On the Parameterized Complexity of Simultaneous Deletion Problems*

Akanksha Agrawal[1], R. Krithika[2], Daniel Lokshtanov[3], Amer E. Mouawad[4], and M. S. Ramanujan[5]

1 **Department of Informatics, University of Bergen, Bergen, Norway**
`akanksha.agrawal@uib.no`

2 **Institute of Mathematical Sciences, HBNI, Chennai, India**
`rkrithika@imsc.res.in`

3 **Department of Informatics, University of Bergen, Bergen, Norway**
`daniello@ii.uib.no`

4 **Department of Informatics, University of Bergen, Bergen, Norway**
`a.mouawad@uib.no`

5 **Algorithms and Complexity Group, Vienna University of Technology, Vienna, Austria**
`ramanujan@ac.tuwien.ac.at`

Abstract

For a family of graphs $\mathcal{F}$, an n-vertex graph G, and a positive integer k, the $\mathcal{F}$-Deletion problem asks whether we can delete at most k vertices from G to obtain a graph in $\mathcal{F}$. $\mathcal{F}$-Deletion generalizes many classical graph problems such as Vertex Cover, Feedback Vertex Set, and Odd Cycle Transversal. A (multi) graph $G = (V, \cup_{i=1}^{\alpha} E_i)$, where the edge set of G is partitioned into α color classes, is called an α-edge-colored graph. A natural extension of the $\mathcal{F}$-Deletion problem to edge-colored graphs is the Simultaneous $(\mathcal{F}_1, \dots, \mathcal{F}_\alpha)$-Deletion problem. In the latter problem, we are given an α-edge-colored graph G and the goal is to find a set S of at most k vertices such that each graph $G_i - S$, where $G_i = (V, E_i)$ and $1 \leq i \leq \alpha$, is in $\mathcal{F}_i$. Recently, a subset of the authors considered the aforementioned problem with $\mathcal{F}_1 = \ldots = \mathcal{F}_\alpha$ being the family of all forests. They showed that the problem is fixed-parameter tractable when parameterized by k and α and can be solved in $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ time[1]. In this work, we initiate the investigation of the complexity of Simultaneous $(\mathcal{F}_1, \dots, \mathcal{F}_\alpha)$-Deletion with different families of graphs. In the process, we obtain a complete characterization of the parameterized complexity of this problem when one or more of the $\mathcal{F}_i's$ is the class of bipartite graphs and the rest (if any) are forests. We show that if $\mathcal{F}_1$ is the family of all bipartite graphs and each of $\mathcal{F}_2 = \mathcal{F}_3 = \ldots = \mathcal{F}_\alpha$ is the family of all forests then the problem is fixed-parameter tractable parameterized by k and α. However, even when $\mathcal{F}_1$ and $\mathcal{F}_2$ are both the family of all bipartite graphs, then the Simultaneous $(\mathcal{F}_1, \mathcal{F}_2)$-Deletion problem itself is already W[1]-hard.

1998 ACM Subject Classification G.2.2 Graph Algorithms, I.1.2 Analysis of Algorithms

Keywords and phrases parameterized complexity, feedback vertex set, odd cycle transversal, edge-colored graphs, simultaneous deletion

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2017.9

* Due to space limitations most proofs have been omitted.
[1] We use the $\mathcal{O}^\star$ notation which hides factors that are polynomial in the input size.

1 Introduction

Given their tremendous modelling power, graphs have become an integral part of theoretical computer science in general, and of algorithm design in particular. One graph problem which encapsulates many problems of both practical and theoretical interest is $\mathcal{F}$-Deletion. For a family of graphs $\mathcal{F}$, an n-vertex graph G, and a positive integer k, the $\mathcal{F}$-Deletion problem asks whether we can delete at most k vertices from G to obtain a graph in $\mathcal{F}$. To state a few, $\mathcal{F}$-Deletion generalizes problems such as Vertex Cover [6], Feedback Vertex Set (FVS) [5, 8, 17], Vertex Planarization [15], Odd Cycle Transversal (OCT) [22, 14, 18], Interval Vertex Deletion [3], Chordal Vertex Deletion [4], and Planar $\mathcal{F}$-Deletion [11, 16].

A graph $G = (V, \cup_{i=1}^{\alpha} E_i)$, where the edge set of G is partitioned into α color classes, is called an α-edge-colored graph. Edge-colored graphs are fundamental in graph theory and have been extensively studied in the literature, especially for alternating cycles and monochromatic subgraphs [2]. A natural extension of the $\mathcal{F}$-Deletion problem to edge-colored graphs is the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem. In the latter problem, we are given an α-edge-colored graph G and the goal is to find a set S of at most k vertices such that each graph $G_i - S$ is in $\mathcal{F}_i$, where $G_i = (V, E_i)$ and $1 \leq i \leq \alpha$. Recently, Cai and Ye [2] studied several problems restricted to 2-edge-colored graphs, where edges are colored either red or blue. They asked, as an open question, whether the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem parameterized by k, with $\alpha = 2$ and $\mathcal{F}_1 = \mathcal{F}_2$ being the family of all forests, is fixed-parameter tractable (FPT), i.e. whether the problem can be solved in $\mathcal{O}^\star(f(k))$ time [10] (for some computable function f). Agrawal et al. [1] and Ye [24] answered this question in the affirmative. In particular, it was shown in [1] that the problem can be solved by an algorithm running in $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ time. This work pointed to a few natural further directions for research. For instance, does Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion remain fixed-parameter tractable when the family of all forests is replaced by the family of all bipartite graphs? What is the complexity of the problem when not all families are equal?

The results in this work allow us to take a significant step towards a better understanding of simultaneous deletion problems in general. To that end, we investigate the complexity of Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion in two settings. First, we consider the problem with $\mathcal{F}_1$ being the family of all bipartite graphs and $\mathcal{F}_2 = \mathcal{F}_3 = \ldots = \mathcal{F}_\alpha$ being the family of all forests. We call this problem Simultaneous FVS/OCT and define it as follows.

Simultaneous FVS/OCT **Parameter(s):** k and α
Input: An α-edge-colored graph $G = (V, \cup_{i=1}^{\alpha} E_i)$ and an integer k.
Question: Is there a set $S \subseteq V$ of size at most k such that $G_1 - S$ is a bipartite graph and $G_2 - S$, …, $G_\alpha - S$ are acyclic, where $G_i = (V, E_i)$ and $1 \leq i \leq \alpha$?

We call a solution S to the Simultaneous FVS/OCT problem a *sim-fvs-oct*. Our first contribution is an algorithm that, given an instance $(G = (V, \cup_{i=1}^{\alpha} E_i), k)$ of Simultaneous FVS/OCT, runs in time $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ and either computes a sim-fvs-oct in G of size at most k or correctly concludes that such a set does not exist.

In the second setting, we consider the Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem where $\mathcal{F}_1 = \ldots = \mathcal{F}_\alpha$ is the family of all bipartite graphs. We call this problem Simultaneous OCT and define it as follows.

Simultaneous OCT **Parameter(s):** k and α
Input: An α-edge-colored graph $G = (V, \cup_{i=1}^{\alpha} E_i)$ and an integer k.
Question: Is there a set $S \subseteq V$ of size at most k such that $G_i - S$ is bipartite, where $G_i = (V, E_i)$ and $1 \leq i \leq \alpha$?

We refer to a solution S to the Simultaneous OCT problem as a *sim-oct*. Our second (and rather surprising) contribution is a negative answer to the first open question of Agrawal et al. [1]. We show that, even for $\alpha = 2$, the Simultaneous OCT problem is W[1]-hard. To prove this result, we first reduce the well-known Multicolored Clique problem [7] to an auxiliary problem we call Simultaneous Cut. Simultaneous Cut is a natural generalization of the classical (s,t)-CUT problem to edge-colored graphs. Finally, we show that Simultaneous Cut can be reduced to Simultaneous OCT. Notice that W[1]-hardness of Simultaneous OCT implies that Simultaneous $(\mathcal{F}_1, \ldots, \mathcal{F}_\alpha)$-Deletion problem with at least two of the families being the family of all bipartite graphs is W[1]-hard.

Overview of the algorithm. Note that for any fixed k and α, our algorithm for solving the Simultaneous FVS/OCT problem runs in polynomial time. The said algorithm can be broken down into four stages, three of which are reductions to auxiliary problems. Initially, as was first proposed by Ye [24], we use the notion of compact representations of feedback vertex sets (see Section 2 for formal definitions) to reduce Simultaneous FVS/OCT into $2^{\mathcal{O}(\alpha k)}$ instances of the Colorful OCT problem, which is formally defined as follows. We note that, in any reduced instance, ℓ will be bounded from above by αk.

Colorful OCT **Parameter(s):** k and ℓ
Input: A graph $G = (V, E)$, integers k and ℓ, and a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$.
Question: Is there a set $S \subseteq V$ of size at most k such that $G - S$ is a bipartite graph and $S \cap P_i \neq \emptyset$, for $i \in \{1, \ldots, \ell\}$?

Intuitively, compact representations give us a partition of a vertex subset of the graph into sets such that picking one vertex from each part is "guaranteed" to constitute a feedback vertex set of each graph G_i, $2 \leq i \leq \alpha$. As such, we are able to encode the feedback vertex set "side" of the Simultaneous FVS/OCT problem (via the reduction) as colors on the vertices (i.e. different sets in $\mathcal{P}$ represent different colors for each vertex) and focus on a "colored" variant of Odd Cycle Transversal. Naturally, the second stage is to solve the Colorful OCT problem within the claimed running time. To do so, we reduce an instance of Colorful OCT to an instance of the compression variant of the problem, i.e. Colorful OCT Compression. This problem assumes an odd cycle transversal of size at most k as part of the input. Note that finding an odd cycle transversal of a graph $G = (V, E)$ of size at most k can be accomplished using the fixed-parameter tractable algorithms for OCT parameterized by solution size [14, 22], both of which run in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time.

Colorful OCT Compression **Parameter(s):** k and ℓ
Input: A graph $G = (V, E)$, integers k and ℓ, a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, and a set $O \subseteq V(G)$ of size at most k such that $G - O$ is bipartite.
Question: Is there a set $S \subseteq V$ of size at most k such that $G - S$ is a bipartite graph and $S \cap P_i \neq \emptyset$, for $i \in \{1, \ldots, \ell\}$?

Now, to solve an instance of Colorful OCT Compression, we reduce it into $2^{\mathcal{O}(k)}$ instances of yet another problem, namely Colorful Separator. This reduction is in many ways similar to the iterative compression algorithm for solving the Odd Cycle Transversal problem [7, 13, 23].

Colorful Separator **Parameter(s):** k and ℓ
Input: A graph $G = (V, E)$, integers k and ℓ, a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, and vertices s and t in $V(G)$.
Question: Is there an (s,t)-separator $S \subseteq V \setminus \{s, t\}$ such that $|S| \leq k$ and $S \cap P_i \neq \emptyset$, for each $i \in \{1, \ldots, \ell\}$?

Finally, and arguably the most technical part of our algorithm, is to show how to solve an instance of Colorful Separator. We will in fact solve a much more general problem, which we define in Section 4 (to keep the presentation clear). Our two main ingredients are a dynamic programming routine and a generalization of the concept of important separators, which has been recently defined to design parameterized algorithms for several "cut" problems [12, 19, 20]. We note that an alternative algorithm for solving Colorful Separator can be obtained by applying the treewidth reduction result of Marx et al. [21]. However, a "simple" application of this result would give an algorithm with a worse running time (double exponential).

2 Preliminaries

We denote the set of natural numbers by $\mathbb{N}$. For $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, 2, \ldots, n\}$. Given a universe $\mathcal{U}$, a set $S \subseteq \mathcal{U}$, and a family of sets $\mathcal{F} = \{F_1, \ldots, F_\ell\}$ over $\mathcal{U}$, we let $\mathcal{F}\big|_S$ denote the *restriction* of F to S, i.e. $\mathcal{F}\big|_S = \{F_1 \cap S, \ldots, F_\ell \cap S\}$. We use standard terminology from the book of Diestel [9] for the graph-related terms which are not explicitly defined here. For a graph G, we use $V(G)$ and $E(G)$ to denote the vertex and edge sets of G, respectively. For $S \subseteq V(G)$, by $N_G(S)$ we denote the set $\{u \in V(G) \setminus S \mid (u, v) \in E(G) \wedge v \in S\}$. We drop the subscript G from $N_G(S)$ when the context is clear. For a vertex subset $S \subseteq V(G)$, by $G[S]$ we denote the graph with vertex set S and edge set $\{(u, v) \in E(G) \mid u, v \in S\}$. By $G - S$ we denote the graph $G[V(G) \setminus S]$. For $X, Y \subseteq V(G)$, an (X, Y)-path in G is a path $v_1, v_2, \ldots, v_\ell$ such that $v_1 \in X$ and $v_\ell \in Y$. We say that X and Y are *linked* in G if there exists an (X, Y)-path in G. We say that vertices in Y are *reachable* from X if, for all $y \in Y$, there exists $x \in X$ such that there is a path from x to y.

A vertex subset $S \subseteq V(G)$ is a *feedback vertex set (fvs)* in G if $G - S$ is a forest. If there is no $S' \subset S$ such that $G - S'$ is a forest then S is a *minimal feedback vertex set (minimal fvs)* in G. A vertex subset $S \subseteq V(G)$ is an *odd cycle transversal (oct)* in G if $G - S$ is bipartite. If there is no $S' \subset S$ such that $G - S'$ is a bipartite graph then S is a *minimal odd cycle transversal (minimal oct)* in G. For a graph G and set $X \subseteq V(G)$, we refer to a partition (A, B) of X as a *valid bipartition* of $G[X]$ if $G[A]$ and $G[B]$ are both edgeless graphs. We refer to a valid bipartition of $V(G)$ as a valid bipartition of the graph G.

▶ **Definition 2.1.** Let G be a graph and X and Y be disjoint subsets of $V(G)$. A vertex set S disjoint from $X \cup Y$ is called an *(X,Y)-separator* if there is no (X, Y)-path in $G - S$. We denote by $R_G(X, S)$ the set of vertices of $G - S$ reachable from vertices of X via paths and by $NR_G(X, S)$ the set of vertices of $G - S$ *not* reachable from vertices of X.

▶ **Definition 2.2.** [13] A *compact representation* of a set $\mathcal{S}$ of minimal feedback vertex sets of a graph G is a collection $\mathcal{C}$ of pairwise disjoint subsets of $V(G)$ such that choosing exactly one vertex from every set in $\mathcal{C}$ results in a minimal feedback vertex set for G that is in $\mathcal{S}$.

▶ **Lemma 2.3.** [13] *The set of all minimal feedback vertex sets of size at most k can be represented by a collection of compact representations of size $2^{\mathcal{O}(k)}$. Furthermore, given a* *graph $G = (V, E)$ and a feedback vertex set F for G of size $k + 1$, we can enumerate the compact representations of all minimal feedback vertex sets for G having size at most k in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time.*

3 From Simultaneous FVS/OCT to Colorful OCT

We first describe how to reduce an instance of Simultaneous FVS/OCT to $2^{\mathcal{O}(\alpha k)}$ instances of Colorful OCT. Note that since both Feedback Vertex Set [20] and Odd Cycle Transversal [22, 14] can be solved in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time, we assume that, along with an instance $(G = (V, \cup_{i=1}^{\alpha} E_i), k)$, we are given sets $O, F_2, \ldots, F_\alpha \subseteq V(G)$ of size at most k such that $G_1 - O$ is a bipartite graph and $G_i - F_i$, $2 \leq i \leq \alpha$, is acyclic (as otherwise we can safely conclude that the given instance is a no-instance).

▶ **Lemma 3.1.** *There is an algorithm that, given an instance $(G = (V, \cup_{i=1}^{\alpha} E_i), k)$ of* Simultaneous FVS/OCT*, runs in time $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ and returns a set of $2^{\mathcal{O}(\alpha k)}$ instances of* Colorful OCT *such that the original instance is a yes-instance if and only if at least one of the returned instances is a yes-instance.*

Proof. Armed with the sets F_i which are of size at most k, we apply the algorithm of Lemma 2.3 to each graph G_i, $2 \leq i \leq \alpha$, to obtain a set of compact representations $\mathbf{C}_i = \{\mathcal{C}_i^1, \mathcal{C}_i^2, \ldots\}$, $2 \leq i \leq \alpha$. Note that each $\mathbf{C}_i$ is of size $2^{\mathcal{O}(k)}$ and each $\mathcal{C}_i^j$ is of size at most k. The said algorithm runs in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time for each graph G_i. For each tuple $\{\mathcal{C}_2^{j_2}, \ldots, \mathcal{C}_\alpha^{j_\alpha}\} \in \mathbf{C}_2 \times \ldots \times \mathbf{C}_\alpha$, we construct an instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT as follows. We let $G' = (V, E_1)$, $k' = k$, and $\ell = \sum_{i=2}^{\alpha} |\mathcal{C}_i^j| \leq \alpha k$.

For each $\mathcal{C} \in \{\mathcal{C}_2^{j_2}, \ldots, \mathcal{C}_\alpha^{j_\alpha}\}$ and for each set $C \in \mathcal{C}$, we add a set $P \in \mathcal{P}$ and we let $P = C$. In other words, all vertices in C are added to P. Observe that $|\mathcal{C}| \leq k$. Since each $\mathbf{C}_i$ is of size $2^{\mathcal{O}(k)}$, it is easy to verify that the number of instances is in fact $2^{\mathcal{O}(\alpha k)}$. We now prove the correctness of the algorithm.

Assume that $(G = (V, \cup_{i=1}^{\alpha} E_i), k)$ is a yes-instance and let S be a solution of size at most k. Note that S need not be a minimal fvs in G_i, $2 \leq i \leq \alpha$. However, for each $i \in \{2, \ldots, \alpha\}$, there exists a set $S' \subseteq S$ such that S' is a minimal fvs for G_i. Hence, by Definition 2.2 and Lemma 2.3, for every $i \in \{2, \ldots, \alpha\}$, there exists a $\mathcal{C}_i^j \in \mathbf{C}_i$ such that for all $C \in \mathcal{C}_i^j$ we have $S' \cap C \neq \emptyset$. Since we enumerate all compact representations and create one instance for each, we know that at least one instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT will correspond to the correct choice. The fact that S is a solution for $(G', \mathcal{P}, k', \ell)$ follows from the fact that S contains a minimal oct for G_1.

For the other direction, let S' be a solution for an instance $(G', \mathcal{P}, k', \ell)$ of Colorful OCT. Since S' is of size at most k, it is clearly an oct for G_1. Moreover, since S' must intersect every $P \in \mathcal{P}$, it follows from the definition of compact representations and our construction that S' is an fvs for G_i, $2 \leq i \leq \alpha$, as needed. ◀

We now focus on solving an instance $(G, \mathcal{P}, k, \ell)$ of Colorful OCT. Recall that we also have access to the set O which is an oct of G of size at most k. Our next step is to reduce $(G, \mathcal{P}, k, \ell)$ to an instance $(G, \mathcal{P}, O, k, \ell)$ of Colorful OCT Compression. The correctness of this reduction is immediate. The final piece in our sequence of reductions is to reduce $(G, \mathcal{P}, O, k, \ell)$ to $2^{\mathcal{O}(k)}$ instances of Colorful Separator.

▶ **Lemma 3.2.** *There is an algorithm that, given an instance $(G, \mathcal{P}, O, k, \ell)$ of* Colorful OCT Compression*, runs in time $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ and returns a set of $2^{\mathcal{O}(k)}$ instances of* Colorful Separator *such that the original instance is a yes-instance if and only if at least one of the returned instances is a yes-instance.*

To summarize, given an instance $(G = (V, \cup_{i=1}^{\alpha} E_i), k)$ of Simultaneous FVS/OCT, we first compute an odd cycle transversal of G_1 and a feedback vertex set of G_i, $i \in [\alpha] \setminus \{1\}$, in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time. Then, we generate $2^{\mathcal{O}(\alpha k)}$ instances of Colorful OCT, of the form $(G, \mathcal{P}, k, \ell \leq \alpha k)$, in $\mathcal{O}^\star(2^{\mathcal{O}(\alpha k)})$ time. Each instance of Colorful OCT is converted into an instance $(G, \mathcal{P}, O, k, \ell)$ of Colorful OCT Compression in polynomial time. Finally, for each instance of Colorful OCT Compression we generate $2^{\mathcal{O}(k)}$ instances of Colorful Separator, with parameters k and $\ell \leq \alpha k$, in $\mathcal{O}^\star(2^{\mathcal{O}(k)})$ time. Lemmas 3.1 and 3.2 together imply that if we can solve an instance of Colorful Separator in $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ time then the algorithm for Simultaneous FVS/OCT follows.

4 An FPT algorithm for finding colorful separators

We in fact give an algorithm for a more general problem, which we call Colorful Multiway Cut (or CMWC for short). Before we proceed, we need a few definitions.

▶ **Definition 4.1.** Given a graph G, a set $T \subseteq V(G)$, and a partition $\mathcal{T}$ of T into (pairwise disjoint) sets $\{T_1, \ldots, T_r\}$, we say that $S \subseteq V(G) \setminus T$ is a *$\mathcal{T}$-multiway cut* if, in $G - S$, no vertex in $T_i \setminus S$ can reach a vertex in $T_j \setminus S$, for all $i, j \in [r]$, such that $i \neq j$. We say that $\mathcal{T}$ is an *edge-free partition* of T if there are no edges (u, v) in $G[T]$ where u and v belong to different sets of $\mathcal{T}$.

Given a grouping $\{P_1, \ldots, P_\ell\}$ of the vertices of a graph G, we define a partial coloring function $\mathsf{col} : V(G) \to 2^{[\ell]}$. That is, we have $i \in \mathsf{col}(v)$ if and only if $v \in P_i$, for some $i \in [\ell]$. In this context, for a set $C \subseteq [\ell]$, a subset S of vertices of G is called *C-colorful* if, for each $i \in C$, there is a vertex v in S such that $i \in \mathsf{col}(v)$. For a subset $S \subseteq V(G)$, we denote by $\mathsf{col}(S)$ the set $\{j \mid v \in S \cap (\bigcup_{i=1}^{\ell} P_i) \wedge j \in \mathsf{col}(v)\}$, i.e. the set of colors appearing in S. The CMWC can now be defined as follows.

Colorful Multiway Cut (CMWC) **Parameter(s):** k, $|T|$, and ℓ
Input: A graph $G = (V, E)$, a set $T \subseteq V(G)$, a partition $\mathcal{T}$ of T into (pairwise disjoint) sets $\{T_1, \ldots, T_r\}$, a grouping $\mathcal{P}$ of the vertices of G into (not necessarily distinct) sets $\{P_1, \ldots, P_\ell\}$, a set $C \subseteq [\ell]$, and an integer k.
Question: Is there a set $S \subseteq V(G) \setminus T$ such that $|S| \leq k$, S is a $\mathcal{T}$-multiway cut in G, and S is C-colorful?

4.1 Setting up the algorithm

Let $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of CMWC. We start by stating a few simple reduction rules (which are applied in the order they are stated).

▶ **Reduction Rule 1.** *If $k < 0$ then return* false*, i.e. $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is a no-instance.*

▶ **Reduction Rule 2.** *If $k = 0$ and $\emptyset$ is a solution to $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ then return* true*, i.e. $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is yes-instance. If $k = 0$ and $\emptyset$ is not a solution then return* false*.*

▶ **Reduction Rule 3.** *If there exists $i \in C$ such that $P_i \subseteq T$ then return* false*.*

▶ **Reduction Rule 4.** *If there exists $i \in C$ such that $P_i \cap T \neq \emptyset$ then set $P_i = P_i \setminus T$.*

▶ **Reduction Rule 5.** *If there exists $i \in C$ such that $P_i = \emptyset$ then return* false*.*

▶ **Reduction Rule 6.** *If $\mathcal{T}$ is not an edge-free partition then return* false*.*

It is easy to see that Reduction Rules 1 to 6 are safe and can be applied in polynomial time. When $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut, we can solve the corresponding instance in time $\mathcal{O}^\star(2^{\mathcal{O}(\ell)})$. The following observation describes how.

▶ **Observation 1.** *Let $\mathcal{I} = (G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of* Colorful Multiway Cut*. If $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut then $\mathcal{I}$ can be solved in $\mathcal{O}(2^{\mathcal{O}(\ell)}n^2)$ time, where $n = |V(G)|$.*

Proof. If $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut then we are left with the problem of finding a set $S \subseteq V(G) \setminus T$ of size at most k such that $S \cap P_i \neq \emptyset$, for each $i \in C$. Hence, we construct a family $\mathcal{F}$ consisting of a set $f_{P_i} = P_i$ for each for each $i \in C$ and we let $\mathcal{U} = \cup_{i \in C} P_i$. Note that $|\mathcal{F}| \leq \ell \leq \alpha k$ and $|\mathcal{U}| \leq |V(G)|$. Since Reduction Rules 3, 4, and 5 are not applicable, for each $i \in C$, we have $f_{P_i} \neq \emptyset$ and $P_i \cap T = \emptyset$. If we can find a subset $U \subseteq \mathcal{U}$ which intersects all the sets in $\mathcal{F}$, such that $|U| \leq k$, then U is the required solution. Otherwise, we have a no-instance. It is known that the Hitting Set problem parameterized by the size of the family $\mathcal{F}$ is fixed-parameter tractable and can be solved in $\mathcal{O}(2^{\mathcal{O}(|\mathcal{F}|)}|\mathcal{U}|^2)$ time [7]. In particular, we can find an optimum hitting set $U \subseteq \mathcal{U}$, hitting all the sets in $\mathcal{F}$. Therefore, we have a subset of vertices that intersects all sets P_i, for $i \in C$. ◀

Before proceeding with the description of the algorithm, we first recall the notion of tight separator sequences introduced in [19]. However, the definition and structural lemmas regarding tight separator sequences used in this paper are from [20]. Note that although [20] contains Definition 4.2 and Lemma 4.3 in terms of directed graphs, the same holds true for undirected graphs because one can represent any undirected graph as a directed graph by adding bidirectional edges between every pair of adjacent vertices.

▶ **Definition 4.2.** Let X and Y be two subsets of $V(G)$ and let $k \in \mathbb{N}$. A *tight (X,Y)-reachability sequence* of order k is an ordered collection $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ of sets in $V(G)$ satisfying the following properties:

- $X \subseteq H_i \subseteq V(G) \setminus N[Y]$ for any $0 \leq i \leq q$;
- $X = H_0 \subset H_1 \subset H_2 \subset \ldots \subset H_q \subset H_{q+1} = V(G) \setminus Y$;
- H_i is reachable from X in $G[H_i]$ and every vertex in $N(H_i)$ can reach Y in $G - H_i$ (implying that $N(H_i)$ is a minimal (X,Y)-separator in G);
- $|N(H_i)| \leq k$ for every $1 \leq i \leq q$;
- $N(H_i) \cap N(H_j) = \emptyset$ for all $1 \leq i, j \leq q$ and $i \neq j$;
- For any $0 \leq i \leq q - 1$, there is no (X,Y)-separator S of size at most k where $S \subseteq H_{i+1} \setminus N[H_i]$ or $S \cap N[H_q] = \emptyset$ or $S \subseteq H_1$.

We let $Q_0 = X$, $Q_i = N(H_i)$, for $1 \leq i \leq q$, $Q_{q+1} = Y$, and $\mathcal{Q} = \{Q_0, Q_1, \ldots, Q_q, Q_{q+1}\}$. We call $\mathcal{Q}$ a *tight (X,Y)-separator sequence* of order k.

▶ **Lemma 4.3.** *(see [20]) There is an algorithm that, given a graph G on n vertices and m edges, subsets $X, Y \subseteq V(G)$ and $k \in \mathbb{N}$, runs in time $\mathcal{O}(k^2nm)$ and either correctly concludes that there is no (X,Y)-separator of size at most k in G or returns the sets $H_0, H_1, H_2 \setminus H_1, \ldots, H_q \setminus H_{q-1}, H_{q+1} \setminus H_q$ corresponding to a tight (X,Y)-reachability sequence $\mathcal{H} = \{H_0, H_1, \ldots, H_q, H_{q+1}\}$ of order k.*

Our algorithm will be a combination of dynamic programming over the sets Q_i, $0 \leq i \leq q + 1$, and recursive calls for solving "smaller" instances of the same problem. Below we state some observations that help understand the structure of a solution and are crucial for achieving the stated running time.

Algorithm 1: Pseudocode for **ALG1**

```
Input: $(G, T, \mathcal{T}, \mathcal{P}, C, k)$
Output: true or false
1 Apply all reduction rules (in order) and return true/false appropriately (if applicable).
2 if $k > 0$ and $\emptyset$ is a $\mathcal{T}$-multiway cut then
3     return true/false appropriately (Observation 1)
4 Let $T_1 \in \mathcal{T}$ such that $T_1$ is linked to some $T_j \in \mathcal{T}$, where $j \neq 1$.
5 Let $\mathcal{H} = \{H_0, H_1, \dots, H_q, H_{q+1}\}$ be a $(T_1, T \setminus T_1)$-reachability sequence of order $k$;
6 Let $\mathcal{Q} = \{Q_0, Q_1, \dots, Q_q, Q_{q+1}\}$ be the corresponding $(T_1, T \setminus T_1)$-separator sequence;
7 if $\mathcal{Q} = \emptyset$ then
8     return false;
9 return ALG2$(G, T, \mathcal{T}, \mathcal{P}, C, k, \mathcal{Q})$;
```

▶ **Observation 2.** *Let $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ be an instance of* Colorful Multiway Cut *and let T_1 be a set in $\mathcal{T}$ which is linked to some set in $\mathcal{T} \setminus \{T_1\}$. Moreover, let $\mathcal{H} = \{H_0, H_1, \dots, H_q, H_{q+1}\}$ be a tight $(T_1, T \setminus T_1)$-reachability sequence of order k and let $\mathcal{Q} = \{Q_0, Q_1, \dots, Q_q, Q_{q+1}\}$ be the corresponding tight separator sequence. Assume $(G, T, \mathcal{T}, \mathcal{P}, C, k)$ is a yes-instance and let S be one of its solution. Then, S can be partitioned into the following (pairwise-disjoint) sets.*

- $Z_1 = S \cap (H_1 \setminus Q_0)$.
- $S_i = S \cap Q_i$ *for* $1 \leq i \leq q$.
- $Z_i = (S \cap (H_i \setminus N[H_{i-1}])) \setminus Q_{q+1}$ *for* $2 \leq i \leq q+1$.

▶ **Observation 3.** $|Z_i| \leq k-1$ *for each* $i \in [q+1]$.

To keep the presentation clean, we shall define two routines **ALG1** and **ALG2**. **ALG1** (Algorithm 1) delegates most of the "heavy lifting" to **ALG2**. That is, **ALG1** simply checks if any of the reduction rules are applicable and solves the instance if it corresponds to one of the base cases. When this is not the case, **ALG1** proceeds by computing a tight separator sequence and calls **ALG2**. Note that we can safely return false when the algorithm fails to construct such a sequence (Lines 7 and 8 of Algorithm 1). We now move to the description of **ALG2**, which takes as additional input the newly constructed tight separator sequence. Roughly speaking, **ALG2** will recursively solve a "large" number of instances restricted to graphs that "reside" between two consecutive separators of a separator sequence. The number of instances will be bounded by the number of possible "interactions" between the two consecutive separators and a hypothetical solution. However, due to Observation 3, each one of those recursive calls can be made with a strictly smaller value of k. Having solved all such instances (and stored the outcomes in tables), **ALG2** then proceeds using a dynamic programming routine which computes the answer in a left-to-right manner, i.e. starting from Q_0 all the way to Q_{q+1}. We now give a formal description.

▶ **Definition 4.4.** For a graph G and a tight separator sequence $\mathcal{Q} = \{Q_0, Q_1, \dots, Q_q, Q_{q+1}\}$, we let $G_i = G - R_G(Q_{q+1}, Q_i)$, *i.e.* the graph obtained after removing the vertices that are reachable from Q_{q+1} after deleting Q_i, and we let $\widehat{G}_i = G_i - (V(G_{i-1}) \setminus Q_{i-1})$.

For each graph G_i, $i \in [q+1]$, we maintain a table Γ_i, where each entry is indexed by a tuple $(X, \mathcal{A}, \overline{C}, \overline{p})$. For each graph $\widehat{G}_i$, $i \in [q+1]$, we maintain a table Λ_i, where each entry is indexed by a tuple $(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$. The tuples are described below.

Algorithm 2: Pseudocode for **ALG2**

```
Input: $(G, T, \mathcal{T}, \mathcal{P}, C, k, \mathcal{Q})$
Output: true or false
1  Initialize all entries in $\Gamma_i$ to false, for $i \in [q+1]$;
2  Initialize all entries in $\Lambda_i$ to false, for $i \in [q+1]$;
3  for each $\widehat{G}_i \in \{\widehat{G}_1, \dots, \widehat{G}_{q+1}\}$ do
4      for each $L \subseteq Q_{i-1} \setminus T$ and each each $R \subseteq Q_i \setminus T$ do
5          for each edge-free partition $\mathcal{B}$ of $(Q_{i-1} \cup Q_i) \setminus (L \cup R)$ do
6              for each $\widehat{C} \subseteq [\ell]$ and each $0 \leq \widehat{p} \leq k - \max\{1, |L \cup R|\}$ do
7                  $\mathbb{I} = (\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}|_{V(G_i-(L\cup R))}, \widehat{p})$;
8                  $\Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p}) = \textbf{ALG1}(\mathbb{I})$;
9  Copy table entries for $\Gamma_1$, i.e. $\Gamma_1(X, \mathcal{A}, \overline{C}, \overline{p}) = \Lambda_1(\emptyset, X, \mathcal{A}, \overline{C}, \overline{p})$;
10 for each $G_i \in \{G_2, \dots, G_{q+1}\}$ (in order) do
11     for each $X \subseteq Q_i \setminus T$ do
12         for each edge-free partition $\mathcal{A}$ of $(Q_i \cup Q_0) \setminus X$ do
13             for each $\overline{C} \subseteq [\ell]$ and each $0 \leq \overline{p} \leq k - |X|$ do
14                 $\tau_1 = (X, \mathcal{A}, \overline{C}, \overline{p})$;
15                 for each tuple $\tau_2 = (L, R, \mathcal{B}, \widehat{C}, \widehat{p}) \in \Lambda_i$ do
16                     for each tuple $\tau_3 = (X', \mathcal{A}', \overline{C}', \overline{p}') \in \Gamma_{i-1}$ do
17                         if $\tau_1$, $\tau_2$, and $\tau_3$ are compatible then
18                             $\Gamma_i(\tau_1) = \Gamma_i(\tau_1) \vee [\Gamma_{i-1}(\tau_3) \wedge \Lambda_i(\tau_2)]$;
19 if $\Gamma_{q+1}(\emptyset, \mathcal{T}, C, p) =$ true (for some $p \leq k$) then
20     return true;
21 return false;
```

- $X \subseteq Q_i \setminus T$ and $L \subseteq Q_{i-1} \setminus T$ and $R \subseteq Q_i \setminus T$;
- $\mathcal{A}$ is an edge-free partition of $(Q_i \cup Q_0) \setminus X$;
- $\mathcal{B}$ is an edge-free partition of $(Q_{i-1} \cup Q_i) \setminus (L \cup R)$;
- $\overline{C}, \widehat{C} \subseteq [\ell]$ and $\overline{p} \leq k - |X|$ and $\widehat{p} \leq k - |L \cup R|$ if $L \cup R \neq \emptyset$ and $\widehat{p} \leq k - 1$, otherwise.

▶ **Definition 4.5.** For a tuple $\tau = (X, \mathcal{A}, \overline{C}, \overline{p})$, we denote by $\mathbb{I}_\tau$ the instance $(G_i - X, (Q_i \cup Q_0) \setminus X, \mathcal{A}, \mathcal{P}|_{V(G_i - X)}, \overline{C}, \overline{p})$ of CMWC. Similarly, for a tuple $\tau = (L, R, \mathcal{B}, \widehat{C}, \widehat{p})$, we denote by $\mathbb{I}_\tau$ the instance $(\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}|_{V(G_i - (L \cup R))}, \widehat{C}, \widehat{p})$ of CMWC. Finally, we define $\Gamma_i(\tau)$ (or $\Lambda_i(\tau)$)= true if and only if $\mathbb{I}_\tau$ is a yes-instance of CMWC.

▶ **Definition 4.6.** Given three tuples $\tau_1 = (X, \mathcal{A}, \overline{C}, \overline{p})$, $\tau_2 = (L, R, \mathcal{B}, \widehat{C}, \widehat{p})$, and $\tau_3 = (X', \mathcal{A}', \overline{C}', \overline{p}')$, we say that they are *compatible* if all of the following conditions hold.

- $\tau_1 \in \Gamma_i$ and $\tau_2 \in \Lambda_i$ and $\tau_3 \in \Gamma_{i-1}$, where $i \in [q+1]$;
- $X' = L$ and $X = R$;
- $\mathcal{A}|_{Q_i \setminus X} = \mathcal{B}|_{Q_i \setminus R}$ and $\mathcal{B}|_{Q_{i-1} \setminus L} = \mathcal{A}'|_{Q_{i-1} \setminus X'}$ and $\mathcal{A}|_{Q_0} = \mathcal{A}'|_{Q_0}$;
- $\overline{p}' + \widehat{p} + |L| \leq p$ and $\overline{C}' \cup \widehat{C} \cup \mathsf{col}(L) = \overline{C}$.

The complete description of **ALG2** is given in Algorithm 2. Initially, we set all table entries to false (Lines 1 and 2). Then, for each $\widehat{G}_i \in \{\widehat{G}_1, \dots, \widehat{G}_{q+1}\}$ and for each possible tuple $(L, R, \mathcal{B}, \widehat{C}, \widehat{p}) \in \Lambda_i$, we solve the corresponding CMWC instance $\mathcal{I} = (\widehat{G}_i - (L \cup R), (Q_{i-1} \cup Q_i) \setminus (L \cup R), \mathcal{B}, \mathcal{P}|_{V(G_i-(L\cup R))}, \widehat{p})$. That is, we set $\Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$ if $\mathcal{I}$ is a yes-instance (Lines 3 to 8). Having computed all those values, we then proceed to filling table Γ_1. Since G_0 is a subgraph of G_1, and $G_1 = \widehat{G}_1$, we simply set $\Gamma_1(X, \mathcal{A}, \overline{C}, \overline{p}) = \Lambda_1(\emptyset, X, \mathcal{A}, \overline{C}, \overline{p})$ (for all tuples). This is justified by the fact that a solution is not allowed to delete any vertex in Q_0. To complete table Γ_i, $i > 1$, we simply use the following:

$$\Gamma_i(X, \mathcal{A}, \overline{C}, \overline{p}) = \bigvee [\Gamma_{i-1}(X', \mathcal{A}', \overline{C}', \overline{p}') \wedge \Lambda_i(L, R, \mathcal{B}, \widehat{C}, \widehat{p})],$$

where tuples $(X, \mathcal{A}, \overline{C}, \overline{p})$, $(X', \mathcal{A}', \overline{C}', \overline{p}')$, and $(L, R, \mathcal{B}, \widehat{C}, \widehat{p})$ are compatible. Finally, **ALG2** returns true whenever there exists a tuple $\Gamma_{q+1}(\emptyset, \mathcal{T}, C, p) = \text{true}$ (for some $p \leq k$).

4.2 Correctness and runtime analysis

We are now ready to prove our main structural lemma which reduces the computation of the entries in Γ_i (when $i > 1$) to those in Γ_{i-1} and Λ_i. The lemma is proved in a purely existential setting and serves as the proof of correctness of the algorithm.

▶ **Lemma 4.7.** *For any $i \in [q+1]$ and tuple $\tau_1 = (X, \mathcal{A}, C_1, p_1) \in \Gamma_i$, $\mathbb{I}_{\tau_1}$ is a yes-instance if and only if there is a tuple $\tau_2 = (L, R, \mathcal{B}, C_2, p_2) \in \Lambda_i$ and a tuple $\tau_3 = (X', \mathcal{A}', C_3, p_3) \in \Gamma_{i-1}$ such that $\mathbb{I}_{\tau_2}$ and $\mathbb{I}_{\tau_3}$ are both yes-instances and all three tuples are compatible.*

▶ **Theorem 4.8.** Colorful Multiway Cut *can be solved in $\mathcal{O}^\star((k+t)^{\mathcal{O}(kt+k^3)} 2^{\mathcal{O}(\ell k)})$ time, where $t = |T|$.*

▶ **Corollary 4.9.** Simultaneous FVS/OCT *can be solved in $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ time.*

Proof. Recall that Lemmas 3.1 and 3.2 together imply that if we can solve an instance of Colorful Separator in $\mathcal{O}^\star(k^{\mathsf{poly}(\alpha,k)})$ time then the algorithm for Simultaneous FVS/OCT follows. Any instance of Colorful Separator can be reduced to an instance of Colorful Multiway Cut with $|T| = 2$. From Theorem 4.8, such an instance can be solved in time $\mathcal{O}^\star(k^{\mathcal{O}(k^3)} 2^{\mathcal{O}(\alpha k)})$. ◀

5 W[1]-hardness of Simultaneous OCT

In this section we show that Simultaneous OCT is W[1]-hard. For notational convenience, we shall use a different encoding of α-edge-colored graphs. Given a graph G with vertex set $V(G)$ and edge set $E(G)$, we define a coloring function $\mathsf{col}(e) \subseteq 2^{[\alpha]}$. In particular, when $\alpha = 2$, we have $\mathsf{col}(e) \subseteq 2^{\{1,2\}}$. We start by establishing W[1]-hardness of Simultaneous Cut, which is formally defined below.

Simultaneous Cut **Parameter(s):** k and α
Input: A graph G, two vertices $s, t \in V(G)$, an integer k, and a coloring function $\mathsf{col} : E(G) \to 2^{[\alpha]}$.
Question: Is there $X \subseteq V(G) \backslash \{s, t\}$ of size at most k such that, for all $i \in [\alpha]$, $G_i - X$ has no (s,t)-paths? Here, for $i \in [\alpha]$, $G_i = (V(G), E_i)$, where $E_i = \{e \in E(G) \mid i \in \mathsf{col}(e)\}$.

We give a parameterized reduction from Multicolored Clique which is known to be W[1]-hard [7]. The Multicolored Clique problem is formally defined below.

Multicolored Clique **Parameter(s):** k
Input: A k-partite graph G with a partition $V_1, V_2, \ldots, V_k$ of $V(G)$ such that for all $i, j \in [k]$, $|V_i| = |V_j|$.
Question: Is there $X \subseteq V(G)$ such that, for all $i \in [k]$, $|X \cap V_i| = 1$ and $G[X]$ is a clique?

Given an instance $(G, V_1, V_2, \ldots, V_k)$ of Multicolored Clique, we proceed by creating an instance $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ of Simultaneous Cut such that $(G, V_1, V_2, \ldots, V_k)$ is a yes-instance of Multicolored Clique if and only if $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ is a yes-instance of Simultaneous Cut.

The intuitive description of the parameterized reduction is as follows. Let $(G, V_1, V_2, \ldots, V_k)$ be an instance of Multicolored Clique. Since $|V_i| = |V_j|$, for all $i, j \in [k]$, we assume that $|V_i| = |V_j| = n$. Furthermore, we assume that for every $i, j \in [k]$, $i \neq j$, there is at least one edge between V_i and V_j, otherwise, the instance is a trivial no-instance of Multicolored Clique and our reduction will simply output a trivial no-instance of Simultaneous Cut with $\alpha = 2$. For each $i \in [k]$ we assume an arbitrary (but fixed) ordering on the vertices in V_i. For each $i \in [k]$, we will have a vertex selection gadget $\mathcal{S}_i$ that will be responsible for selecting a vertex in V_i. To achieve this, $\mathcal{S}_i$ will have $k-1$ copies of each vertex in V_i, so that each vertex in V_i has a copy corresponding to every $j \in [k] \setminus \{i\}$. For each $j \in [k] \setminus \{i\}$, we have an ($s$,$t$)-path with all edges having color 1. Each path contains exactly one copy of every vertex in V_i. Furthermore, these vertices appear in the order given by the ordering we already fixed on the vertices of V_i.

The jth copy of the vertex set V_i will be used to ensure that there is an edge between the selected vertex in V_i and a vertex in V_j. The copies of any single vertex will form an (s,t)-separator of size $k-1$. Furthermore, the size of minimum (s,t)-separator in $\mathcal{S}_i$ will be $k-1$ and there will be exactly n distinct minimum separator each of which will correspond to a set comprising of $k-1$ copies of a vertex in V_i. By construction of the gadget and by setting budget constraints appropriately we will ensure that we must select a vertex from each of the $k-1$ copies of V_i, for each $i \in [k]$ and the selected $k-1$ vertices correspond to copies of the same vertex, i.e. we select a minimum separator. This will ensure that we have selected exactly one vertex from each V_i, for $i \in [k]$.

For $i, j \in [k]$, $i \neq j$, we will have edge selection gadgets E_{ij} which will ensure that there is an edge selected between V_i and V_j, and the selected edge is incident to the vertex selected from the vertex selection gadget. Finally, we will have a compatibility gadget which will ensure that the edges selected by E_{ij} and E_{ji} correspond to the same edge in G. We need to differentiate between gadgets E_{ij} and E_{ji} for technical reasons that will become clear later. We will now move to the formal description of the reduction.

Construction. Initially, $V(G') = \emptyset$ and $E(G') = \emptyset$. We add two special vertices s and t to $V(G')$, which are the vertices we want to separate, and which will be common to all the gadgets. For $i \in [k]$ we let v^i_j be the jth vertex in V_i. We now formally describe the construction of the various gadgets. We note that the gadgets are not necessarily vertex or edge disjoint (in addition to intersecting with $\{s, t\}$).

Vertex Selection Gadget. For each $i \in [k]$ we have a vertex selection gadget $\mathcal{S}_i$ defined as follows. For each $j \in [k] \setminus \{i\}$, $\mathcal{S}_i$ contains vertices in $V_{ij} = \{v^i_{j1}, v^i_{j2}, \ldots, v^i_{jn}\}$. Here, the vertices $v^i_{j1}, v^i_{j2}, \ldots, v^i_{jn}$ corresponds to one copy of the vertices $v^i_1, v^i_2, \ldots v^i_n$ in V_i. Note that for $j, j' \in [k] \setminus \{i\}$ vertices $v^i_{j\ell}, v^i_{j'\ell}$ correspond to copies of the same vertex, namely $v^i_\ell \in V_i$. For $i \in [k]$ and $\ell \in [n]$, we let $V^i_\ell = \{v^i_{j\ell} \mid j \in [k] \setminus \{i\}\}$, i.e. V^i_ℓ denotes the set comprising of

$k-1$ copies of the vertex $v_\ell^i \in V_i$. For $i \in [k]$, $\ell \in [n-1]$, and for each $u \in V_\ell^i$ and $u' \in V_{\ell+1}^i$ we add the edge $(u,u') \in E(G')$ and set $\mathsf{col}'((u,u')) = \{1\}$. Note that $G'[V_\ell^i \cup V_{\ell+1}^i]$ is a complete bipartite graph with all edges having the color 1 in their color set. For $i \in [k]$, $u \in V_1^i$ we add the edge $(s,u) \in E(G')$ and set $\mathsf{col}'((s,u)) = \{1\}$. Similarly, for $i \in [k]$, $u \in V_n^i$ we add the edge $(u,t) \in E(G')$ and set $\mathsf{col}'((u,t)) = \{1\}$.

Edge Selection Gadget. For $i \in [k]$ and $j \in [k] \setminus \{i\}$ the edge selection gadget $\mathcal{E}_{ij}$ is constructed as follows. The vertex set of $\mathcal{E}_{ij}$ contains a vertex $e_{uu'}$, for each edge $(u,u') \in E(G)$ with $u \in V_i$ and $u' \in V_j$. We note here that $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ denote distinct gadgets. For $\ell \in [n]$, we let $E_\ell^{ij} = \{e_{v_\ell^i u'} \mid u' \in V_j, (v_\ell^i, u) \in E(G)\}$, i.e. E_ℓ^{ij} contains vertices corresponding to those edges between V_i and V_j that are incident to the vertex $v_\ell^i \in V_i$. We let $E_{ij} = \cup_{\ell \in [n]} E_\ell^{ij}$. For $\ell \in [n]$ and each $u \in E_\ell^{ij}$, we add the edge $(u, v_{j\ell}^i)$ to $\mathcal{E}_{ij}$. We add an induced path P_ℓ^{ij} on the vertices in E_ℓ^{ij} (where the vertices appear in the natural order implied by the ordering of the vertices in V_j) and add these edges to $\mathcal{E}_\ell^{ij}$. For each edge $e \in E(P_\ell^{ij})$, we let $\mathsf{col}'(e) = \{2\}$. For $\ell \in [n+1]$, we let $\mathbf{K}_\ell^{ij}$ denote a $K_{3,3}$ (complete bipartite graph with 3 vertices on both side) with vertex bipartition $(\{p_\ell^{ij}, q_\ell^{ij}, r_\ell^{ij}\}, \{\bar{p}_\ell^{ij}, \bar{q}_\ell^{ij}, \bar{r}_\ell^{ij}\})$ and add it to $\mathcal{E}_{ij}$. We will refer to $\mathbf{K}_\ell^{ij}$s as *barrier blocks* of $\mathcal{E}_{ij}$. Finally, we join s, t and E_ℓ^{ij}, for $\ell \in [n]$ using the barrier blocks. This is done as follows.

For $\ell \in [n]$, let a_ℓ^{ij}, b_ℓ^{ij} be the first and the last vertex respectively, in the path P_ℓ^{ij}. We add the edges $(a_\ell^{ij}, \bar{p}_\ell^{ij}), (a_\ell^{ij}, \bar{q}_\ell^{ij}), (a_\ell^{ij}, \bar{r}_\ell^{ij})$ and $(b_\ell^{ij}, p_{\ell+1}^{ij}), (b_\ell^{ij}, q_{\ell+1}^{ij}), (b_\ell^{ij}, r_{\ell+1}^{ij})$ to $E(\mathcal{E}_{ij})$. Also, for $\ell \in [n]$, we add the edges $(v_{j\ell}^i, \bar{p}_\ell^{ij}), (v_{j\ell}^i, \bar{q}_\ell^{ij}), (v_{j\ell}^i, \bar{r}_\ell^{ij})$ and $(v_{j\ell}^i, p_{\ell+1}^{ij}), (v_{j\ell}^i, q_{\ell+1}^{ij}), (v_{j\ell}^i, r_{\ell+1}^{ij})$ to $E(\mathcal{E}_{ij})$. In addition, we add the edges $(s, p_1^{ij}), (s, q_1^{ij}), (s, \bar{r}_1^{ij})$, $(\bar{p}_{n+1}^{ij}, t), (\bar{q}_{n+1}^{ij}, t), (\bar{r}_{n+1}^{ij}, t)$ to $\mathcal{E}_{ij}$. For each $e \in E(\mathcal{E}_{ij})$, we set $\mathsf{col}'(e) = \{2\}$. This completes the description of the edge selection gadget.

Edge Compatibility Gadget. This gadget is used to ensure that the edge selected by $\mathcal{E}_{ij}$ and $\mathcal{E}_{ji}$ corresponds to the same edge of G. For $i,j \in [k]$, $i<j$, the edge compatibility gadget $\mathcal{C}_{ij}$ is constructed as described below. Basically, $\mathcal{C}_{ij}$ comprises of a set of edges between vertices in E_{ij} and vertices in E_{ji}. Recall that E_{ij} and E_{ji} contains vertices corresponding to the same edges, namely the edges between V_i and V_j in G. Hence, we can think of E_{ji} as a set comprising of a copy of the vertices in E_{ij}. We fix a lexicographic ordering on vertices in E_{ij} which we obtain as follows. For $e_{v_a^i, v_x^j}, e_{v_b^i, v_y^j} \in E_{ij}$, $e_{v_a^i, v_x^j} < e_{v_b^i, v_y^j}$ if (i) $a<b$ or (ii) $a=b$ and $x<y$. We denote the ordering of vertices in E_{ij} by $e_1^{ij}, e_2^{ij}, \ldots, e_m^{ij}$. Note this also fixes an ordering of vertices in E_{ji} which we denote by $e_1^{ji}, e_2^{ji}, \ldots, e_m^{ji}$. Here, m is the number of edges between V_i and V_j in G. For $\ell \in [m-1]$, we add the edges $(e_\ell^{ij}, e_{\ell+1}^{ij}), (e_\ell^{ij}, e_{\ell+1}^{ji}), (e_\ell^{ji}, e_{\ell+1}^{ij}), (e_\ell^{ji}, e_{\ell+1}^{ji})$ to $\mathcal{C}_{ij}$. That is we add all the edges in the bipartition between each consecutive pair of vertices in the ordered sets E_{ij} and E_{ji}. We add edges $(s, e_1^{ij}), (s, e_1^{ji})(e_m^{ij}, t), (e_m^{ji}, t)$ to $\mathcal{C}_{ij}$. For each edge $e \in \mathcal{C}_{ij}$, we set $\mathsf{col}'(e) = \{1\}$. We note here that in case we have created multiple edges say e, e' between vertices u, v then we delete e' and set $\mathsf{col}'(e) := \mathsf{col}'(e) \cup \mathsf{col}'(e')$.

We finally set $k' = k(k-1) + 2\binom{k}{2}$. We denote the graph constructed above by G' with the coloring function on the edge set denoted by col'.

▶ **Lemma 5.1.** $(G, V_1, V_2, \ldots, V_k)$ *is a yes-instance of* Multicolored Clique *if and only if* $(G', s, t, k', \mathsf{col}' : E(G') \to 2^{\{1,2\}})$ *is a yes-instance of* Simultaneous OCT.

▶ **Theorem 5.2.** *For all* $\alpha \geq 2$, Simultaneous Cut *is* W[1]-hard *when parameterized by* k. *Here,* α *is the number of colors in the coloring function of the edge set.*

We give a parameterized reduction from Simultaneous Cut to Simultaneous OCT, which implies the following theorem.

▶ **Theorem 5.3.** *For all $\alpha \geq 2$,* Simultaneous OCT *is* W[1]-hard *when parameterized by k. Here, α is the number of colors in the coloring function of the edge set.*

6 Conclusion

In light of Theorem 4.8, it is natural to ask whether one can improve the running time of our algorithm for Colorful Multiway Cut. In particular, is it possible to solve the problem in $\mathcal{O}^\star(k^{\mathcal{O}(k)})$ time when the number of terminals is constant and the number of colors is at most k? Another interesting question which remains open is whether the Simultaneous FVS/OCT problem admits a (randomized) polynomial kernel. Finally, we would also like to point out another interesting consequence of Theorem 5.3, i.e. the fact that Simultaneous OCT is W[1]-hard when parameterized by k. If we replace minimal feedback vertex sets by minimal odd cycle transversals in Lemma 2.3 then Theorem 5.3 implies that such a lemma cannot be true.

Acknowledgements. The authors are thankful to Saket Saurabh for helpful discussions.

References

1 Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous feedback vertex set: A parameterized perspective. In *33rd Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 7:1–7:15, 2016.

2 Leizhen Cai and Junjie Ye. Dual connectedness of edge-bicolored graphs and beyond. In *39th International Symposium on Mathematical Foundations of Computer Science*, pages 141–152, 2014.

3 Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015.

4 Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016.

5 Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved Algorithms for Feedback Vertex Set Problems. *Journal of Computer and System Sciences*, 74(7):1188–1198, 2008.

6 Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved Upper Bounds for Vertex Cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.

7 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

8 Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset Feedback Vertex Set Is Fixed-Parameter Tractable. *SIAM Journal of Discrete Mathematics*, 27(1):290–309, 2013.

9 Reinhard Diestel. *Graph Theory*. Springer-Verlag, Heidelberg, 4th edition, 2010.

10 Rod G. Downey and Michael R. Fellows. *Parameterized complexity*. Springer-Verlag, 1997.

11 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-Deletion: Approximation, Kernelization and Optimal FPT Algorithms. In *53rd Annual IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 470–479, 2012.

12 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Transactions on Algorithms*, 13(2):29:1–29:32, 2017.

13 Jiong Guo, Jens Gramm, Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *Journal of Computer and System Sciences*, 72(8):1386–1396, 2006.

14 Yoichi Iwata, Keigo Oka, and Yuichi Yoshida. Linear-time FPT algorithms via network flow. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1749–1761, 2014.

15 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1802–1811, 2014.

16 Eun Jung Kim, Alexander Langer, Christophe Paul, Felix Reidl, Peter Rossmanith, Ignasi Sau, and Somnath Sikdar. Linear kernels and single-exponential algorithms via protrusion decompositions. *ACM Transactions on Algorithms*, 12(2):21:1–21:41, 2016.

17 Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic Feedback Vertex Set. *Information Processing Letters*, 114(10):556–560, 2014.

18 Daniel Lokshtanov, N. S. Narayanaswamy, Venkatesh Raman, M. S. Ramanujan, and Saket Saurabh. Faster Parameterized Algorithms Using Linear Programming. *ACM Transactions on Algorithms*, 11(2):15:1–15:31, 2014.

19 Daniel Lokshtanov and M. S. Ramanujan. Parameterized Tractability of Multiway Cut with Parity Constraints. In *39th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 750–761, 2012.

20 Daniel Lokshtanov, M. S. Ramanujan, and Saket Saurabh. A linear time parameterized algorithm for directed feedback vertex set. *ArXiv e-prints*, 2016. `arXiv:1609.04347`.

21 Dániel Marx, Barry O'Sullivan, and Igor Razgon. Finding small separators in linear time via treewidth reduction. *ACM Transactions on Algorithms*, 9(4):30:1–30:35, 2013.

22 M. S. Ramanujan and Saket Saurabh. Linear time parameterized algorithms via skew-symmetric multicuts. In *25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1739–1748, 2014.

23 Bruce Reed, Kaleigh Smith, and Adrian Vetta. Finding odd cycle transversals. *Operations Research Letters*, 32(4):299–301, 2004.

24 Junjie Ye. A note on finding dual feedback vertex set. *ArXiv e-prints*, 2015. `arXiv:1510.00773`.