as *color-constraints*.

Note that in order to restrict the initial number of constraints and thus speed-up the construction of the ILP we at first do not include the color-constraints in the ILP. Instead, using the callback functionality of Gurobi we check in each of our callbacks (that gets called if the current solution

[1] see https://www.gurobi.com/

satisfies all current constraints, that is if the `where` variable of the callback class has the value MIPSOL) whether the current solution graph contains an induced $K_{(2,2)}$. We then add the corresponding color constraints to our model as lazy constraints, if they are violated by the current solution.

The base ILP is given by

$$\begin{array}{llll}
\text{minimize} & m - \sum_{e \in E} x_e, & & \\
\text{subject to} & x_{\{u,v\}} + x_{\{v,w\}} & \leq 1 & \forall (u,v,w) \in \mathcal{P}_G, \\
& & & \\
& -x_{\{u',v'\}} + x_{\{v',w'\}} + x_{\{u',w'\}} & \leq 1 & \forall \{u',v',w'\} \in \mathcal{T}_G, \\
& x_{\{u',v'\}} - x_{\{v',w'\}} + x_{\{u',w'\}} & \leq 1 & \\
& x_{\{u',v'\}} + x_{\{v',w'\}} - x_{\{u',w'\}} & \leq 1 & \\
& & & \\
& x_e \in \{0,1\} & & \forall e \in E.
\end{array}$$

Using Gurobi callbacks we also add the color constraints

$$\sum_{\substack{u,v \in K \\ u \neq v}} x_{\{u,v\}} \quad \leq 3 \qquad \forall K \in \mathcal{K}_{G_S}$$

if there are any induced $K_{(2,2)}$s in the solution graph G_S of the current solution S, for which all previous constraints are satisfied. This procedure is continued until a solution S is found that satisfies all constraints and for which no induced $K_{(2,2)}$ is contained in G_S.

For BCE we use a similar ILP formulation. Let $\overline{E} \coloneqq \binom{V}{2} \setminus E$ denote the set of *missing edges* in G. Besides a binary variable x_e for each edge $e \in E$, we now also have a binary variable $x_{e'} \in \{0,1\}$ for each missing-edge $e' \in \overline{E}$. If for a missing-edge e' we have $x_{e'} = 1$, then e' is inserted by the solution S, while $x_{e'} = 0$ corresponds to e' still not being present in G_S. For the objective function we expand the objective function used for BCD by also adding the sum of all missing-edge variables $x_{e'}$. This is because setting a missing-edge variable to 1 represents an edge insertion, which we want to minimize.

Let $\mathcal{P}_G$, $\mathcal{T}_G$, and $\mathcal{K}_G$ again denote the set of induced P_3s, triangles and induced $K_{(2,2)}$s in G, respectively. Since for BCE edges can also be inserted, instead of a single constraint we now also have a set of three constraints for each induced P_3 in G, similar to the triangle constraints. Moreover, inserting edges can create new induced P_3s that must be handled. Therefore we check in each callback whether the current solution graph G_S contains any induced P_3 and add the corresponding constraints as lazy constraints to our model. We also again include the color constraints via callbacks for each induced $K_{(2,2)}$ in the solution graph G_S of the current solution S.

The base ILP is thus given by

$$
\begin{array}{lrll}
\text{minimize} & m - \sum\limits_{e \in E} x_e + \sum\limits_{e' \in \overline{E}} x_{e'}, & & \\
\text{subject to} & -x_{\{u,v\}} + x_{\{v,w\}} + x_{\{u,w\}} & \leq 1 & \forall (u,v,w) \in \mathcal{P}_G, \\
& x_{\{u,v\}} - x_{\{v,w\}} + x_{\{u,w\}} & \leq 1 & \\
& x_{\{u,v\}} + x_{\{v,w\}} - x_{\{u,w\}} & \leq 1 & \\
& & & \\
& -x_{\{u',v'\}} + x_{\{v',w'\}} + x_{\{u',w'\}} & \leq 1 & \forall \{u',v',w'\} \in \mathcal{T}_G, \\
& x_{\{u',v'\}} - x_{\{v',w'\}} + x_{\{u',w'\}} & \leq 1 & \\
& x_{\{u',v'\}} + x_{\{v',w'\}} - x_{\{u',w'\}} & \leq 1 & \\
& & & \\
& x_e \in \{0,1\} & & \forall e \in E, \\
& x_{e'} \in \{0,1\} & & \forall e' \in \overline{E}.
\end{array}
$$

Using Gurobi callbacks we also add constraints

$$
\begin{array}{rll}
-x_{\{u,v\}} + x_{\{v,w\}} + x_{\{u,w\}} & \leq 1 & \forall (u,v,w) \in \mathcal{P}_{G_S}, \\
x_{\{u,v\}} - x_{\{v,w\}} + x_{\{u,w\}} & \leq 1 & \\
x_{\{u,v\}} + x_{\{v,w\}} - x_{\{u,w\}} & \leq 1 &
\end{array}
$$

if there are any induced P_3s in the solution graph G_S of the current solution S, for which all previous constraints are satisfied, and the color constraints

$$\sum_{\substack{u,v \in K \\ u \neq v}} x_{\{u,v\}} \quad \leq 3 \qquad \forall K \in \mathcal{K}_{G_S}$$

if there are any induced $K_{(2,2)}$s in G_S. This procedure is continued until a solution S is found that satisfies all constraints and for which no induced P_3 and no induced $K_{(2,2)}$ is contained in G_S.

7.2 Implementation Details

The experiments were run on an Intel(R) Core(TM) i5-8300H CPU 2.30GHz machine with 8GB RAM under the Windows 10 Pro operating system. Our implementation[2] is done with Java using NetBeans IDE 8.2, running under the OpenJDK runtime environment in version 1.8.0_252. To construct and solve our ILPs we used the Gurobi Optimizer[3] in version 9.0.3 under an academic license.

[2]The source code of our implementation and the result files can be found under https://www.uni-marburg.de/en/fb12/research-groups/algorith/bce.zip.

[3]see https://www.gurobi.com/

For our experiments we used as input the graphs obtained from biological data used by Fertin et al. [10]. In their work Fertin et al. constructed graphs with black and white vertices from the genomes of *Ricinus communis* [6] (castor bean), *Populus trichocarpa* [19] (western balsam poplar) and *Theobroma cacao* [2] (cacao tree). In both sets the black vertices represented the genes of *Populus trichocarpa*. In one set of graphs the genes of *Ricinus communis* were represented by white vertices, another set of graphs had the genes of *Theobroma cacao* as white vertices. Edges between genes were inserted based on BLAST Expect (E) values [1]. An edge was added between to genes if the E-value was below some threshold. Three values were used for the threshold T, with $T \in \{0, 10^{-80}, 10^{-140}\}$, thus giving two sets of graphs with three graphs each. Each increment of the threshold value resulted in approximately double the number of edges. Note that all graphs do not contain edges between two white vertices.

In the following we refer to the input graphs as Cacao-Poplar-X or Ricinus-Poplar-X, with X being the respective threshold value.

According to Lemma 2 and Lemma 4 for an instance of BCD or BCE we can solve each connected component individually. We therefore separately solved the ILP for each connected component of the input graph and then aggregated the results. We only considered components that contain at least two black and at least two white vertices, since those are the particularly interesting components, where from a biological viewpoint the orthology relations still have to be resolved. By this we also ignore very small and trivial components, of which there are a lot in the input graphs. Because of the huge amount of edge variables and constraints involved, we also excluded some extraordinarily large components with > 800 vertices. Since especially for BCE we have a relatively high number of constraints and callbacks even for smaller components, not all components could be solved in reasonable time. We therefore set the time limit for the solver to ten minutes for each component.

7.3 Results

We now present the results of our experiments. Table 1 shows the total running time and number of unsolved components for each instance alongside some properties of the input graphs.

For BCD for each instance almost all components could be solved within the time limit, with only one (for Cacao-Poplar-0) to five (for Ricinus-Poplar-10^{-80}) unsolved components. For BCE the number of unsolved components ranged from 22 (for Ricinus-Poplar-0) to 73 (for Ricinus-Poplar-10^{-80}). For Ricinus-Poplar-10^{-80} two of the unsolved components are large components

Table 1: Statistics for each input graph with T denoting the threshold value. n_B and n_W denote the number of black and white vertices, respectively, m the number of edges, K the number of connected components and $K_{nontriv}$ the number of considered (nontrivial) components. $K^{\mathrm{BCD}}_{unsolv}$ and $K^{\mathrm{BCE}}_{unsolv}$ are the number of unsolved components and t_{BCD} and t_{BCE} the total running-time (in minutes) for BCD and BCE, respectively.

	Cacao-Poplar			Ricinus-Poplar		
T	0	10^{-140}	10^{-80}	0	10^{-140}	10^{-80}
n_B	14 287	18 140	25 165	14 235	18 079	24 472
n_W	9 907	12 488	16 963	9 075	11 333	15 111
m	73 837	131 568	368 430	64 162	109 352	269 862
K	6 152	7 043	7 847	6 247	7 102	7 980
$K_{nontriv}$	1 336	1 659	2 223	1 175	1 505	2 022
$K^{\mathrm{BCD}}_{unsolv}$	1	5	4	1	4	5
$K^{\mathrm{BCE}}_{unsolv}$	31	54	-	22	35	73
t_{BCD}	11.8	51.1	48.6	10.9	47.9	54.1
t_{BCE}	358.2	598.3	-	232.4	407.5	794.4

(with 306 and 651 vertices) that could not be handled and terminated with an error.

For Cacao-Poplar-10^{-80} while consecutively solving the ILP for each component an unsalvageable crash in the Gurobi framework occurred and we thus could not generate data for that instance for BCE. For that reason we exclude Cacao-Poplar-10^{-80} from further analysis.

When only considering the components that were solved by both variants we get the following results, which we use to compare the running-time, solution size and distribution of inferred clusters between the two variants. Table 2 shows the statistics for the components that were solved for both BCD and BCE.

For BCD the total running time was for all instances under 20 seconds. In contrast, for BCE even for the components that could be solved by both variants the total running time still ranged from around 12 minutes to 1.4 hours, taking significantly longer than for BCD. For BCE a rather small number of edges was inserted, ranging from 3.2% to 3.9% of the total number of edge modifications. The size of the largest resulting cluster ranged from 23 to 33 vertices for BCD and from 29 to 38 vertices for BCE, with the average cluster size being slightly higher for BCE across all instances. Somewhat surprisingly, more singletons were created for BCE than for BCD. Conversely, the resulting graph contained more isolated edges as well as more black-dominated clusters in the case of BCD. This is due to the fact that for BCE fewer, but bigger clusters are created. For both variants, the resulting graph contained no monochromatic white and only for BCE a few white-dominated clusters, since in the input graphs no white vertices are adjacent.

Table 2: Results for the components that were solved for both BCD and BCE. Here t is the total time (in seconds) needed for the instance; k_{del} and k_{ins} denote the number of deletions/insertions; K_1 and P_2 denote the number of singletons and isolated edges, respectively; mB and mW are the number of monochromatic clusters of the respective color; dB and dW are the number of non-monochromatic valid clusters; $\varnothing C$ denotes the average size of the clusters.

	t	k_{del}	k_{ins}	K_1	P_2	mB	mW	dB	dW	$\varnothing C$
Cacao-Poplar-0 / $n_B = 5\,983$, $n_W = 4\,034$, $m = 32\,375$										
BCD	13.4	15 181	-	2 111	455	1	0	1 507	0	2.46
BCE	2893.3	14 030	504	2 220	354	0	0	1 470	6	2.47
Cacao-Poplar-10^{-140} / $n_B = 7\,669$, $n_W = 5\,256$, $m = 44\,125$										
BCD	11.2	21 569	-	2 889	522	1	0	1 881	0	2.44
BCE	3 498.2	19 965	661	3 047	411	0	0	1 814	2	2.45
Ricinus-Poplar-0 / $n_B = 5\,506$, $n_W = 3\,434$, $m = 29\,117$										
BCD	7.9	12 708	-	1 747	399	1	0	1 332	0	2.57
BCE	743.2	11 701	407	1 846	308	1	0	1 302	2	2.58
Ricinus-Poplar-10^{-140} / $n_B = 7\,605$, $n_W = 4\,682$, $m = 45\,773$										
BCD	13.3	21 414	-	2 468	496	2	0	1 780	0	2.59
BCE	3 451.8	19 333	792	2 616	403	1	0	1 692	3	2.60
Ricinus-Poplar-10^{-80} / $n_B = 10\,387$, $n_W = 6\,463$, $m = 64\,664$										
BCD	19.7	30 599	-	3 491	735	0	0	2 318	0	2.57
BCE	5 064.8	28 235	996	3 710	548	0	0	2 235	12	2.59

In summary, both variants give similar results in regards to the solution size and the distribution of the clusters. Considering the drastically higher running-time and slightly higher number of singletons for BCE, according to this preliminary results BCD appears to be the more favorable model, which should be further investigated with more elaborated experiments that also take additional biological information into account.

Note that we did not conduct our experiments for the strict variants SBCD and SBCE, since their ILP-formulation involves far more variables and constraints. However, it is worth mentioning that for our input graphs the results for SBCD and SBCE would likely be very similar to those we obtained for BCD and BCE, since almost no monochromatic cluster (which are not allowed for the strict variants) were created.

8 Conclusion

In this section we summarize our results, pose open questions and give directions for future work.

8.1 Summary

In this work we presented several decision problems in the context of graph-based orthology assignment that can be seen as a generalization of previous models [10].

In Section 3 we formulated the problems we analyzed in this work and showed some properties of the desired solution graphs. In Section 4 we showed the **NP**-hardness of our problems. More precisely, we showed that all of the considered problems are **NP**-complete, even when restricted to graphs with maximum degree six.

In Section 5 we then analyzed the parameterized complexity of the deletion variants BCD and SBCD for the solution size k as parameter. We provided **FPT**-algorithms for both problems and showed that they can be solved in polynomial time on cluster graphs. We then showed that BCD admits a linear-vertex kernel and SBCD admits a subquadratic-vertex kernel for k. In Section 6 we continued our analysis for the editing variants BCE and SBCE. We again provided **FPT**-algorithms for both problems and showed that both variants admit quadratic-vertex kernels for k. Table 3 summarizes the results of our complexity analysis.

In Section 7 we then ran experiments on graphs obtained from biological data [10]. Using an Integer Linear Program formulation we solved the optimization versions of BCD and BCE on the input graphs and compared the results for both variants, providing a first practical assessment of the proposed problems.

Table 3: Complexity analysis results for all variants considered in this work, parameterized by the solution size k.

problem	FPT-algorithm	problem kernel
BCD	$O(2^k \cdot (n+m))$	$4k$-vertex kernel
SBCD	$O(2^k \cdot (n+m))$	$(2k^{\frac{3}{2}} + 2k)$-vertex kernel
BCE	$O(3^k \cdot (n+m))$	$(2k^2 + 4k)$-vertex kernel
SBCE	$O(3^k \cdot (n+m))$	$(4k^2 + 6k)$-vertex kernel

8.2 Future Work

In all variants we considered in this work a cluster can also consist of a single vertex, which can be interpreted as an unmatched gene. Note that in the deletion case not allowing singletons would in many cases lead to not finding a clustering at all. Take for example an induced path of length two, $P = (v_1, v_2, v_3)$, as an input graph G. The only possibilities to transform G into a cluster graph are deleting either of the two edges $\{v_1, v_2\}, \{v_2, v_3\}$ or both, in any case creating a singleton. In the editing case, however, a clustering without singletons can always be achieved by a sufficient amount of edge modifications. This could motivate another editing variant, in which the resulting graph does not contain any singletons.

We considered problems where the goal is to achieve a cluster graph and a cluster is defined as a connected component that is a clique and contains at most one vertex of one of the two colors. Another variation of interest would be to consider as a cluster a connected component that must not necessarily be a clique. A special case of this variation would be where the input graph is bipartite, with both partitions including all vertices of one of the two colors. Another extension of our models would be to consider edge-weighted graphs as in other orthology assignment problems [10, 20].

Moreover, it would be of interest to explore the biological significance of the problems proposed in this work. To this end further experiments that incorporate phylogenetic information could be conducted.

References

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] Xavier Argout, Jerome Salse, Jean-Marc Aury, Mark J Guiltinan, Gaetan Droc, Jerome Gouzy, Mathilde Allegre, Cristian Chaparro, Thierry Legavre, Siela N Maximova, et al. The genome of theobroma cacao. *Nature genetics*, 43(2):101–108, 2011.

[3] Lars Arvestad, Ann-Charlotte Berglund Sonnhammer, J. Lagergren, and B. Sennblad. Bayesian gene/species tree reconciliation and orthology analysis using mcmc. *Bioinformatics*, 19 Suppl 1:i7–15, 2003.

[4] Ann-Charlotte Berglund, Erik Sjölund, Gabriel Ostlund, and Erik Sonnhammer. Inparanoid 6: Eukaryotic ortholog clusters with inparalogs. *Nucleic acids research*, 36:D263–6, 02 2008.

[5] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey.* Society for Industrial and Applied Mathematics, 1999.

[6] Agnes P Chan, Jonathan Crabtree, Qi Zhao, Hernan Lorenzi, Joshua Orvis, Daniela Puiu, Admasu Melake-Berhan, Kristine M Jones, Julia Redman, Grace Chen, et al. Draft genome sequence of the oilseed species ricinus communis. *Nature biotechnology*, 28(9):951–956, 2010.

[7] Jianer Chen and Jie Meng. A 2k kernel for the cluster editing problem. *Journal of Computer and System Sciences*, 78(1):211–220, 2012.

[8] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 4. Springer, 2015.

[9] Rosa Fernández, Toni Gabaldón, and Christophe Dessimoz. Orthology: definitions, inference, and impact on species phylogeny inference, 2019.

[10] Guillaume Fertin, Falk Hüffner, Christian Komusiewicz, and Manuel Sorge. Matching algorithms for assigning orthologs after genome duplication events. *Computational Biology and Chemistry*, 74:379–390, 2018.

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

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

[13] René Heijden, Berend Snel, Vera van Noort, and Martijn Huynen. Orthology prediction at scalable resolution by phylogenetic tree analysis. *BMC bioinformatics*, 8:83, 02 2007.

[14] Jaime Huerta-Cepas, Damian Szklarczyk, Kristoffer Forslund, Helen Cook, Davide Heller, Mathias C. Walter, Thomas Rattei, Daniel R. Mende, Shinichi Sunagawa, Michael Kuhn, Lars Juhl Jensen, Christian von Mering, and Peer Bork. eggNOG 4.5: a hierarchical orthology framework with improved functional annotations for eukaryotic, prokaryotic and viral sequences. *Nucleic Acids Research*, 44(D1):D286–D293, 11 2015.

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

[16] David M Kristensen, Yuri I Wolf, Arcady R Mushegian, and Eugene V Koonin. Computational methods for gene orthology inference. *Briefings in bioinformatics*, 12(5):379–391, 2011.

[17] Marcus Lechner, Maribel Hernandez-Rosales, Daniel Doerr, Nicolas Wieseke, Annelyse Thévenin, Jens Stoye, Roland K Hartmann, Sonja J Prohaska, and Peter F Stadler. Orthology detection combining clustering and synteny for very large datasets. *PLoS one*, 9(8):e105015, 2014.

[18] Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.

[19] Gerald A Tuskan, Stephen Difazio, Stefan Jansson, Jörg Bohlmann, Igor Grigoriev, Uffe Hellsten, Nicholas Putnam, Steven Ralph, Stephane Rombauts, Asaf Salamov, et al. The genome of black cottonwood, populus trichocarpa (torr. & gray). *science*, 313(5793):1596–1604, 2006.

[20] Chunfang Zheng, Krister Swenson, Eric Lyons, and David Sankoff. Omg! orthologs in multiple genomes–competing graph-theoretical formulations. In *International Workshop on Algorithms in Bioinformatics*, pages 364–375. Springer, 2011.