Strong Parameterized Deletion: Bipartite Graphs

Ashutosh Rai[1] **and M. S. Ramanujan**[2]

1 **The Institute of Mathematical Sciences, HBNI, Chennai, India**
ashutosh@imsc.res.in
2 **Vienna Institute of Technology, Vienna, Austria**
ramanujan@ac.tuwien.ac.at

Abstract

The purpose of this article is two fold: (a) to formally introduce a stronger version of graph deletion problems; and (b) to study this version in the context of bipartite graphs. Given a family of graphs $\mathcal{F}$, a typical instance of parameterized graph deletion problem consists of an undirected graph G and a positive integer k and the objective is to check whether we can delete at most k vertices (or k edges) such that the resulting graph belongs to $\mathcal{F}$. Another version that has been recently studied is the one where the input contains two integers k and ℓ and the objective is to check whether we can delete at most k vertices *and* ℓ edges such that the resulting graph belongs to $\mathcal{F}$. In this paper, we propose and initiate the study of a more general version which we call *strong deletion*. In this problem, given an undirected graph G and positive integers k and ℓ, the objective is to check whether there exists a vertex subset S of size at most k such that *each connected component* of $G - S$ can be transformed into a graph in $\mathcal{F}$ by deleting at most ℓ edges. In this paper we study this stronger version of deletion problems for the class of bipartite graphs. In particular, we study Strong Bipartite Deletion, where given an undirected graph G and positive integers k and ℓ, the objective is to check whether there exists a vertex subset S of size at most k such that each connected component of $G - S$ can be made bipartite by deleting at most ℓ edges. While fixed-parameter tractability when parameterizing by k or ℓ alone is unlikely, we show that this problem is fixed-parameter tractable (FPT) when parameterized by both k and ℓ.

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

Keywords and phrases fixed parameter tractable, bipartite-editing, recursive understanding

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2016.21

1 Introduction

Graph-modification by either deleting vertices or deleting edges or adding edges such that the resulting graph satisfies certain properties or becomes a member of some well-understood graph class is one of the basic problems in graph theory and graph algorithms. However, most of these problems are NP–complete [18, 26] and thus they are subjected to intensive study in algorithmic paradigms that are meant for coping with NP-completeness [9, 10, 21, 22]. These paradigms among others include applying restrictions on inputs, approximation algorithms and parameterized complexity. The goal of this paper is to introduce a 'stronger' notion of graph deletion in the realm of parameterized complexity and illustrate the difficulties that arise when considering the family of bipartite graphs and provide an approach to obtain fixed-parameter tractability by overcoming these difficulties.

A typical instance of parameterized graph deletion is of the following form. Let $\mathcal{F}$ be a family of graphs – such as edgeless graphs, forests, cluster graphs, chordal graphs, interval graphs, bipartite graphs, split graphs or planar graphs. The deletion problem corresponding to $\mathcal{F}$ is formally stated as follows.

36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2016).
Editors: Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen; Article No. 21; pp. 21:1–21:14

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

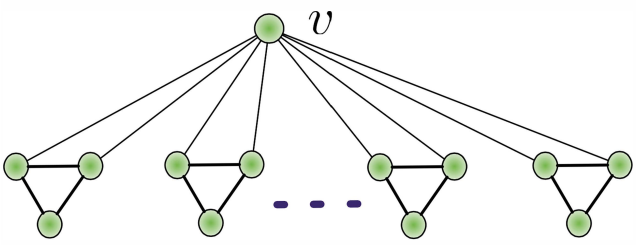

Figure 1 An example showing the strength of the new parameter for editing problems.

$\mathcal{F}$-Vertex (Edge) Deletion	**Parameters:** k
Input: An undirected graph G and a non-negative integer k.	
Question: Does there exist $S \subseteq V(G)$ (or $S \subseteq E(G)$), such that $\lvert S\rvert \leq k$ and $G - S$ is in $\mathcal{F}$?	

In other words, given a graph G, can we delete at most k vertices or k edges such that the resulting graph belongs to $\mathcal{F}$? An algorithm for $\mathcal{F}$-Vertex (Edge) Deletion that runs in time $f(k) \cdot |V(G)|^{\mathcal{O}(1)}$ is called *fixed-parameter tractable* (FPT) algorithm and the problem itself is said to be FPT.

The study of parameterized graph deletion problems together with their various restrictions and generalizations has been an extremely active sub-area over the last few years. In fact, just over the course of the last couple of years there have been results on parameterized algorithms for Chordal Editing [6], Unit Vertex (Edge) Deletion [3], Interval Vertex (Edge) Deletion [5, 4], Planar $\mathcal{F}$ Deletion [9, 17], Planar Vertex Deletion [14], Block Graph Deletion [16] and Simultaneous Feedback Vertex Set [1]. Several known parameterized algorithms for $\mathcal{F}$-Edge Deletion or the version where the objective is to delete k vertices and ℓ edges utilize the fact that given a yes-instance (G, k) or (G, k, ℓ) to the problem, there exists a vertex set S^* of $V(G)$ of size $k^* = k + \ell$ such that $G - S^*$ belongs to $\mathcal{F}$. Clearly, this is true for any hereditary family of graphs; that is, if $G \in \mathcal{F}$ then all its induced subgraphs belong to $\mathcal{F}$. Thus, if the corresponding $\mathcal{F}$-Vertex Deletion is FPT then we can apply this algorithm and find a vertex subset S^* of size at most k such that $G - S^* \in \mathcal{F}$. Having the set S^* allows one to infer numerous structural properties of the input which can then be utilized in non-trivial ways to solve the original $\mathcal{F}$-Vertex (Edge) Deletion problem. However, the existence of such a set is no longer guaranteed in the proposed stronger version of this problem.

Let $\mathcal{F}$ be a polynomial-time recognizable family of graphs; that is, given a graph G, in polynomial time we can decide whether G belongs to $\mathcal{F}$. For a fixed integer ℓ, let $\mathcal{F} + \ell e$ denote the class of graphs that can be obtained by adding ℓ edges to a graph in $\mathcal{F}$ [2]. Furthermore, suppose that $\mathcal{F}$-Edge Deletion is FPT with running time $\mathcal{O}(g(\ell) \cdot n^c)$. Here, $n = |V(G)|$ and c is a fixed constant. That is, for any fixed integer ℓ, $\mathcal{F} + \ell e$ can be recognized in time $\mathcal{O}(n^c)$. The Strong $\mathcal{F}$-Deletion problem is defined as follows.

Strong $\mathcal{F}$-Deletion	**Parameters:** k, ℓ
Input: An undirected graph G and non-negative integers k and ℓ.	
Question: Does there exist $S_1 \subseteq V(G)$ such that $\lvert S_1\rvert \leq k$ and every *connected component* of $G - S_1$ belongs to $\mathcal{F} + \ell e$?	

A close inspection easily shows that Strong $\mathcal{F}$-Deletion is much stronger than $\mathcal{F}$-Deletion. For example, let $\mathcal{F}$ be the family of bipartite graphs and consider the graph

G depicted in Figure 1. The graph G has $\frac{n-1}{3}$ vertex disjoint triangles and the vertex v is adjacent to two vertices from each of the triangles. Clearly, after deleting v every connected component can be made bipartite by deleting exactly one edge. Thus, in the traditional sense this is a yes-instance of Bipartite Deletion with parameters $(1, \frac{n-1}{3})$; however this is already a yes-instance of Strong Bipartite Deletion with parameters $(1, 1)$. Thus, these problems seem much harder than traditional editing problems as the parameters are much smaller.

An alternate viewpoint is that when G has a vertex set S of size at most k such that every connected component of $G - S$ is in $\mathcal{F} + \ell e$ then S can be considered a *modulator* into the graph class where every connected component of the graph belongs to $\mathcal{F} + \ell e$. While modulators to various polynomial-time recognizable graph classes have been studied in a very systematic way [8, 9, 13], the same is not true of modulators to NP-complete graph classes. Studying the Strong $\mathcal{F}$-Deletion problem on the other hand allows us to do precisely this. In fact, it is not even necessary that the class $\mathcal{F}$ is a conventional graph class and instead can be the 'composition' of various graph classes. For instance $\mathcal{F}$ could be defined as the set of all graphs where each connected component is either chordal *or* a bipartite graph. The computation of such modulators is a problem with several algorithmic applications. For instance, in a recent work, Ganian et al. [11] used a similar notion in order to design algorithms for the classic Constraint Satisfaction Problem.

In this article we study the Strong $\mathcal{F}$-Deletion, when $\mathcal{F}$ is the family of bipartite graphs. Henceforth, $\mathcal{F}$ denotes the family of bipartite graphs. We call a graph G, *ell-pseudobipartite*, if every connected component of G belongs to $\mathcal{F} + \ell e$. The problem we study is as follows.

Strong Bipartite Deletion	**Parameters:** k, ℓ
Input: An undirected graph G and non-negative integers k and ℓ.	
Question: Does there exist $S \subseteq V(G)$ such that $\lvert S\rvert \leq k$ and every connected component of $G - S$ belongs to $\mathcal{F} + \ell e$?	

We refer to the set S as the *solution* for this instance. The primary reason behind the selection of the family of bipartite graphs for our study is that the problems where we are required to delete vertices and/or edges to obtain a bipartite graph are some of the most basic and well studied problems in parameterized complexity and studying these problems has led to the discovery of several new techniques and tools. These problems are called Odd Cycle Transversal and Edge Bipartization in literature and the algorithms with best dependence on the parameter have running time $\mathcal{O}(2.3146^k n^{\mathcal{O}(1)})$ and $\mathcal{O}(1.977^k n^{\mathcal{O}(1)})$, respectively [19, 24]. We would like to add that the problem is unlikely to be FPT when parameterized by k or ℓ alone, as it would imply polynomial time algorithms for Edge Bipartization and Odd Cycle Transversal respectively. It is also important to mention that strong deletion to the class of forests, Strong Forest Deletion, was studied in related but different context in [23]. In that work, the authors used the concept of graph minors and Courcelle's theorem to show that the strong deletion problem is FPT. As is the case with algorithms using Courcelle's theorem, the dependence on the parameter is very high.

Our Result and Methodology. Our main result is the following theorem.

▶ **Theorem 1.** Strong Bipartite Deletion *is* FPT.

The first big obstacle that needs to be overcome is the fact that the input graph can have a minimum odd cycle transversal of unbounded size. The first part of our algorithm is devoted

to overcoming this obstacle. We utilise the technique of iterative compression to reduce it to a bounded number of equivalent instances each having an odd cycle transversal of size $poly(k, \ell)$. Then we use the recursive understanding technique introduced by Grohe et al. [12] (also see [7]) to first find a small separator in the graph which separates the graph into two parts, each of sufficiently large size and then recursively solve a 'border' version of the same problem on one of the two sides. The subroutines that we use to compute the separators are those of Chitnis et al.[7] who built upon the work of Kawarabayashi and Thorup [15]. The border version of the problem is a generalization which also incorporates a special bounded set of vertices, called terminals. During the course of our algorithm, we will attempt to solve the border problem on various subgraphs of the input graph. The objective in the border problem is to find a bounded set of vertices contained within a particular subgraph such that any vertex in this subgraph *not* in the computed set is not required in *any* solution for the given instance irrespective of the vertices chosen outside this subgraph.

Given the output of the border problem, the standard approach is to either delete the remaining vertices or simply 'bypass' these vertices. In our case, no such simple reduction seems likely. However, we show that by blowing up the size of the computed set by a function of the parameter, we can use a 'parity preserving' bypassing operation to get rid of the remaining vertices.

This leaves us with the base case of the recursion, that is when we are unable to find a small separator. At this stage, we know that the graph has a bounded sized odd cycle transversal *and* has a highly connected structure. Interestingly, even this seemingly significant structural information regarding the input does not seem enough to imply a straightforward algorithm. Instead, we compute an oct solution and design a branching rule that has as its base case, the case when the oct is not separated by the solution. Here, we rephrase this problem as a Mixed Multiway Cut-Uncut (MMCU) problem and invoke the result in [25] to solve it. In the MMCU problem, the input is a multigraph G, integers k and ℓ, a set of terminals $T \subseteq V(G)$, and equivalence relation $\mathcal{R}$ on the set T. The objective is to output a solution $\mathcal{X} = (X, F)$ such that $X \subseteq (V(G) \setminus T)$, $F \subseteq E(G)$, $|X| \leq k$, $|F| \leq \ell$ and for all $u, v \in T$, u and v belong to the same connected component of $G - \mathcal{X}$ if and only if $(u, v) \in \mathcal{R}$ and $\perp$ if no such solution exists. The structure of our presentation follows that of Chitnis et al.[7] for the most part, except for the high connectivity phase.

Due to space constraints, many proofs of results are omitted or shortened, and will appear in the full version of the paper.

2 Preliminaries

Notations and Definitions: For a graph G, we denote the set of vertices of the graph by $V(G)$ and the set of edges of the graph by $E(G)$. We denote $|V(G)|$ and $|E(G)|$ by n and m respectively, where the graph is clear from context. For a set $S \subseteq V(G)$, the *subgraph of G induced by S* is denoted by $G[S]$ and it is defined as the subgraph of G with vertex set S and edge set $\{(u, v) \in E(G) : u, v \in S\}$ and the subgraph obtained after deleting S is denoted as $G - S$. For $F \subseteq E(G)$, by $V(F)$ we denote the set $\{v \mid \exists u$ such that $(u, v) \in F\}$. For a tuple $\mathcal{X} = (X, F)$ such that $X \subseteq V(G)$ and $F \subseteq E(G)$, by $G - \mathcal{X}$ we denote the graph $G' = (V(G) \setminus X, E(G - X) \setminus F)$. All vertices adjacent to a vertex v are called neighbours of v and the set of all such vertices is called *open* neighbourhood of v, denoted by $N_G(v)$. For a set of vertices $S \subseteq V(G)$, we define $N_G(S) = (\cup_{v \in S} N(v)) \setminus S$. We drop the subscript G when the graph is clear from the context.

An *oct* of a graph G, is a set $X \subseteq V(G)$ such that $G - X$ is bipartite. Similarly, an *edge-oct* of a graph G is set $F \subseteq E(G)$ such that the graph $G - F$ is bipartite. We call a

graph ℓ-*pseudobipartite*, if each of the connected components of G has an edge-oct of size at most ℓ. An ℓ-*pseudobipartite deletion set* of a graph G is a set $X \subseteq V(G)$ such that $G - X$ is ℓ-pseudobipartite. It is easy to see that ℓ-pseudobipartite graphs can be recognized in time $\mathcal{O}(1.977^k n^{\mathcal{O}(1)})$ using the algorithm for Edge Bipartization given in [24]. Now we state the definitions of good node separations and flower separations from [7]. Then we state the lemmas that talk about the running time to find such separations and the properties of the graph if such separations do not exist.

▶ **Definition 2** (C.1 in [7]). Let G be a connected graph and $V^\infty \subseteq V(G)$ be a set of undeletable vertices. A triple (Z, V_1, V_2) of subsets of $V(G)$ is called a (q,k)-good node separation, if $|Z| \leq k$, $Z \cap V^\infty = \emptyset$, V_1 and V_2 are vertex sets of two different connected components of $G - Z$ and $|V_1 \setminus V^\infty|, |V_2 \setminus V^\infty| > q$.

▶ **Definition 3** (C.2 in [7]). Let G be a connected graph, $V^\infty \subseteq V(G)$ be a set of undeletable vertices, and $T_b \subseteq V(G)$ a set of border terminals in G. A pair $(Z, (V_i)_{i=1}^r)$ is called a (q,k)-*flower separation* in G (with regard to border terminals T_b), if the following holds:

- $1 \leq |Z| \leq k$ and $Z \cap V^\infty = \emptyset$; the set Z is the *core* of the flower separation $(Z, (V_i)_{i=1}^r)$;
- V_i are vertex sets of pairwise different connected components of $G - Z$, each set V_i is a *petal* of the flower separation $(Z, (V_i)_{i=1}^r)$;
- $V(G) \setminus (Z \cup \bigcup_{i=1}^r V_i)$, called a *stalk*, contains more than q vertices of $V(G) \setminus V^\infty$;
- for each petal V_i we have $V_i \cap T_b = \emptyset$, $|V_i \setminus V^\infty| \leq q$ and $N_G(V_i) = Z$;
- $|(\bigcup_{i=1}^r V_i) \setminus V^\infty| > q$.

▶ **Lemma 4** (C.3 in [7]). *Given a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and integers q and k, one may find in $\mathcal{O}(2^{\mathcal{O}(\min(q,k)\log(q+k))} n^3 \log n)$ time a (q,k)-good node separation of G, or correctly conclude that no such separation exists.*

▶ **Lemma 5** (C.4 in [7]). *Given a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and border terminals $T_b \subseteq V(G)$ and integers q and k, one may find a (q,k)-flower separation in G w.r.t. T_b in $\mathcal{O}(2^{\mathcal{O}(\min(q,k)\log(q+k))} n^3 \log n)$ time, or correctly conclude that no such flower separation exists.*

▶ **Lemma 6** (C.5 in [7]). *If a connected graph G with undeletable vertices $V^\infty \subseteq V(G)$ and border terminals $T_b \subseteq V(G)$ does not contain a (q,k)-good node separation or a (q,k)-flower separation w.r.t. T_b then, for any $Z \subseteq V(G) \setminus V^\infty$ of size at most k, the graph $G - Z$ contains at most $(2q+2)(2^k - 1) + |T_b| + 1$ connected components containing a vertex of $V(G) \setminus V^\infty$, out of which at most one has more than q vertices not in V^∞.*

3 Overview of the algorithm

To solve Strong Bipartite Deletion, we first reduce it to a slightly more general problem where we also have a set U of undeletable vertices and we are not allowed to select vertices in our potential solution from U. In particular the problem we will study is as follows.

Pseudobipartite Deletion	**Parameters:** k, ℓ
Input: A graph G, integers k and ℓ and a set $U \subseteq V(G)$.	
Question: Does there exist $X \subseteq V(G)$ such that $\lvert X\rvert \leq k$, $X \cap U = \emptyset$ and $G - X$ is ℓ-pseudobipartite?	

Observe that when we set $U = \emptyset$ in Pseudobipartite Deletion, we get the Strong Bipartite Deletion problem. In rest of the paper, we design an algorithm for Pseudobipartite Deletion using the method of iterative compression and recursive understanding

technique introduced by Grohe et al. [12]. We start off by deleting the connected components from the graph which are already ℓ-pseudobipartite. Then using the technique of iterative compression, in Lemma 8, we reduce an instance of Pseudobipartite Deletion to $2^{\mathcal{O}(k)}$ many instances of the same problem such that the original instance is a Yes instance if and only if at least one of the new instances is a Yes instance. In addition to this, we also show that all the new instances have an oct of size bounded by a polynomial in k and ℓ. This will help later in the *high connectivity* phase of the algorithm by giving some structure to the problem, as we will know that the graph has a reasonably small oct. Then we take care of the case when the graph has more than one connected component. This can be easily done by solving the problem optimally on each connected component. Then we define the border problem, where we are additionally provided with some border terminals, say T.

Bordered-Pseudobipartite Deletion (B-PBD) **Parameters:** k,ℓ
Input: A Pseudobipartite Deletion instance $\mathcal{I} = (G, k, \ell, U)$ with G being connected and a set $T \subseteq V(G)$ such that $|T| \leq 2k$; denoted by $\mathcal{I}_b = (G, k, \ell, U, T)$.
Output: Output one of the following three. (a) Find a *special* vertex v such that for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, there is a minimum solution $\mathsf{sol}^*_{\mathcal{P}}$ which contains v, or (b) for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, output $\mathsf{sol}_{\mathcal{P}} = X_{\mathcal{P}}$ which is a *minimum* solution to $(\mathcal{I}_b, \mathcal{P})$, or (c) output $\mathsf{sol}_{\mathcal{P}} = \perp$ if none of the earlier two cases apply.

Here, the set $\mathbb{P}(\mathcal{I}_b)$ denotes the set of 'interactions' of the border terminals of instance $\mathcal{I}_b$ with a solution and the objective is to find a solution that corresponds to each possible interaction or to find a *special* vertex which is part of a solution for each possible interaction. It can be easily seen that for a Pseudobipartite Deletion instance $\mathcal{I} = (G, k, \ell, U)$, solving the B-PBD instance $(G, k, \ell, U, \emptyset)$ either gives a solution to the instance $\mathcal{I}$ or outputs a vertex which is part of a minimum solution for the instance $\mathcal{I}$, so in any case we make progress.

As is the case in algorithms based on the recursive understanding approach, to solve the border problem, we proceed to check whether a good separator T' exists in the graph. We use the notions of good node separations and good flower separations defined by Chitnis et al.[7] and look for good (q, k) node separation or (q, k) flower separation. The definitions are given in the preliminaries section. The running times required to compute such separations (if they exist) are argued by Lemmas 4 and 5. If we succeed in finding such a separation, then the graph gets divided into two large parts using a small separator. The definitions of node separations and flower separation help us argue that one of the parts (containing at most half of the border terminals) is connected. We call the smaller graph H.

Now we update the set of terminals to include the separator, and solve the border problem $\mathcal{I}'_b$ recursively on the smaller graph. That is, for every behavior $\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)$ of the new border terminals, we get an optimum solution. Let Z denote the set $\bigcup_{\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)} \mathsf{sol}_{\mathcal{P}}$ where $\mathsf{sol}_{\mathcal{P}}$ is the solution for the smaller graph for behavior $\mathcal{P}$ of the border terminals. Then in Lemma 9, we argue that there exists a solution for the instance $\mathcal{I}_b$, which intersects with the smaller graph only in Z. At this time, we apply certain operations on the graph, such that the total number of the vertices in the graph G reduces by a sufficient amount. The approach of simply bypassing all the vertices not required by a solution does not quite work, as it could conceivably create spurious odd cycles which could lead to a Yes-instance turning into a No-instance. Hence, we make use of a 'parity preserving' bypassing operation to reduce the vertices of the graph, and show in Lemma 11 that this operation results in an equivalent instance. By proving a bound on the size of the set $\mathbb{P}(\mathcal{I}_b)$ as a function of k and ℓ, we guarantee that after the application of the recursive step (Step 2 in the algorithm) the number of vertices in the graph undergoes a sufficient reduction.

We then describe the algorithm for the problem in the case when there is no good-separation to recurse upon. Here, we need to solve the Border-Pseudobipartite Deletion instance (G, k, ℓ, U, T) in the case that Step 2 is not applicable on the graph. For this, for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)$, we solve the instance $(G_{\mathcal{P}}, k, \ell, U')$, where the graph $G_{\mathcal{P}}$ encoded the interaction, $\mathcal{P}$, of the border terminals T. We argue that in the absence of a good node separation or a flower separation of size at most k in the graph G, Lemma 6 guarantees an appropriate type of *high connectivity* in the graph G. In Lemma 12, we exploit this to get a similar high connectivity property for the graph $G_{\mathcal{P}}$. Since the graph G has an oct of bounded size because of Lemma 8, we can show a similar bound on oct of the graph $G_{\mathcal{P}}$ also. At this stage, we use the algorithm for finding oct given in [19] to find an oct of bounded size and get an instance of the following problem.

OCT-PBD **Parameters:** k,ℓ
Input: An instance (G, k, ℓ, U) of Pseudobipartite Deletion along with an oct O of G of size at most $g(k, \ell)$ such that for any $Z \subseteq (V(G) \setminus U)$ of size at most k, in the graph $G - Z$, at most one connected component containing a vertex of $V(G) \setminus U$ has more than $h(k, \ell)$ vertices not in U.
Output: A minimum sized ℓ-pseudobipartite deletion set X of G of size at most k such that $X \cap U = \emptyset$. Output $\perp$ if such a set does not exist.

Then we guess the intersection of the oct with the solution. This lets us assume that the solution is disjoint from the oct. Formally, we branch into $2^{|O|}$ instances of the following problem which we call OCT-PBD(I). Here, the input is an instance (G, k, ℓ, U, O) of OCT-PBD and the objective is to compute a minimum sized ℓ-pseudobipartite deletion set X of G of size at most k such that $X \cap (O \cup U) = \emptyset$ or return $\perp$ if such a set does not exist. To solve an instance of OCT-PBD(I), we guess which vertices of the oct are going to be in the same connected component after deleting the solution. This gives us $g(k, \ell)^{g(k,\ell)}$ instances of following variant of OCT-PBD(I), which we call OCT-PBD(II). Here, the input is an instance (G, k, ℓ, U, O) of OCT-PBD(I) and an equivalence relation $\S$ on O with the guarantee that there exists a minimum sized ℓ-pseudobipartite deletion set $X \subseteq V(G) \setminus (U \cup O)$ of G such that for all $u, v \in O$, u and v belong to the same connected component of $G - X$ if and only if $(u, v) \in \S$. The objective is to compute a minimum sized ℓ-pseudobipartite deletion set X of G of size at most k such that $X \cap (O \cup U) = \emptyset$. We output $\perp$ if such a set does not exist.

To solve an instance $(G, k, \ell, U, O, \S)$ of OCT-PBD(II), we look at the number of equivalence classes in $\S$. If it is more than one, then either we solve the problem via brute force on some connected component (of size at most $h(k, \ell)$) or we give a branching step where we solve at most $h(k, \ell)+1$ instances OCT-PBD(II), where the size of the solution decreases by at least one. In the other case, there is only one equivalence class in S, which means that all the vertices of oct are going to be part of a single connected component resulting by deleting a solution, we also guess how the oct vertices themselves will be bipartitioned eventually upon deleting the k vertices in the solution and an arbitrary but fixed set of at most ℓ edges which make the connected component containing these vertices bipartite. So for each instance of OCT-PBD(II), we get $2^{|O|}$ instances of the problem OCT-PBD(III) as defined below.

OCT-PBD(III) **Parameters:** k,ℓ

Input: An instance $(G, k, \ell, U, O, \S)$ of OCT-PBD(II) such that for all $u, v \in O$, $(u, v) \in \S$ and a bipartition $(O_1 \uplus O_2)$ of O with the guarantee that there exists a minimum sized ℓ-pseudobipartite deletion set $X' \subseteq V(G) \setminus (U \cup O)$ of G such that all vertices of O belong to the same connected component of $G - X'$ having vertex set C and there exists an edge-oct F of $G[C]$ of size at most ℓ and a bipartition $(C_1 \uplus C_2)$ of C such that $G[C_1] - F$ and $G[C_2] - F$ are independent sets and $O_1 \subseteq C_1$ and $O_2 \subseteq C_2$.

Output: A minimum sized ℓ-pseudobipartite deletion set X of G of size at most k such that $X \cap (O \cup U) = \emptyset$. Output $\perp$ if such a set does not exist.

In Lemma 13, we argue that this branching is correct. Finally, to solve an instance $\mathcal{I}$ of OCT-PBD(III), we reduce it to an instance of MMCU* – a special instance of MMCU with some undeletable vertices.

MMCU* **Parameters:** k,ℓ

Input: A graph G, integers k and ℓ, $T \subseteq V(G)$, an equivalence relation $\mathcal{R}$ on T having at most two equivalence classes, and set of undeletable vertices $U \subseteq V(G)$.

Output: Output a minimal solution $\mathcal{X} = (X, F)$ to MMCU instance $(G, T, \mathcal{R}, k, \ell)$ such that $X \cap U = \emptyset$ and $\perp$ if no such solution exists.

Finally, to solve MMCU* we show that it can be reduced to $(|U|+2)^{|U|}$ instances of MMCU, which is solved using the algorithm in [25].

4 An algorithm for Pseudobipartite Deletion

In this section, we describe the FPT algorithm for Pseudobipartite Deletion. In the first step of the algorithm, we get rid of connected components of the graph which are already ℓ-pseudobipartite.

▶ Step 1. Let (G, k, ℓ, U) be an instance of Pseudobipartite Deletion. Let $\mathcal{C} := \{C \mid G[C] \text{ is a connected component of } G\}$. For each $C \in \mathcal{C}$, find if the graph $G[C]$ is ℓ-pseudobipartite. Let $\mathcal{C}' = \{C \mid C \in \mathcal{C} \text{ and } G[C] \text{ is } \ell\text{-pseudobipartite}\}$ and let $C' = \bigcup_{C \in \mathcal{C}'} C$. Pass the Pseudobipartite Deletion instance $(G - C', k, \ell, U \setminus C')$ to the next step.

Now we use the method of iterative compression to reduce an instance of Pseudobipartite Deletion to at most $n - k$ instances of PBD Compression (PBD-C) problem, where we are given an instance (G, k, ℓ, U) of Pseudobipartite Deletion along with an ℓ-pseudobipartite deletion set of G of size at most $k + 1$ and we are asked whether one exists of size at most k. We prove the following lemma which helps us in bounding the size of an oct.

▶ **Lemma 7.** *Let (G, k, ℓ, U, S) be an instance of* PBD-C *such that none of the connected components of G are ℓ-pseudobipartite. If G has an ℓ-pseudobipartite deletion set Z disjoint from S of size at most k, then it has an oct of size at most $g'(k, \ell) := 2k + k\ell^2 + 1$.*

The proof of the lemma follows from the argument that $G - Z$ can have at most $k + k\ell + 1$ connected components which are not bipartite. Now we prove the lemma which reduces a compression instance to many instances of Pseudobipartite Deletion with a guarantee on the size of an oct for all of them.

▶ **Lemma 8.** *There is an algorithm, which given an instance (G, k, ℓ, U, S) of* PBD-C, *runs in time $2^{\mathcal{O}(k\ell^2)}n^{\mathcal{O}(1)}$ and returns at most 2^{k+1} instances $\{I_1, I_2, \ldots, I_q\}$ of* Pseudobipartite Deletion, *where $I_i = (G_i, k_i, \ell, U)$ and $q \leq 2^{k+1}$ such that the following holds.*

- (G, k, ℓ, U, S) *is a* YES *instance of* PBD-C *if and only if there is an* $1 \leq i \leq q$ *such that* (G_i, k_i, ℓ, U) *is a* YES *instance of* PSEUDOBIPARTITE DELETION.
- *For each* $1 \leq i \leq q$, $k_i \leq k$ *and* G_i *has an oct of size at most* $2k + k\ell^2 + 1$.

Henceforth, we will assume that the given PSEUDOBIPARTITE DELETION instance is returned by the algorithm of Lemma 8 and hence has an oct of bounded size.

Borders and Recursive Understanding. After applying Lemma 8, we still need to solve 2^{k+1} PSEUDOBIPARTITE DELETION instances. We give an algorithm to solve instances which are given by Lemma 8. We first do some preprocessing to assume that the instances of PSEUDOBIPARTITE DELETION which we are going to solve are connected. Now we are ready to define the border problem and describe the recursive phase of the algorithm.

Let $\mathcal{I} = (G, k, \ell, U)$ be a PSEUDOBIPARTITE DELETION instance. The input to the border problem consists of an instance $\mathcal{I} = (G, k, \ell, U)$ of PSEUDOBIPARTITE DELETION along with a set T of at most $2k$ vertices of G disjoint from U. The output to the border problem either consists of *several* solutions, one for each relevant 'behaviour' defined on the set of terminals, or it consists of a single *special* vertex, which is part of some minimum solution for every behaviour. In what follows, we will formalize this statement.

Let $\mathcal{I} = (G, k, \ell, U)$ be an instance of PSEUDOBIPARTITE DELETION and $T \subseteq V(G)$. We let $\mathbb{P}(\mathcal{I}_b)$ denote the set of all tuples $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L)$, such that $X_T \subseteq T$, $\mathcal{R}$ is an equivalence relation on $T \setminus X_T$, $\mathcal{B}$ is a bipartition of $T \setminus X_T$ and $L = \{(R_1, \ell_1), (R_2, \ell_2,) \dots (R_q, \ell_q)\}$ is a set of pairs which associates an integer $\ell_i \leq \ell$ with each equivalence class R_i in $\mathcal{R}$. For a tuple $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L)$ with the equivalence classes of $\mathcal{R}$ being $R_1, \dots, R_q$, the bipartition induced on the class R_i by $\mathcal{B}$ is denoted by (R_{i_1}, R_{i_2}).

For each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, we define a super-graph $G_{\mathcal{P}}$ of G with the following additional vertices and edges.

- For each equivalence class R_i of $\mathcal{R}$, we add sets of vertices R'_{i_1} and R'_{i_2} such that $|R'_{i_1}| = |R'_{i_2}| = \ell + 1$. Then we add all the edges between vertices u and v such that $u \in R_{i_1} \cup R'_{i_1}$ and $v \in R_{i_2} \cup R'_{i_2}$. If $(R_i, \ell_i) \in L$, then we pick an arbitrary vertex $u^i \in R_i$ and add ℓ_i-many edge disjoint triangles which only intersect in u^i. That is, we add $2\ell_i$ new vertices $u^{i,a}_1, \dots, u^{i,a}_{\ell_i}, u^{i,b}_1, \dots, u^{i,b}_{\ell_i}$ and edges $\{(u^i, u^{i,a}_j), (u^i, u^{i,b}_j), (u^{i,a}_j, u^{i,b}_j)\}$ for each $1 \leq j \leq \ell_i$.
- For each vertex $u \in X_T$, we add $\ell + 1$-many edge disjoint triangles which only intersect in u. That is, we add $2(\ell + 1)$ new vertices $u^a_1, \dots, u^a_r, u^b_1, \dots, u^b_r$ where $r = 2\ell + 2$ and edges $\{(u, u^a_j), (u, u^b_j), (u^a_j, u^b_j)\}$ for each $1 \leq j \leq r$.

This completes the description of the graph $G_{\mathcal{P}}$. It can be seen that $|V(G_{\mathcal{P}}) \setminus V(G)| \leq 2k(4\ell + 1)$ and $|E(G_{\mathcal{P}}) \setminus E(G)| \leq 4k(2k + 3)(\ell + 1)$. The intuition behind the newly added vertices and edges is the following. Consider an instance of PSEUDOBIPARTITE DELETION and suppose that G is a subgraph of the input instance with the terminal set T separating this subgraph from the rest of the input graph. The tuple $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L)$ essentially captures the interaction of the terminal set with the solution and the rest of the graph. That is, X_T denotes the intersection of the solution with T. The partition $\mathcal{R}$ denotes the way the remaining vertices of T are partitioned as connected components after deleting a solution. The bipartition $\mathcal{B}$ denotes the bipartition of T induced by an arbitrary bipartition of the graph obtained from the input graph by deleting the k vertices in the solution and then a minimum edge-oct in the rest of the graph. Finally, for each $R \in \mathcal{R}$, if $(R, x) \in L$, it means that after deleting the k vertices in a solution, there is a set of at most x edges in the connected component containing R such that they are part of a minimum edge-oct of

this component *and* they lie outside G. The newly added vertices essentially simulate this interaction of the terminals with the vertices in the solution.

We denote by $U_{\mathcal{P}}$ the union of the set U with the vertices in $V(G_{\mathcal{P}}) \setminus V(G)$. Also, for $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L)$, for each $R_i \in \mathcal{R}$, we denote by $\mathfrak{H}^G_{\mathcal{P}}(R_i)$ the set of vertices in $V(G_{\mathcal{P}}) \setminus V(G)$ which are adjacent to a vertex of R_i. We drop the reference to G if it is clear from the context.

We say that a set $X \subseteq V(G) \setminus U$ is a *solution* to $(\mathcal{I}_b, \mathcal{P})$ where $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L) \in \mathbb{P}(\mathcal{I}_b)$ if X is a solution to the Pseudobipartite Deletion instance $(G_{\mathcal{P}}, k, \ell, U'_{\mathcal{P}})$ where $U'_{\mathcal{P}} = U_{\mathcal{P}} \cup (T \setminus X_T)$. Recall that when we say that a set S is a solution to a Pseudobipartite Deletion instance (G, k, ℓ, U), we mean that $|S| \leq k$, $S \cap U = \emptyset$ and $G - S$ is ℓ-pseudobipartite.

It can be easily seen that for a Pseudobipartite Deletion instance $\mathcal{I} = (G, k, \ell.U)$, solving the B-PBD instance $(G, k, \ell.U, \emptyset)$ either gives a solution to the instance $\mathcal{I}$ or outputs a vertex which is part of a minimum solution for the instance $\mathcal{I}$, so in any case we make progress. In what follows we will show that *given* a correct output for an instance of B-PBD, there is an FPT algorithm which either computes an equivalent instance whose size is bounded by a function of k and ℓ or it outputs a special vertex as described in the problem definition.

▶ **Lemma 9.** *Let $\mathcal{I}_b = (G, k, \ell, U, T)$ be an instance of* B-PBD. *Let $T' \subseteq V(G) \setminus U$ and let C be the vertex set of a connected component of $G - T'$ such that $N(C) = T'$. Let $H = G[C \cup T']$ and let $Q = T' \cup (C \cap T)$ and suppose that $|Q| \leq 2k$. Let $\mathcal{I}'_b$ denote the* B-PBD *instance $(H, k, \ell, (U \cap V(H)), Q)$. Let Z denote the set $\bigcup_{\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)} \mathsf{sol}_{\mathcal{P}}$. Then, for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, if there is a solution for $(\mathcal{I}_b, \mathcal{P})$, then there is one whose intersection with C is in Z. Moreover, if there exists a vertex v such that $v \in \mathsf{sol}_{\mathcal{P}'}$ for all $\mathcal{P}' \in \mathbb{P}(\mathcal{I}'_b)$, then for all $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, there exists a minimum solution to $(\mathcal{I}_b, \mathcal{P})$ which contains v.*

Before we state our next lemma, we need to recall the notion of *parity-torso* (see for example [20]).

▶ **Definition 10.** Let G be a graph and $S \subseteq V(G)$. We denote by $PT(G, S)$ the graph obtained from $G - S$ as follows. For every pair of vertices u, v in $V(G) \setminus S$, if there is an odd length u-v path in G whose internal vertices all lie in S then we add an edge (u, v) and if there is an even length u-v path in G whose internal vertices all lie in S then we add a *subdivided* edge (path of length 2) between u and v.

We are now ready to state our next crucial lemma which essentially gives a way to *get rid of* vertices which we know will never be required in our solution.

▶ **Lemma 11.** *Let $\mathcal{I}_b = (G, k, \ell, U, T)$ be an instance of* B-PBD. *Let $T' \subseteq V(G) \setminus U$ and let C be the vertex set of a connected component of $G - T'$ such that $N(C) = T'$. Let $H = G[C \cup T']$ and let $Q = T' \cup (C \cap T)$ and suppose that $|Q| \leq 2k$. Let $\mathcal{I}'_b$ denote the* B-PBD *instance $(H, k, \ell, (U \cap V(H), Q)$. Suppose that for every $v \in V(H)$, there is a $\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)$ such that $v \notin \mathsf{sol}_{\mathcal{P}}$. Let Z denote the set $\bigcup_{\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)} \mathsf{sol}_{\mathcal{P}}$. Then,*

- *$|Z| = 2^{\mathcal{O}(k \log(k+\ell))}$ and*
- *there are functions τ, α and an algorithm that, given $\mathcal{I}_b$ and Z, runs in time $(\tau(k, \ell) n^{\mathcal{O}(1)})$ and computes a set $W \subseteq V(H)$ such that $W \supseteq Q$ and has size at most $\alpha(k, \ell)$ such that the instance $\mathcal{I}_b$ is equivalent to the instance $\mathcal{I}^1_b = (G', k, \ell, U, T)$ where the graph G' is defined as $PT(G, V(H) \setminus W)$. Here, $\alpha(k, \ell)$ and $\tau(k, \ell)$ are both $2^{\mathcal{O}(k \log(k+\ell))}$.*

Now we describe the recursive step of the algorithm.

▶ Step 2. Assume we are given a B-PBD instance $\mathcal{I}_b = (G, k, \ell, U, T)$ and let $q := \alpha(k, \ell) + \binom{\alpha(k,\ell)}{2} + 1$. Invoke first the algorithm of Lemma 4 in a search for (q, k)-good node separation

(with $V^\infty = U$). If it returns a good node separation (Z, V_1, V_2), let $j \in \{1, 2\}$ be such that $|V_j \cap T| \leq k$ and denote $T' = N(V_j) \subseteq Z$, $C = V_j$. Otherwise, if it returns that no such good node separation exists in G, invoke the algorithm of Lemma 5 in a search for (q, k)-flower separation w.r.t. T_b (with $V^\infty = U$ again). If it returns that no such flower separation exists in G, pass the instance $\mathcal{I}_b$ to the next step (high connectivity phase). Otherwise, if it returns a flower separation $(Z, (V_i)_{i=1}^r)$, denote $C = \bigcup_{i=1}^r V_i$ and $T' = N(C) \subseteq Z$. Let $H = G[C \cup T']$. In the case we have obtained T' and C (either from Lemma 4 or Lemma 5), invoke the algorithm recursively for the B-PBD instance $\mathcal{I}'_b$ defined as in the statement of Lemma 9 for sets T' and set C, obtaining an output $\mathsf{sol}_\mathcal{P}$ for each $\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)$. If there exists $v \in V(H)$ such that $v \in \mathsf{sol}_\mathcal{P}$ for every $\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)$, we return v as the special vertex. Otherwise, compute the set $Z = \bigcup_{\mathcal{P} \in \mathbb{P}(\mathcal{I}'_b)} \mathsf{sol}_\mathcal{P}$. Use the algorithm of Lemma 11 on $\mathcal{I}'_b$ and Z to compute the set W. Generate the graph $G' = PT(G, V(H) \setminus W)$. Let $\mathcal{I}^1_b = (G', k, \ell, (U \cap V(H)), \mathcal{Q})$, where $\mathcal{Q} = T' \cup (C \cap T)$.

Restart this step on instance $\mathcal{I}^1_b$. If it returns a special vertex v, then return v as a special vertex for the instance $\mathcal{I}_b$. Otherwise, obtain a family of solutions $(\mathsf{sol}'_\mathcal{P})_{\mathcal{P} \in \mathbb{P}}$ and return this family as output to the instance $\mathcal{I}_b$.

The correctness of Step 2 follows from Lemmas 9 and 11. Now we do a running time analysis for Step 2. Since $q = \mathcal{O}(2^{\mathcal{O}(k \log(k+\ell))})$, finding a good (q, k)-node separation or flower separation takes time $\mathcal{O}(2^{\mathcal{O}(\min(q,k) \log(q+k))} n^3 \log n) = \mathcal{O}(2^{\mathcal{O}(k^2 \log(k+\ell))} n^3 \log n)$. Let $|V(H)| = n'$, and hence by definitions of good node separation and flower separation we have that $q + 1 \leq n' \leq n - q - 1$. The first recursive call is applied to an instance on n' vertices. While taking the torso operation, we have that $|W| = \alpha(k, \ell) = 2^{\mathcal{O}(k \log(k+\ell))}$ and finding the set W takes $\tau(k, \ell) n^{\mathcal{O}(1)} = 2^{\mathcal{O}(k \log(k+\ell))} n^{\mathcal{O}(1)}$ time.

Let $H' = G - V(H)$. We know that T' separates H' from rest of the graph and $T' \subseteq W$. So, for any $u, v \in V(G)$ such that $u \in V(H')$, we do not have any path from u to v having its internal vertices entirely in $V(H) \setminus W$. So if a new vertex z is added because of an even length path from u to v, we have that $u, v \in V(H)$. As $G - (V(H) \setminus W)$ has at most $n - n' + \alpha(k, \ell)$ vertices and none of the vertices in H' contribute to the torso operation, we have that $|V(G')| \leq n - n' + \alpha(k, \ell) + \binom{\alpha(k,\ell)}{2} < |V(G)|$. The base case for the recursive calls is the high connectivity phase, which we will argue takes time $2^{\mathcal{O}((k+\ell)^3 (\log k + \ell))} n^{\mathcal{O}(1)}$.

Solving the resulting recurrence gives $T(n) = 2^{\mathcal{O}((k+\ell)^3 (\log k + \ell))} n^{\mathcal{O}(1)}$ in the worst case, which is the running time for Step 2. We remark that we never actually introduce any new undeletable vertices in the graph in this step.

High Connectivity phase. Assume we have a B-PBD instance $\mathcal{I}_b = (G, k, \ell, U, T)$ where Step 2 is not applicable. Let us fix $\mathcal{P} = (X_T, \mathcal{R}, \mathcal{B}, L) \in \mathbb{P}(\mathcal{I}_b)$ and let $U' := U \cup ((T \setminus X_T) \cup R_{new})$, where $R_{new} = V(G_\mathcal{P}) \setminus V(G)$. We iterate through all possible values of $\mathcal{P}$ and try to find a minimum solution to $(G_\mathcal{P}, k, \ell, U')$. Since $|\mathbb{P}(\mathcal{I}_b)| = 2^{\mathcal{O}(k \log(k+\ell))}$, this results in a factor of $2^{\mathcal{O}(k \log(k+\ell))}$ in the running time. Thus, from now onwards we focus on one such $\mathcal{P}$. Furthermore, here we only solve the instances where $|U'| \leq \mathcal{O}(k\ell)$, which is sufficient for our purpose as we will argue later. We first prove the following lemma.

▶ **Lemma 12.** *Let $\mathcal{I}_b = (G, k, \ell, U, T)$ be an instance of* B-PBD *where Step 2 is not applicable. Let $U' := U \cup ((T \setminus X_T) \cup R_{new})$, where $R_{new} = V(G_\mathcal{P}) \setminus V(G)$. Then for every $\mathcal{P} \in \mathbb{P}(\mathcal{I}_b)$, the graph $G_\mathcal{P}$ satisfies the following.*

- *for any $Z \subseteq (V(G_\mathcal{P}) \setminus U')$ of size at most k, the graph $G_\mathcal{P} - Z$ contains at most $f(k, \ell) = (2q+2)(2^k - 1) + 2k + 1$ connected components containing a vertex of $V(G) \setminus U'$, out of which at most one has more than $h(k, \ell)$ vertices not in U' where $h(k, \ell) := q(4k(2k+3)(\ell+1)+1)$, and*
- *$G_\mathcal{P}$ has an oct of size at most $g(k, \ell) := 2k + k\ell^2 + 1 + |U'|$.*

So, now we can generate an instance of Pseudobipartite Deletion where an oct is also given and we have a bound on the number of vertices from $V(G) \setminus U$ in all connected components after deleting a solution except one. We recall the formal problem definition from section 3.

OCT-PBD **Parameters:** k,ℓ
Input: An instance (G, k, ℓ, U) of Pseudobipartite Deletion along with an oct O of G of size at most $g(k, \ell)$ such that for any $Z \subseteq (V(G) \setminus U)$ of size at most k, in the graph $G - Z$, at most one connected component containing a vertex of $V(G) \setminus U$ has more than $h(k, \ell)$ vertices not in U.
Output: A minimum sized ℓ-pseudobipartite deletion set X of G of size at most k such that $X \cap U = \emptyset$. Output $\perp$ if such a set does not exist.

▶ Step 3. Find an oct O of $G_{\mathcal{P}}$ of size $g(k, \ell)$ using algorithm in [19]. Return an instance (G, k, ℓ, U, O) of OCT-PBD.

The correctness of Step 3 is immediate from Lemma 12. After doing some simple guessing as explained in the overview section we arrive at an instance of OCT-PBD(II). Now to solve OCT-PBD(II). We return No if $k < 0$. We first look at the case when the equivalence relation § has more than one equivalence class. In this case, we know that there exists a solution X after deleting which, at least one of the equivalence classes of § is in a connected component containing at most $h(k, \ell)$ vertices not from U.

We proceed by guessing this equivalence class S_i in §. Then we arbitrarily pick a vertex v in S_i and look at a connected subgraph H of G containing v which has $h(k, \ell) + 1$ vertices not from U. We know that at least one of the vertices in $V(H)$ has to be part of the solution X, because after deleting X, the connected component containing v has at most $h(k, \ell)$ vertices not from U. Then we pick a vertex of $V(H)$ and branch on it. Since each branching call decreases solution size we are looking for by at least one, the depth of the recursion tree is bounded by k.

If such a subgraph H does not exist, we have that the connected component containing v has at most $h(k, \ell)$ vertices not in U, and then we solve the problem on that connected component using brute force, which takes time $h(k, \ell)^k n^{\mathcal{O}(1)}$.

Now we deal with the case when § has only one equivalence class. That is, we know that there exists a solution $X \subseteq V(G) \setminus (U \cup O)$ such that $G - X$ is ℓ-pseudobipartite and for all $(u, v) \in O$, u and v belong to the same connected component of $G - X$. In other words, there exists a solution X of minimum size such that all the vertices of O lie in the same connected component of $G - X$. To solve this problem, we first prove the following.

▶ **Lemma 13.** *Let $(G, k, \ell, U, O, \S)$ be an* OCT-PBD(II) *instance such that for all $u, v \in O$, $(u, v) \in \S$. Then there exists a bipartition $(O_1 \uplus O_2)$ and $X' \subseteq V(G) \setminus (U \cup O)$ such that X' is a minimum sized ℓ-pseudobipartite deletion set of G, all vertices of O belong to the same connected component of $G - X'$ having vertex set C and there exists an edge-oct F of $G[C]$ of size at most ℓ and a bipartition $(C_1 \uplus C_2)$ of C such that $G[C_1] - F$ and $G[C_2] - F = \emptyset$ are independent sets and $O_1 \subseteq C_1$ and $O_2 \subseteq C_2$.*

This lemma helps us reduce an instance of OCT-PBD(II) into $2^{|O|}$ instances of OCT-PBD(III). We can solve an instance of OCT-PBD(III) by appropriately casting it as an instance of MMCU* problem.

A careful analysis shows that the total running time for high connectivity phase is $2^{\mathcal{O}((k+\ell)^3 \log(k+\ell))} n^{\mathcal{O}(1)}$. This finishes the description of the algorithm. Theorem 1 follows from this algorithm and the fact that the set of undeletable vertices is initially empty, and the

only time we actually add undeletable vertices to a graph is while solving the high connectivity phase, in which case, we add at most $2k(4\ell + 1) = \mathcal{O}(k\ell)$ vertices to the undeletable set.

5 Conclusions

We have introduced and studied a stronger version of the classical $\mathcal{F}$-Deletion problem where $\mathcal{F}$ is the class of bipartite graphs and the new objective is to find a set of k vertices and ℓ edges such that upon deletion of these k vertices, each component of the resulting graph is in the class $\mathcal{F} + \ell e$. We believe that a systematic study of this problem for various well-understood classes $\mathcal{F}$, for instance Interval Graphs, Chordal Graphs and so on, will prove to be a fruitful research direction driving the development of new tools and techniques for graph modification problems. We also think that the idea of combining iterative compression with recursive understanding is quite general in its approach and can be useful in getting similar results for other strong editing problems.

References

1 Akanksha Agrawal, Daniel Lokshtanov, Amer E. Mouawad, and Saket Saurabh. Simultaneous Feedback Vertex Set: A Parameterized Perspective. In Nicolas Ollinger and Heribert Vollmer, editors, *33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, volume 47 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:15, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.STACS.2016.7.

2 Leizhen Cai. Parameterized complexity of vertex colouring. *Discrete Applied Mathematics*, 127(3):415–429, 2003.

3 Yixin Cao. Unit interval editing is fixed-parameter tractable. In *Automata, Languages, and Programming – 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 306–317. Springer, 2015.

4 Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1096–1115, 2016.

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

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

7 Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing FPT algorithms for cut problems using randomized contractions. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 460–469, 2012.

8 Michael R. Fellows, Bart M. P. Jansen, and Frances A. Rosamond. Towards fully multivariate algorithmics: Parameter ecology and the deconstruction of computational complexity. *Eur. J. Comb.*, 34(3):541–566, 2013.

9 Fedor V. Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar F-deletion: Approximation, kernelization and optimal FPT algorithms. In *FOCS*, 2012.

10 Toshihiro Fujito. A unified approximation algorithm for node-deletion problems. *Discrete Appl. Math.*, 86:213–231, September 1998. doi:10.1016/S0166-218X(98)00035-3.

11 Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. In *Proceedings of the Twenty-Seventh Annual*

ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016, pages 1670–1681, 2016.

12 Martin Grohe, Ken-ichi Kawarabayashi, Dániel Marx, and Paul Wollan. Finding topological subgraphs is fixed-parameter tractable. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 479–488, 2011.

13 Bart M. P. Jansen. The power of data reduction: Kernels for fundamental graph problems. *Ph.D. Thesis*, 2013.

14 Bart M. P. Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1802–1811, 2014.

15 Ken-ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 160–169, 2011.

16 Eun Jung Kim and O.-joung Kwon. A polynomial kernel for block graph deletion. In *10th International Symposium on Parameterized and Exact Computation, IPEC 2015, September 16-18, 2015, Patras, Greece*, volume 43 of *LIPIcs*, pages 270–281, 2015.

17 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. In *Automata, Languages, and Programming – 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 613–624, 2013.

18 John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is NP-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980. doi:10.1016/0022-0000(80)90060-4.

19 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.

20 Daniel Lokshtanov and M. S. Ramanujan. Parameterized tractability of multiway cut with parity constraints. In *Automata, Languages, and Programming – 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 750–761, 2012. doi:10.1007/978-3-642-31594-7_63.

21 Carsten Lund and Mihalis Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41:960–981, September 1994. doi:10.1145/185675.306789.

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

23 Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. In *Mathematical Foundations of Computer Science 2015 – 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, volume 9235 of *Lecture Notes in Computer Science*, pages 517–528. Springer, 2015.

24 Marcin Pilipczuk, Michal Pilipczuk, and Marcin Wrochna. Edge bipartization faster than 2^k. In *11th International Symposium on Parameterized and Exact Computation, IPEC 2016, August 24-26, 2016, Aarhus, Denmark*, LIPIcs, 2016.

25 Ashutosh Rai, M. S. Ramanujan, and Saket Saurabh. A parameterized algorithm for mixed-cut. In *LATIN 2016: Theoretical Informatics – 12th Latin American Symposium, Ensenada, Mexico, April 11-15, 2016, Proceedings*, pages 672–685, 2016.

26 Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, STOC'78, pages 253–264, New York, NY, USA, 1978. ACM. doi:10.1145/800133.804355.