

# Deletion to Scattered Graph Classes II - Improved FPT Algorithms for Deletion to Pairs of Graph Classes \*

Ashwin Jacob<sup>1</sup>, Diptapriyo Majumdar<sup>2</sup>, and Venkatesh Raman<sup>3</sup>

<sup>1</sup>*Ben-Gurion University of the Negev, Beersheva, Israel* [ashwinj@bgu.ac.il](mailto:ashwinj@bgu.ac.il)

<sup>2</sup>*Indraprastha Institute of Information Technology Delhi, New Delhi, India*  
[diptapriyo@iiitd.ac.in](mailto:diptapriyo@iiitd.ac.in)

<sup>3</sup>*The Institute of Mathematical Sciences, HBNI, Chennai, India* [vraman@imsc.res.in](mailto:vraman@imsc.res.in)

## Abstract

The problem of deletion of vertices to a hereditary graph class is a well-studied problem in parameterized complexity. Recently, a natural extension of the problem was initiated where we are given a finite set of hereditary graph classes and we determine whether  $k$  vertices can be deleted from a given graph so that the connected components of the resulting graph belong to one of the given hereditary graph classes. The problem is shown to be fixed parameter tractable (FPT) when the deletion problem to each of the given hereditary graph classes is fixed-parameter tractable, and the property of being in any of the graph classes is expressible in the counting monodic second order (CMSO) logic. This paper focuses on pairs of specific graph classes  $(\Pi_1, \Pi_2)$  in which we would like the connected components of the resulting graph to belong to, and design simpler and more efficient FPT algorithms.

## 1 Introduction

Graph modification problems are a class of problems in which the input instance is a graph, and the goal is to check if the input can be transformed into a graph of a specified graph class by using some “allowed” graph operations. Depending on the allowed operations, *vertex or edge deletion problems*, *edge editing* or *contraction problems* have been extensively studied in various algorithmic paradigms.

In the last two decades, graph modification problems, specifically vertex deletion problems, have been extensively studied in the field of parameterized complexity. Examples of vertex deletion problems include VERTEX COVER, CLUSTER VERTEX DELETION, FEEDBACK VERTEX SET, and CHORDAL DELETION SET. We know from the classical result by Lewis and Yannakakis [25] that the problem of whether removing a set of at most  $k$  vertices results in a graph satisfying a hereditary property  $\pi$  is NP-complete for every non-trivial property  $\pi$ . It is well-known that any hereditary graph

---

\*Preliminary versions of the paper appeared in proceedings of IPEC 2020 [19] and FCT 2021 [20].

class<sup>1</sup> can be described by a forbidden set of graphs, finite or infinite, that contains all minimal forbidden graphs in the class. It is also well-known [7] that if a hereditary graph class has a finite forbidden set, then deletion to the graph class has a simple fixed-parameter tractable (FPT) algorithm using a hitting set based approach.

Recently Jacob et al. [18,19], building on the work of Ganian et al. [16] for constraint satisfaction problems, introduced a natural extension of vertex deletion problems to deletion to scattered graph classes. Here we want to delete vertices from a given graph to put the connected components of the resulting graph to one of a few given graph classes. A scattered graph class  $(\Pi_1, \dots, \Pi_d)$  consists of graphs whose connected components are in one of the graph classes  $\Pi_1, \dots, \Pi_d$ . The vertex deletion problem to this class cannot be solved by a hitting set based approach, even if the forbidden graphs for these classes are finite. In particular, it is possible that the solution could be disjoint from the forbidden subgraphs present in the input instance. It is sufficient if the solution separates a forbidden subgraph from one class from a forbidden subgraph of another class so that the forbidden subgraphs of the  $d$  classes don't belong to the same component.

Jacob et al. [19] proved that the vertex deletion problem for the scattered graph class  $(\Pi_1, \dots, \Pi_d)$  is FPT with running time  $2^{\text{poly}(k)} n^{\mathcal{O}(1)}$  ( $\text{poly}(k)$  denotes a polynomial in  $k$ ) if the forbidden families corresponding to all the graph classes  $\Pi_1, \dots, \Pi_d$  are finite. The technique involves iterative compression and important separator variants. In a later result [18], they showed that when the vertex deletion problem to each of the individual graph classes is FPT and for each graph class, the property that a graph belongs to the graph class is expressible by Counting Monadic Second Order (CMSO) logic. Unfortunately, the running time of the algorithm incurs a gargantuan constant factor (a function of  $k$ ) overhead.

Since the algorithms in [18,19] incur a huge running time and use sophisticated techniques, it is interesting to see whether we can get simpler and faster algorithms for some special cases of the problem. In this paper, we do a deep dive on the vertex deletion problems to a pair of graph classes when at least one of the graph classes has an infinite forbidden family.

**Our Problems, Results, and Techniques:** We look at specific variants of the following problem.

$\Pi_1$  OR  $\Pi_2$  DELETION

**Input:** An undirected graph  $G = (V, E)$ , two hereditary graph classes  $\Pi_1$  and  $\Pi_2$  with  $\mathcal{F}_i$  as the forbidden family for graphs whose each connected component belongs to  $\Pi_i$  for  $i \in \{1, 2\}$ .

**Parameter:**  $k$

**Question:** Is there a set  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is in  $\Pi_1$  or in  $\Pi_2$ ?

We emphasize that two distinct components of  $G - S$  can be in two distinct classes  $\Pi_1$  and  $\Pi_2$ . We do not ask that every component of the resulting graph is in  $\Pi_1$  or that every component of the resulting graph is in  $\Pi_2$ . Also, note that the forbidden families  $\mathcal{F}_i$  are for graphs whose each connected component is a graph belonging to  $\Pi_i$  for  $i \in \{1, 2\}$ . It is not the forbidden family of graphs associated to the graph class  $\Pi_i$ . This distinction does not make a difference for most of the popular graph classes as the union of connected components of such graph classes still belong to the graph class.

<sup>1</sup>A hereditary graph class is a class of graphs that is closed under induced subgraphs

Examples include bipartite graphs, chordal graphs, planar graphs, interval graphs, and forests. However, it is important for classes such as cliques and split graphs. The forbidden family for cliques is the singleton graph  $2K_1$ . But if the graph class is such that each connected component is a clique,  $2K_1$  is present by taking a single vertex from two different components of the graph. The forbidden family in this case can be proven to be the singleton graph  $P_3$ . In our definition of  $\Pi_1$  OR  $\Pi_2$  DELETION, when the graph class  $\Pi_1$  is the class of cliques  $\mathcal{F}_1 = \{P_3\}$  is the forbidden family of graphs where each component is a clique.

We describe a general algorithm for  $\Pi_1$  OR  $\Pi_2$  DELETION under some conditions which cover pairs of several graph classes. While the specific conditions on the pairs of classes to be satisfied by this algorithm are somewhat technical and are explained in Section 4.2 and 4.3.2, we give a high-level description here. We note that we do not put any CMSO logic based conditions; thus the algorithm solves fixed parameter tractability for pairs of graph classes not coming under the result by Jacob et al in [18].

We first make the reasonable assumption that the vertex deletion problems to the graph class  $\Pi_1$  and to  $\Pi_2$  have FPT algorithms. As we want every connected component of the graph after removing the solution vertices to be in  $\Pi_1$  or in  $\Pi_2$ , any pair of forbidden subgraphs  $H_1 \in \mathcal{F}_1$  and  $H_2 \in \mathcal{F}_2$  cannot both be in a connected component of  $G$ . Let us look at such a component  $C$  with  $J_1, J_2 \subseteq V(C)$  such that  $G[J_i]$  is isomorphic to  $H_i$  for  $i \in \{1, 2\}$  and look at a path  $P$  between the sets  $J_1$  and  $J_2$ . Assuming that the graphs in families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are connected graphs, we can conclude that the solution has to hit the set  $J_1 \cup J_2 \cup P$  allowing a branching on such sets.

However, if the path is too large, such a branching does not lead to efficient algorithms. The generalization comes up from our observation that for certain pairs of graph classes, if we focus on a pair of forbidden subgraphs  $H_1 \in \mathcal{F}_1$  and  $H_2 \in \mathcal{F}_2$  that are “closest” to each other, then there is always a solution that does not intersect the shortest path  $P$  between them. This helps us to branch on the vertex sets of these forbidden graphs. However, note that the forbidden graphs may have unbounded sizes. We come up with a notion of *forbidden pair* (Definition 5 in Section 4.1) and show that there are pairs of graph classes that have a finite number of forbidden pairs even if each of them has infinite forbidden sets. For some such pairs, we can bound the branching step to obtain the FPT algorithm.

**Organization of the paper:** In Section 2, we state the notations used in this paper and give the necessary preliminaries on various graph classes and parameterized complexity. We also state some preliminary observations and reduction rules for  $\Pi_1$  OR  $\Pi_2$  DELETION. In Section 3, we give algorithms and kernels for the simplest case of the problem when both of the forbidden families associated with the pair of graph classes are finite, and one of them has a path graph present in it.

In Section 4, we give algorithms  $\Pi_1$  OR  $\Pi_2$  DELETION having a constant sized forbidden pair families. We define the notion of forbidden pair family and associated characterizations in Section 4.1. In Section 4.2, we first give an algorithm for  $\Pi_1$  OR  $\Pi_2$  DELETION having a constant sized forbidden pair families assuming that one of the families has a path graph present in it. Later, we give some examples of pairs of graph classes satisfying these properties.

In Section 4.3, we give algorithms for  $\Pi_1$  OR  $\Pi_2$  DELETION having a constant sized forbidden pair families satisfying some additional conditions. For these problems, the path between the closest forbidden pair need not be finite like the examples in Section

4. We motivate this algorithm with the example of CLAW-FREE-OR-TRIANGLE-FREE DELETION in Section 4.3.1. In Section 4.3.2, we give the general algorithm for this case. In Section 5, we give more examples for this case, such as INTERVAL-OR-TREE DELETION, PROPER INTERVAL-OR-TREE DELETION, and CHORDAL-OR-BIPARTITE PERMUTATION DELETION.

## 2 Preliminaries

**Sets and Graph Theory:** Given  $r \in \mathbb{N}$ , we use  $[r]$  to denote the set  $\{1, \dots, r\}$ . Given a finite set  $A$ , and an integer  $t$ , we use  $\binom{A}{t}$  to denote the collection of all subsets of  $A$  of size exactly  $t$  and use  $\binom{A}{\leq t}$  to denote the collection of all subsets of  $A$  of size at most  $t$ . We consider undirected graphs throughout this paper. We use standard graph-theoretic notations for undirected graphs from [14]. For  $\ell \in \mathbb{N}$ , we use  $P_\ell$  to denote the path on  $\ell$  vertices. Similarly, for  $\ell \in \mathbb{N}$ , we use  $C_\ell$  to denote an induced cycle on  $\ell$  vertices. Let  $u, v \in V(G)$  and  $K_\ell$  to denote a clique with  $\ell$  vertices. We use  $d_G(u, v)$  to denote the length of a *shortest path* from  $u$  to  $v$  in  $G$ . For  $P, Q \subseteq V(G)$ , we define  $d_G(P, Q) = \min_{u \in P, v \in Q} \{d_G(u, v)\}$ .

Given a graph  $G = (V, E)$ , and  $Y \subseteq V(G)$ , we denote by  $G[Y]$  the subgraph of  $G$  induced by the vertex set  $Y$ . A graph  $G$  is called a *bipartite graph* if there exists a partition of  $V(G) = A \uplus B$  such that for every edge  $uv \in E(G)$ ,  $u \in A$  and  $v \in B$ . A graph  $G$  is called a *split graph* if its vertex set can be partitioned into two parts  $V(G) = C \uplus I$  such that  $C$  is a clique and  $I$  is an independent set. A graph is called a *cactus graph* if every edge of the graph is contained in at most one cycle. Let  $A$  be a set of three arbitrary vertices of a graph  $G$ . Then,  $A$  is called an *asteroidal triple (AT)* if between every two vertices of  $A$ , there is a path avoiding the third vertex. A *chord* of a cycle  $C = v_0 v_1 \dots v_p v_0$  is an edge  $(v_i, v_j)$  with  $|i - j| \geq 2$ . A chordless cycle is a cycle having no chord. A *hole* is a chordless cycle of length at least 4. A graph is called a *chordal graph* if it has holes as induced subgraphs. A graph is called an *interval graph* if it is chordal and AT-free. Alternatively, any interval graph has an interval representation. It means that every vertex of an interval graph can be represented as an interval on the real line and two vertices are adjacent if and only if the intervals representing the corresponding vertices intersect. A graph is called a *proper interval graph* if it is an interval graph with an interval representation such that no interval properly contains any other interval. A graph is called a *bipartite permutation graph* if it is bipartite and AT-free.

A *sunflower* with  $k$  *petals* and *core*  $Y$  is a family of sets  $\{S_1, \dots, S_k\}$  such that  $S_i \cap S_j = Y$  for all  $i \neq j$ . The sets  $S_i \setminus Y$  are petals and we require none of them to be empty. We have the following lemma.

**Lemma 1.** *Let  $\mathcal{F}$  be a family of sets over a universe  $U$ , such that each set in  $\mathcal{F}$  has cardinality exactly  $d$ . If  $|\mathcal{F}| > d!(k-1)^d$ , then  $\mathcal{F}$  contains a sunflower with  $k$  petals and such a sunflower can be computed in time polynomial in  $|\mathcal{F}|, |U|$  and  $k$ .*

**Parameterized Complexity:** A parameterized problem  $L$  is a subset of  $\Sigma^* \times \mathbb{N}$  for some finite alphabet  $\Sigma$ . An instance of a parameterized problem  $L$  is a pair  $(x, k) \in \Sigma^* \times \mathbb{N}$  where  $k$  is called the parameter and  $x$  is called the input. We assume that  $k$  is

given in unary and without loss of generality  $k \leq |x|$ , such that  $|x|$  is the length of the input.

**Definition 1** (Fixed-Parameter Tractability). *A parameterized problem  $L \subseteq \Sigma^* \times \mathbb{N}$  is said to be fixed-parameter tractable if there exists an algorithm  $\mathcal{A}$  that given an input  $(x, k)$ , runs in  $f(k)|x|^c$  time and correctly decides if  $(x, k) \in L$  where  $c$  is a fixed constant independent of  $|x|$  and  $k$ .*

A closely related notion to fixed-parameter tractability is the notion of Kernelization defined below.

**Definition 2** (Kernelization). *Let  $L \subseteq \Sigma^* \times \mathbb{N}$  be a parameterized language. Kernelization is a procedure that replaces the input instance  $(I, k)$  by a reduced instance  $(I', k')$  such that*

- $k' \leq f(k)$ ,  $|I'| \leq g(k)$  for some function  $f, g$  depending only on  $k$ .
- $(I, k) \in L$  if and only if  $(I', k') \in L$ .

*The reduction from  $(I, k)$  to  $(I', k')$  must be computable in  $\text{poly}(|I| + k)$  time. If  $g(k) = k^{O(1)}$  then we say that  $L$  admits a polynomial kernel.*

For more details on parameterized complexity, we refer to [12].

We now end the section by introducing some notations and observations on  $\Pi_1$  OR  $\Pi_2$  DELETION that we use in the paper. Throughout this paper, we assume that the graphs in the forbidden families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  associated to  $\Pi_1$  OR  $\Pi_2$  DELETION are connected which is true for most of the well-known graph classes. We use  $\Pi_{(1,2)}$  to denote the class of graphs whose connected components are in the graph classes  $\Pi_1$  or  $\Pi_2$ .

**Definition 3** (Minimal Forbidden Family). *A forbidden family  $\mathcal{F}$  for a graph class  $\Pi$  is said to be minimal if for all graphs  $H \in \mathcal{F}$ , we have that  $\mathcal{F} - \{H\}$  is not a forbidden family for  $\Pi$ .*

Let  $\mathcal{F}_1 \times \mathcal{F}_2 = \{(H_1, H_2) : H_1 \in \mathcal{F}_1 \text{ and } H_2 \in \mathcal{F}_2\}$ . The following characterization for  $\Pi_{(1,2)}$  is easy to see.

**Observation 1.** *A graph  $G$  is in the graph class  $\Pi_{(1,2)}$  if and only if no connected component  $C$  of  $G$  contains  $H_1$  and  $H_2$  as induced graphs in  $C$ , where  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ .*

Let  $J_1, J_2 \subseteq V(G)$  such that  $G[J_i]$  is isomorphic to graphs  $H_i$  for  $i \in \{1, 2\}$  and  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ . We call the sets  $J_1$  and  $J_2$  as the vertex sets of the pair  $(H_1, H_2)$ .

In the following lemma, we show that any solution to  $\Pi_1$  OR  $\Pi_2$  DELETION must either hit the vertex sets of a pair in  $\mathcal{F}_1 \times \mathcal{F}_2$  or a path connecting them.

**Lemma 2.** *Let  $J_1, J_2 \subseteq V(G)$  such that  $G[J_i]$  is isomorphic to graphs  $H_i$  for  $i \in \{1, 2\}$  and  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ . Let  $P$  be a path between  $J_1$  and  $J_2$  in the graph  $G$ . Then any solution for  $\Pi_1$  OR  $\Pi_2$  DELETION with input graph  $G$  contains one of vertices in the set  $J_1 \cup J_2 \cup P$ .*

*Proof.* Suppose this is not the case. Since the graphs  $H_1$  and  $H_2$  are connected, the graph  $G'$  induced by the set  $J_1 \cup J_2 \cup P$  is a connected subgraph of  $G$ . If a solution  $X$  does not intersect  $J_1 \cup J_2 \cup P$ , then  $G'$  occurs in a connected component  $C$  of the remaining graph  $G - X$ . The presence of graphs  $H_1$  and  $H_2$  in  $C$  implies that  $C$  is neither in  $\Pi_1$  nor in  $\Pi_2$  giving a contradiction that  $X$  is a solution.  $\square$

We use the following reduction rule for  $\Pi_1$  OR  $\Pi_2$  DELETION whose correctness easily follows.

**Reduction Rule 1.** *If a connected component  $C$  of  $G$  is in  $\Pi_1$  or in  $\Pi_2$ , then delete  $C$  from  $G$ . The new instance is  $(G - V(C), k)$ .*

### 3 FINITE $\Pi_1$ OR $\Pi_2$ DELETION with forbidden paths

In this section, we restrict the problem to the case where both the forbidden families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are finite and there exists a path  $P_\alpha$  in one of the families, say  $\mathcal{F}_1$  where  $\alpha$  is some constant. Observe that for several natural graph classes (like cluster graphs, edgeless graphs, split cluster graphs, cographs) paths above a certain length is forbidden.

We define the problem below.

FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH

**Input:** An undirected graph  $G$ , and an integer  $k$ . Furthermore, for a fixed integer  $\alpha$ , the path  $P_\alpha \in \mathcal{F}_1$ .

**Question:** Does  $G$  have a set  $S$  of at most  $k$  vertices such that every connected component of  $G - S$  is in  $\Pi_1$  or in  $\Pi_2$ ?

Since both  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are finite, the set  $\mathcal{F}_1 \times \mathcal{F}_2$  is also finite.

Let us look at a pair  $(H_1, H_2)$  such that the distance between its vertex sets is the smallest among all the pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$ . We call such a pair the *closest* pair. We claim below that this distance is bounded by  $\alpha$ .

**Lemma 3.** *Let  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$  be a closest pair in the graph  $G$  and let  $(J_1, J_2)$  be a pair of vertex subsets corresponding to the pair. Let  $P$  be a shortest path between  $J_1$  and  $J_2$ . Then  $|V(P)| \leq \alpha$ .*

*Proof.* Suppose this is not the case. Then let us look at the set  $J'_1$  of the last  $\alpha$  vertices of  $P$  which is isomorphic to  $P_\alpha \in \mathcal{F}_1$ . Then the pair of vertex subsets  $(J'_1, J_2)$  corresponds to the pair  $(P_\alpha, H_2)$  in the graph  $G$  with the distance between them as zero. This contradicts that  $(J_1, J_2)$  is the vertex subsets of the closest pair.  $\square$

Hence, we have the following branching rule for closest pairs where we branch on the vertex subsets plus the vertices of the path. The correctness follows from Lemma 2.

**Branching Rule 1.** *Let  $(J^*, T^*)$  be the vertex subsets of a closest pair  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ . Let  $P^*$  be a path corresponding to this forbidden pair. Then for each  $v \in J^* \cup T^* \cup P^*$ , we delete  $v$  and decrease  $k$  by 1, resulting in the instance  $(G - v, k - 1)$ .*

Using this branching rule, we have an FPT algorithm for FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH. Let  $d_i$  be the size of a maximum sized finite forbidden graph in

$\mathcal{F}_i$  for  $i \in \{1, 2\}$ . Observe that  $|(J^* \cup T^*) \cap P^*| \geq 2$  as  $P^*$  be a shortest path between a vertex of  $J^*$  and a vertex of  $P^*$ . Therefore,  $|J^* \cup T^* \cup P^*| \leq |J^*| + |T^*| + |P^*| - 2$ . As  $|J^*| \leq d_1, |T^*| \leq d_2$  and  $|P^*| \leq \alpha$ , we have that  $|J^* \cup T^* \cup P^*| \leq d_1 + d_2 + \alpha - 2$ . Let  $c = d_1 + d_2 + \alpha - 2$ .

**Theorem 1.** FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH *can be solved in  $c^k \text{poly}(n)$  time.*

*Proof.* We describe our algorithm as follows. Let  $(G, k)$  be an input instance of FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH. We exhaustively apply Reduction Rule 1 and Branching Rule 1 in sequence to get an instance  $(G', k')$ . The algorithm finds the closest pair to apply Branching Rule 1 by going over all pairs in  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$  and going over all subsets of size  $|V(H_1)| + |V(H_2)|$  of the graph (which is still a polynomial in  $n$ ) and checking the distance between them.

Every component of  $G'$  is such that it is  $\mathcal{F}_1$ -free or  $\mathcal{F}_2$ -free or in other words in  $\Pi_1$  or  $\Pi_2$ . Hence if  $k' \geq 0$ , we return yes-instance. Otherwise, we return no-instance. Since the largest sized obstruction in these rules is at most  $c = d_1 + d_2 + \alpha - 2$ , the bounded search tree of the algorithm has  $c^k$  nodes bounding the running time to  $c^k \text{poly}(n)$ . This completes the proof.  $\square$

We now describe a family of graphs such that instead of focusing that each component is free of pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$ , we can check whether the graph is free of graphs in this family.

Let  $\mathcal{F}'$  be the minimal family of graphs (as in Definition 3) such that for each member  $H \in \mathcal{F}'$ , there exist subsets  $J_1, J_2 \subseteq V(H)$  such that  $H[J_i]$  is isomorphic to  $H_i \in \mathcal{F}_i$  for  $i \in \{1, 2\}$  and  $d_H(J_1, J_2) \leq \alpha - 1$ .

From Lemma 2, it can be inferred that any graph without any members from  $\mathcal{F}'$  as induced subgraphs belong to the graph class  $\Pi_{1,2}$ . Hence, the set  $\mathcal{F}'$  is the forbidden family for the graph class  $\Pi_{1,2}$ . We now use this family to give a kernel and approximation algorithm for FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH.

The size of any member  $H \in \mathcal{F}'$  is bounded by  $d = d_1 + d_2 + \alpha - 2$ . Suppose not. Then we can identify a vertex  $v \in V(H)$  which is not part of  $J_1, J_2$  and a path  $P$  of length at most  $\alpha - 1$  between them. But then the graph  $H \setminus \{v\}$  is also in  $\mathcal{F}'$  contradicting that  $\mathcal{F}'$  is minimal. This also proves that the size of  $\mathcal{F}'$  is bounded by  $2^{\binom{d+1}{2}}$  which is the bound on the number of graphs of at most  $d$  vertices.

**Theorem 2.** FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH *admits a  $d$ -approximation algorithm, and a  $\mathcal{O}(k^d)$  sized kernel.*

*Proof.* We show that an instance of FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH can be reduced to an instance of  $d$ -HITTING SET problem defined as follows.

**$d$ -HITTING SET**

**Input:** A family  $\mathcal{S}$  of sets over a universe  $U$ , where each set in  $\mathcal{S}$  has size at most  $d$ , and an integer  $k$ .

**Parameter:**  $k$ .

**Question:** Does there exist a subset  $X \subseteq U, |X| \leq k$  such that  $X$  contains at least one element from each set in  $\mathcal{S}$ ?

An instance  $(G, k)$  of FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH, we construct an instance  $(U, \mathcal{S}, k)$   $d$ -HITTING SET problem with  $U = V(G)$  and  $\mathcal{S}$  being the vertex sets

of all induced graphs of  $G$  that is isomorphic to a graph in  $\mathcal{F}$ . Note that every set in  $\mathcal{S}$  has size at most  $d = d_1 + d_2 + \alpha - 2$ . It is easy to see that  $(G, k)$  is a yes-instance of FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH if and only if  $(U, \mathcal{S}, k)$  is a yes-instance of  $d$ -HITTING SET.

It is folklore that the  $d$ -HITTING SET problem has a  $d$ -approximation algorithm and a kernel with  $\mathcal{O}(k^d)$  size. We adapt the techniques used in this to give a  $d$ -approximation algorithm and a  $\mathcal{O}(k^d)$  sized kernel for FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH.

**Approximation Algorithm.** Let us define the family  $\mathcal{S}$  as follows. Initially  $\mathcal{S} = \emptyset$ . In polynomial time, we find a subset  $T \subseteq V(G)$  such that  $G[T]$  is isomorphic to a member in  $\mathcal{F}$ . We add  $T$  to  $\mathcal{S}$  and update  $G$  to  $G - T$ . We repeat this step until it is no longer applicable.

Let  $S_{\text{OPT}}$  be the minimum sized set such that in the graph  $G - S_{\text{OPT}}$ , every connected component is either in  $\Pi_1$  or  $\Pi_2$ . Let  $|S_{\text{OPT}}| = \text{OPT}$ . Let  $\mathcal{S}$  be the set of vertices that is present in any pair of graphs in  $\mathcal{S}$ . From Lemma 2, we can conclude that any feasible solution of  $G$  must contains a vertex from each member of the family  $\mathcal{S}$ . Since the members of  $\mathcal{S}$  are pairwise disjoint, we have that  $|S_{\text{OPT}}| \geq |\mathcal{S}|$ .

We have  $|\mathcal{S}| \leq \max_{T \in \mathcal{F}} |T| \cdot |\mathcal{S}| \leq d|S_{\text{OPT}}|$ . Thus we have a  $d$ -approximation algorithm for FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH.

**Kernel.** Given an instance  $(G, k)$  of FINITE  $\Pi_1$  OR  $\Pi_2$  DELETION WITH PATH, we construct an equivalent instance  $(U, \mathcal{S}, k)$   $d$ -HITTING SET as described earlier. For each  $d' \in [d]$ , we repeatedly do the following. If  $|\mathcal{S}| > d'!(k+1)^{d'}$ , from Lemma 1, we obtain a sunflower of size  $k+2$  in polynomial time. Let  $Y$  be the core of sunflower and  $S_1 \setminus Y, \dots, S_{k+2} \setminus Y$  be the  $k+2$  petals, which are non-empty. We remove  $S_{k+2}$  from  $\mathcal{S}$ . Let  $T = \bigcup_{S \in \mathcal{S}} S$ . We reduce  $(G, k)$  to  $(G[T], k)$ .

We now claim that the reduction rule is safe. The forward direction is trivial as  $G[T]$  is an induced subgraph of  $G$ . In the reverse direction, suppose  $X$  is a solution of size  $k$  of the instance  $(G[T], k)$ . Suppose there exist a subset  $U \subseteq V(G) \setminus X$  such that  $G[U] \in \mathcal{F}$ . If  $U \subseteq V(G[T])$ , it contradicts that  $X$  is a solution of  $(G[T], k)$ . In the other case,  $U$  part of a sunflower. Let this sunflower be  $S_1, S_2, \dots, S_{k+1}, U$  with core  $Y$ . Since graphs induced by  $S_1, \dots, S_{k+1}$  in  $G[T]$  are in  $\mathcal{F}$ ,  $X$  should intersect each of them. Suppose  $X$  does not intersect  $Y$ . Then  $X$  must contain an element from each of the petals  $S_1 \setminus Y, \dots, S_{k+1} \setminus Y$ . However, since  $S_i \cap S_j = Y$  for all  $i, j \in [k+1], i \neq j$ , we have  $|X| > k$ , a contradiction. Thus  $X$  must contain an element from  $Y$ . But in this case,  $X$  intersects  $U$  as well, giving a contradiction.

When the reduction rule is not applicable, we have  $|\mathcal{S}| \leq d!(k+1)^d$  and thus,  $|T| \leq d \cdot d!(k+1)^d$ . Thus we have a kernel with  $\mathcal{O}(k^d)$  vertices.  $\square$

## 4 $\Pi_1$ OR $\Pi_2$ DELETION with a constant number of forbidden pairs

### 4.1 Forbidden Characterization for $\Pi_1$ OR $\Pi_2$ DELETION

Unfortunately, the algorithm in Section 3 does not work when at least one of the sets  $\mathcal{F}_1$  or  $\mathcal{F}_2$  is infinite as the family  $\mathcal{F}_1 \times \mathcal{F}_2$  is no longer finite. But we observed that for many problems, branching on most of the pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$  could be avoided.

We aim to identify such ‘redundant’ pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$ . Instead of ensuring that such pairs are absent in a graph for an instance of  $\Pi_1$  OR  $\Pi_2$  DELETION, we identify some graphs which are forbidden in such a graph. Since hitting the vertices of forbidden induced graphs is the familiar framework in designing algorithms for vertex deletion problems, such a characterization for  $\Pi_{(1,2)}$  with forbidden graphs and irredundant pairs would be useful.

For example, let  $\mathcal{F}_1 = \{C_3, C_4\}$  and  $\mathcal{F}_2 = \{D_4, C_4\}$  where  $D_4$  is the graph obtained after removing an edge from  $K_4$ . We have  $\mathcal{F}_1 \times \mathcal{F}_2 = \{(C_3, D_4), (C_3, C_4), (C_4, D_4), (C_4, C_4)\}$ . Since  $C_4 \in \mathcal{F}_1 \cap \mathcal{F}_2$ , the graph  $C_4$  is forbidden for the graph class  $\Pi_{1,2}$ . Hence the pairs  $(C_4, C_4)$ ,  $(C_3, C_4)$  and  $(C_4, D_4)$  are redundant by identifying that  $C_4$  is forbidden. Now note that  $C_3$  is an induced subgraph of the graph  $D_4$ . Hence the pair  $(C_3, D_4)$  can also be made redundant by identifying that  $D_4$  is forbidden for  $\Pi_{(1,2)}$ .

We now formalize such forbidden graphs in the graph class  $\Pi_{(1,2)}$  by defining the notion of super-pruned family. Recall that if a family of graphs is *minimal*, no element of it is an induced subgraph of some other element of the family.

**Definition 4** (Super-Pruned Family). *An element of a super-pruned family  $\text{sp}(\mathcal{G}_1, \mathcal{G}_2)$  of two minimal families of graphs  $\mathcal{G}_1$  and  $\mathcal{G}_2$  is a graph that (i) belongs to one of the two families and (ii) has an element of the other family as induced subgraph.*

*The family  $\text{sp}(\mathcal{G}_1, \mathcal{G}_2)$  can be obtained from an enumeration of all pairs in  $\mathcal{G}_1 \times \mathcal{G}_2$  and adding the supergraph if one of the graphs is an induced subgraph of the other. The family obtained is made minimal by removing the elements that are induced subgraphs of some other elements.*

For example, let  $(\Pi_1, \Pi_2)$  be (Interval, Trees), with the forbidden families  $\mathcal{F}_1 = \{\text{net, sun, long-claw, whipping top, } \dagger\text{-AW, } \ddagger\text{-AW}\} \cup \{C_i : i \geq 4\}$  (See Figure 2) and  $\mathcal{F}_2$  as the set of all cycles. Note that all graphs  $C_i$  with  $i \geq 4$  are in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as they occur in both  $\mathcal{F}_1$  and  $\mathcal{F}_2$ . The remaining pairs of  $\mathcal{F}_1 \times \mathcal{F}_2$  contain triangles from  $\mathcal{F}_2$ . If the graph from  $\mathcal{F}_1$  is a net, sun, whipping top,  $\dagger$ -AW or  $\ddagger$ -AW, it contains triangle as an induced subgraph. Hence these graphs are also in the family  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ .

We now show that graphs in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  are forbidden in the graph class  $\Pi_{(1,2)}$ .

**Lemma 4.** *If a graph  $G$  is in the graph class  $\Pi_{(1,2)}$ , then no connected component of  $G$  contains a graph in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as induced subgraphs.*

*Proof.* Suppose a graph  $H \in \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  occur as induced subgraph of a connected component  $C$  of  $G$ . From the definition of Super-Pruned Family, we can associate a pair  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$  to  $H$  such that either  $H$  is isomorphic to  $H_1$  and  $H_2$  is an induced subgraph of  $H_1$  or vice-versa. Without loss of generality, let us assume the former. Since  $H_1 \in \mathcal{F}_1$ , we know that  $C$  is not in the graph class  $\Pi_i$  for  $i \in \{1, 2\}$ . This contradicts that  $G$  is in the graph class  $\Pi_{(1,2)}$ .  $\square$

Hence any pair containing a graph from  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  are redundant. But  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  does not capture all the pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$ . We now define the following family to capture the remaining pairs.

**Definition 5** (Forbidden Pair Family). *A forbidden pair family  $\mathcal{F}_p$ , of  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , consists of all pairs  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$  such that both  $H_1 \notin \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  and  $H_2 \notin \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ .*

Informally, a pair  $(H_1, H_2)$  is part of a forbidden pair family  $\mathcal{F}_p$  if at least one of  $H_1$  and  $H_2$  does not belong to  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . For example, if  $\Pi_1$  is the class of interval graphs and  $\Pi_2$  is the class of forests, we have already shown that  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  contains all the graphs in  $\mathcal{F}_1$  except long-claw. The only remaining pair is (long-claw, triangle) and the singleton set containing this pair forms the forbidden pair family. Now we characterize  $\Pi_{(1,2)}$  based on the super-pruned family and the forbidden pair family associated with  $\mathcal{F}_1$  and  $\mathcal{F}_2$  as follows. This is used in the algorithms in Section 4.

**Lemma 5.** *The following statements are equivalent.*

- *Each connected component of  $G$  is either in  $\Pi_1$  or  $\Pi_2$ .*
- *The graph  $G$  does not contain graphs in the super-pruned family  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as induced subgraphs. Furthermore, for pairs  $(H_1, H_2)$  in the forbidden pair family of  $\mathcal{F}_1$  and  $\mathcal{F}_2$ ,  $H_1$  and  $H_2$  both cannot appear as induced subgraphs in a connected component of  $G$ .*

*Proof.* To prove the forward direction, note that from Lemma 4, the  $G$  does not contain graphs in the super-pruned family  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as induced subgraphs. Hence, suppose that there exists a pair  $(H_1, H_2) \in \mathcal{F}_p$  in a connected component  $\chi$  of  $G$ . But then  $\chi$  cannot be in  $\Pi_1$  due to the presence of  $H_1$  and cannot be in  $\Pi_2$  due to the presence of  $H_2$  giving a contradiction. To prove the converse, suppose that  $G$  contains a component  $\chi$  which is neither in  $\Pi_1$  nor in  $\Pi_2$ . Then there exist graphs  $H_1 \in \mathcal{F}_1$  and  $H_2 \in \mathcal{F}_2$  occurring as induced subgraphs of  $\chi$ . If  $H_1$  occurs as an induced subgraph of  $H_2$  or vice-versa, then the supergraph occurs in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  giving a contradiction. Else we have  $H_1 \in \mathcal{F}_1 \setminus \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  and  $H_2 \in \mathcal{F}_2 \setminus \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . Hence  $(H_1, H_2) \in \mathcal{F}_p$  giving a contradiction.  $\square$

We now define useful notions of forbidden sets and closest forbidden pairs for the graph class  $\Pi_{(1,2)}$ .

**Definition 6.** *We call a minimal vertex subset  $Q \subseteq V(G)$  as a forbidden set corresponding to the graph class  $\Pi_{(1,2)}$  if  $G[Q]$  is isomorphic to a graph in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  or  $G[Q]$  is connected and contains both  $H_1$  and  $H_2$  as induced subgraphs for some forbidden pair  $(H_1, H_2)$  of  $\Pi_{(1,2)}$ .*

**Definition 7.** *We say that a forbidden pair  $(H_1, H_2)$  is a closest forbidden pair in a graph  $G$  if there exists subsets  $J_1, J_2 \subseteq V(G)$  such that  $G[J_1]$  is isomorphic to  $H_1$ ,  $G[J_2]$  is isomorphic to  $H_2$  and the distance between  $J_1$  and  $J_2$  in  $G$  is the smallest among all such pairs over all forbidden pairs of  $\mathcal{F}_1$  and  $\mathcal{F}_2$ . We call the pair of vertex subsets  $(J_1, J_2)$  as the vertex subsets corresponding to the closest forbidden pair. We call a shortest path  $P$  between  $J_1$  and  $J_2$  as the path corresponding to the closest forbidden pair.*

## 4.2 The case with forbidden paths

We now aim to give an algorithm for  $\Pi_1$  OR  $\Pi_2$  DELETION when the forbidden families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  be infinite but the forbidden pair family  $\mathcal{F}_p$  is finite. We also assume that  $P_\alpha \in \mathcal{F}_1$  and  $\mathcal{F}_p$  is given as input.

Let us list the conditions that  $\Pi_1$  OR  $\Pi_2$  DELETION is required to satisfy.

1. The vertex deletion problems for the graph classes  $\Pi_1$  and  $\Pi_2$  are FPT with algorithms to the respective classes being  $\mathcal{A}_1$  and  $\mathcal{A}_2$ .
2.  $\mathcal{F}_p$ , the forbidden pair family of  $\mathcal{F}_1$  and  $\mathcal{F}_2$  is of constant size.
3. The path  $P_\alpha \in \mathcal{F}_1$ .

**$P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION**

**Input:** An undirected graph  $G$ , graph classes  $\Pi_1, \Pi_2$  with associated forbidden families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  such that Conditions 1 - 3 are satisfied and an integer  $k$ .

**Question:** Does  $G$  have a set  $S$  of at most  $k$  vertices such that every connected component of  $G - S$  is either in  $\Pi_1$  or in  $\Pi_2$ ?

Since the forbidden pair set is finite, we have the following Branching Rule for closest forbidden pairs which is similar to Branching Rule 1 where we branch on the vertex subsets plus the vertices of the path. The correctness follows from Lemma 2.

**Branching Rule 2.** *Let  $(J^*, T^*)$  be the vertex subsets of a closest forbidden pair  $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ . Let  $P^*$  be a path corresponding to this forbidden pair. Then for each  $v \in J^* \cup T^* \cup P^*$ , we delete  $v$  and decrease  $k$  by 1, resulting in the instance  $(G - v, k - 1)$ .*

From here on, assume that  $(G, k)$  be an instance at which Reduction Rule 1 and Branching Rule 2 are not applicable. Note that any component of  $G$  is now free of forbidden pairs.

Let  $\mathcal{F}_p^1$  denote the family of graphs  $H_1$  where  $(H_1, H_2) \in \mathcal{F}_p$ . Similarly define  $\mathcal{F}_p^2$  as the family of graphs  $H_2$  where  $(H_1, H_2) \in \mathcal{F}_p$ . By the definition of forbidden pairs, the set of pairs  $(H_1, H_2)$  with  $H_i \in \mathcal{F}_p^i$  is the forbidden pair set  $\mathcal{F}_p$ . Hence a graph that does not contain any forbidden pairs is  $\mathcal{F}_p^1$ -free or  $\mathcal{F}_p^2$ -free. With this observation, the following results are easy to see.

**Lemma 6.** *Let  $C$  be a connected component of  $G$  that is  $\mathcal{F}_p^i$ -free for some  $i \in \{1, 2\}$ . If  $G[C]$  has no  $\Pi_i$  vertex deletion set of size  $k$ , then  $(G, k)$  is a no-instance. Otherwise, let  $X$  be a minimum  $\Pi_i$  vertex deletion set of  $G[C]$ . Then  $(G, k)$  is a yes-instance if and only if  $(G - V(C), k - |X|)$  is a yes-instance.*

*Proof.* Suppose that the premise of the statement holds and  $k' = k - |X|$ .

( $\Leftarrow$ ) The backward direction is trivial. If  $G - V(C)$  has a feasible solution  $S'$  of size at most  $k'$ , then we can add the minimum sized  $\Pi_i$  vertex deletion set  $X$  of  $G$  and output  $S' \cup X$  has a feasible solution of size  $k' + |X| = k$ .

( $\Rightarrow$ ) We prove the forward direction now. Suppose that  $S^*$  be a feasible solution of size at most  $k$  to  $(G, k)$  and let  $Y = S^* \cap V(C)$ . We prove that  $D = (S^* \setminus Y) \cup X$  is also a feasible solution to  $(G, k)$  and  $|Y| \geq |X|$ . If we manage to prove that  $Y$  is a  $\Pi_i$ -deletion vertex set of  $C$  then we are done. This is because since  $X$  is a minimum  $\Pi_i$ -deletion vertex set of  $C$ ,  $|X| \leq |Y|$ .

We now prove that  $Y$  is indeed a  $\Pi_i$ -deletion vertex set of  $C$ . Suppose not. Then there exist a forbidden set  $Q$  in  $C - Y$ . Note that  $C$  does not contain any forbidden pairs. Hence from Lemma 5,  $Q$  is isomorphic to a graph in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . But in that case, from the definition of Super-Pruned Family, any graph  $H \in \text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  contains an induced subgraph which is isomorphic to some graph in  $\mathcal{F}_i$ . Hence the presence of  $Q$  contradicts that  $Y$  is a  $\Pi_i$ -deletion set. This completes the proof.  $\square$

We are ready to prove our main theorem statement of this section. Let  $f(\mathbf{k}) = \max\{f_1(\mathbf{k}), f_2(\mathbf{k})\}$  where  $f_i(\mathbf{k})\text{poly}(\mathbf{n})$  is the running time for the algorithm  $\mathcal{A}_i$ . Also let  $\mathbf{c}$  be the maximum among the size of graphs in  $\mathcal{G}_1$  and the integer  $\max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2| + \alpha - 2)$ .

**Theorem 3.**  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION can be solved in  $\max\{f(\mathbf{k}), \mathbf{c}^{\mathbf{k}}\}\text{poly}(\mathbf{n})$ -time.

*Proof.* We describe our algorithm as follows. Let  $(G, \mathbf{k})$  be an input instance of  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION. We exhaustively apply Reduction Rule 1 and Branching Rule 2 in sequence to get an instance  $(G', \mathbf{k}')$ . The algorithm finds the closest pair to apply Branching Rule 2 by going over all pairs in  $(H_1, H_2) \in \mathcal{F}_p$  and going over all subsets of size at most  $|V(H_1)| + |V(H_2)|$  of the graph (which is still a polynomial in  $\mathbf{n}$ ) and checking the distance between them. Since the largest sized obstruction in these rules is at most  $\mathbf{c}$ , the bounded search tree of the algorithm so far has  $\mathbf{c}^{\mathbf{k}-\mathbf{k}'}$  nodes. Hence, every component of  $G'$  is such that it is  $\mathcal{F}_p^1$ -free but has graphs in  $\mathcal{F}_p^2$  as induced subgraphs, or vice-versa. In the first case, we invoke the  $f_1(|X|)\text{poly}(\mathbf{n})$ -time algorithm for  $\Pi_1$  VERTEX DELETION on  $G[C]$  to compute a minimum  $\Pi_1$  vertex deletion set  $X$  of  $G[C]$ . In the second case, we invoke the  $f_2(|X|)\text{poly}(\mathbf{n})$ -time algorithm for  $\Pi_2$  VERTEX DELETION to compute a minimum  $\Pi_2$  vertex deletion set  $X$  of  $G[C]$ . The correctness follows from Lemma 6. This creates the total number of nodes in the search tree to  $\mathbf{c}^{\mathbf{k}-\mathbf{k}'}f(\mathbf{k}')$ , bounding the running time to  $\mathbf{c}^{\mathbf{k}-\mathbf{k}'}f(\mathbf{k}')\text{poly}(\mathbf{n})$ . This completes the proof.  $\square$

We now give an approximation algorithm for  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION when for  $i \in \{1, 2\}$ ,  $\Pi_i$  VERTEX DELETION has an approximation algorithm with approximation factor  $\mathbf{c}_i$ .

**Theorem 4.**  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION has a  $\mathbf{d}$ -approximation algorithm where  $\mathbf{d} = \max\{\mathbf{c}, \mathbf{c}_1, \mathbf{c}_2\}$ .

*Proof.* Let  $G$  be the input graph. Let  $S_{\text{OPT}}$  be the minimum sized set such that in the graph  $G - S_{\text{OPT}}$ , every connected component is either in  $\Pi_1$  or  $\Pi_2$ . Let  $|S_{\text{OPT}}| = \text{OPT}$ .

Let us define the family  $\mathcal{S}_1$  as follows. Initially  $\mathcal{S}_1 = \emptyset$ . In polynomial time, we find the closest forbidden pair  $(J^*, T^*)$  in  $G$  with  $P^*$  being a shortest path between the pair, add  $J^* \cup T^* \cup P^*$  to  $\mathcal{S}_1$  and delete  $J^* \cup T^* \cup P^*$  from  $G$ . We repeat this step until it is no longer applicable. Let  $S_1$  be the set of vertices that is present in any pair of graphs in  $\mathcal{S}_1$ . From Lemma 2, we can conclude that any feasible solution of  $G$  must contains a vertex from each member of the family  $\mathcal{S}_1$ . Since the members of  $\mathcal{S}_1$  are pairwise disjoint, we have that  $|S_{\text{OPT}} \cap S_1| \geq |\mathcal{S}_1|$ .

Let  $G' = G - S_1$ . We now construct a set  $S_2$  as follows. Let  $C_1, \dots, C_q$  be the connected components of  $G'$ . If a connected component  $C_i$  has no graphs in  $\mathcal{F}_p^j$  as induced subgraph for  $j \in \{1, 2\}$ , we apply the  $\mathbf{c}_j$ -approximation algorithm for  $\Pi_j$  VERTEX DELETION on  $G'[C_i]$  to obtain a solution  $Z_i$ . The correctness comes from Lemma 6. We have  $S_2 = \bigcup_{i \in [q]} Z_i$ . Since  $(S_{\text{OPT}} - S_1) \cap C_i$  is a feasible solution for the connected component  $C_i$  of  $G'$ , we have that  $|Z_i| \leq (\max\{\mathbf{c}_1, \mathbf{c}_2\})|(S_{\text{OPT}} - S_1) \cap C_i|$  for all  $i \in [q]$ .

We set  $S = S_1 \cup S_2$ . We have

$$\begin{aligned}
|S| &= |S_1| + |S_2| \\
&\leq \left( \max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2| + \alpha - 2) \right) |S_1| + \sum_{i=1}^q |Z_i| \\
&\leq c|S_{\text{OPT}} \cap S_1| + \\
&\quad \sum_{i=1}^q (\max\{c_1, c_2\}) |(S_{\text{OPT}} - S_1) \cap C_i| \\
&\leq (\max\{c, c_1, c_2\}) |S_{\text{OPT}}|
\end{aligned}$$

Thus we have a  $d$ -approximation algorithm for  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION.  $\square$

We now give some examples of  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION.

#### 4.2.1 Cliques or $K_t$ -free graph subclass

We focus on the case of  $\Pi_1$  OR  $\Pi_2$  DELETION when  $\Pi_1$  is the class of cluster graphs (where every connected component of the graph is a clique) and  $\Pi_2$  is any graph class such that the complete graph  $K_t$  for some constant  $t$  is forbidden in this graph and the problem  $\Pi_2$  VERTEX DELETION is known to be FPT. We show that this problem is an example of  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION. We will see later that  $\Pi_2$  can be many of the popular classes including planar graphs, cactus graphs,  $t$ -treewidth graph.

Let us formalize the problem.

**CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION**

**Input:** An undirected graph  $G = (V, E)$ , an integer  $k$  and  $\Pi_2$  is the graph class which is  $K_t$ -free for some constant  $t$  and  $\Pi_2$  VERTEX DELETION has an FPT algorithm  $\mathcal{A}$  with running time  $f_2(k')\text{poly}(n)$  for solution size  $k'$ .

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is either a clique graph, or in  $\Pi_2$ ?

We now show that Conditions 1 - 3 are satisfied by CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION. The CLIQUE VERTEX DELETION problem is same as VERTEX COVER in the complement graph; thus has a  $1.27^k \text{poly}(n)$  [11] time algorithm parameterized by the solution size  $k$ . The  $\Pi_2$  VERTEX DELETION has an  $f_2(k)\text{poly}(n)$  time FPT algorithms parameterized by the solution size  $k$ . Hence Condition 1 is satisfied by CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION. Since  $P_3 \in \mathcal{F}_1$ , Condition 3 is satisfied by CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION. The only condition remaining to be proven is Condition 2 which we do below.

**Lemma 7.** *Condition 2 is satisfied by CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION.*

*Proof.* Let us first infer what the forbidden pair family corresponding to CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION is. We have the forbidden family for  $\Pi_1$  as  $\mathcal{F}_1 = \{P_3\}$ . Note that we don't know what the forbidden family  $\mathcal{F}_2$  for the graph class  $\Pi_2$  is. We only know that the graph  $K_t$  is present in  $\mathcal{F}_2$ . A crucial observation is that this is all needed to infer that the forbidden pair family for CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION.

We know that a graph is  $\mathcal{P}_3$ -free if and only if it is a collection of cliques. Hence, every graph  $H \in \mathcal{F}_2$  that is not a collection of cliques, contains  $\mathcal{P}_3 \in \mathcal{F}_1$  as induced graphs. Hence all such graphs  $H$  belong to the Super-Pruned family  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . We also know that every graph in  $\mathcal{F}_2$  is connected. Hence, the only graphs possibly in  $\mathcal{F}_2$  that is  $\mathcal{P}_3$ -free are the clique graphs  $K_r$  with  $r \neq t$ . Note that  $K_r$  with  $r > t$  contains  $K_t$  as an induced subgraph. Thus, we can ignore these cliques as well. Hence the forbidden pair family is the set of pairs  $(\mathcal{P}_3, K_r)$ ,  $1 \leq r \leq t$  which is of size at most  $t$ .  $\square$

Since all the conditions of  $\mathcal{P}_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION is satisfied by CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION, we have the following theorem.

**Theorem 5.** CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION has an FPT algorithm with running time  $\max\{(t+2)^k, f_2(k)\}\text{poly}(n)$ .

We also give an approximation algorithm for CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION. But we appropriately change the definition of CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION where instead of the assumption that  $\Pi_2$  VERTEX DELETION has an FPT algorithm with running time  $f_2(k')\text{poly}(n)$  for solution size  $k'$ , we assume that  $\Pi_2$  VERTEX DELETION has a polynomial time approximation algorithm with approximation factor  $f_2$ .

Observing that  $d = \max\{c, c_1, c_2\} = \max\{t+2, f_2\}$  from Theorem 4, we have the following theorem.

**Theorem 6.** CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION has an approximation algorithm with approximation factor  $\max\{t+2, f_2\}$ .

We now give examples for the graph class  $\Pi_2$  in CLIQUE-OR- $K_t$ -FREE- $\Pi_2$  DELETION resulting in FPT and approximation algorithms for the corresponding problems.

- Let  $\Pi_2$  be the class of trees. Since triangles are forbidden in trees, we have  $t = 3$ . The problem  $\Pi_2$  VERTEX DELETION corresponds to FEEDBACK VERTEX SET which has a  $3.619^k\text{poly}(n)$  time FPT algorithm [23] and a 2-approximation algorithm [3]. We notice here that the closest  $(\mathcal{P}_3, \mathcal{C}_3)$  pair always intersect on at least two vertices. Hence the branching factor of Branching Rule 2 can be improved to  $t+1 = 4$  in this case.
- Let  $\Pi_2$  be the class of cactus graphs. The graph  $K_4$  is forbidden in cactus graphs as it has two triangles sharing an edge. Hence we have  $t = 4$ . The problem  $\Pi_2$  VERTEX DELETION corresponds to CACTUS VERTEX DELETION which has a  $17.64^k\text{poly}(n)$  time FPT algorithm [2].
- Let  $\Pi_2$  be the class of planar graphs. Since  $K_5$  is not planar, we have  $t = 5$ . The problem  $\Pi_2$  VERTEX DELETION corresponds to PLANAR VERTEX DELETION which has an  $k^{O(k)}\text{poly}(n)$  time FPT algorithm [21] and a  $\log^{O(1)}n$ -approximation algorithm with running time  $n^{O(\log n / \log \log n)}$  [22].
- Let  $\Pi_2$  be the class of  $\eta$ -treewidth graphs. Since  $K_{\eta+2}$  has treewidth  $\eta+1$ , it is forbidden in such graphs. The problem  $\Pi_2$  VERTEX DELETION corresponds to  $\eta$  TREEWIDTH VERTEX DELETION which has a  $2^{O(k)}n^{O(1)}$  time FPT algorithm and a  $O(1)$ -approximation algorithm [15].

We have the following corollary.

**Corollary 1.**  $\Pi_1$  OR  $\Pi_2$  DELETION when  $\Pi_1$  is the class of cliques and  $\Pi_2$  is the class of

- trees has an  $4^k \text{poly}(n)$  time FPT algorithm and a 4-approximation algorithm.
- cactus graphs has an  $17.64^k \text{poly}(n)$  time FPT algorithm.
- planar graphs has an  $k^{O(k)} \text{poly}(n)$  time FPT algorithm and a  $\log^{O(1)} n$ -approximation algorithm with running time  $n^{O(\log n / \log \log n)}$ .
- $\eta$ -treewidth graphs has a  $\max\{(\eta + 4)^k, 2^{O(k)}\} n^{O(1)}$  time FPT algorithm and a  $O(1)$ -approximation algorithm.

#### 4.2.2 Split or Bipartite Graphs

SPLIT-OR-BIPARTITE DELETION

**Input:** An undirected graph  $G = (V, E)$ , an integer  $k$ .

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is either a split graph or bipartite?

The family  $\mathcal{F}_1$  for graphs whose each connected component is a split graph is  $\mathcal{F}_1 = \{C_4, C_5, P_5, \text{necktie}, \text{bowtie}\}$  [6]. A necktie is the graph with vertices  $\{a, b, c, d, e\}$  where  $\{a, b, e\}$  forms a triangle and  $\{a, b, c, d\}$  forms a  $P_4$ . A bowtie is the graph obtained from a necktie by adding the edge  $(b, d)$ . The family  $\mathcal{F}_2$  for graphs whose each connected component is a bipartite graph is the set of odd cycles.

The problems SPLIT VERTEX DELETION and ODD CYCLE TRANSVERSAL has  $1.27^k k^{O(\log k)} \text{poly}(n)$  [13] and  $2.314^k \text{poly}(n)$  time [26] FPT algorithms parameterized by the solution size  $k$  respectively. Hence Condition 1 is satisfied by SPLIT-OR-BIPARTITE DELETION. Since  $P_5 \in \mathcal{F}_1$ , Condition 3 is satisfied by SPLIT-OR-BIPARTITE DELETION.

The graph  $C_5$  is common in both families whereas  $P_5$  is an induced subgraph of odd cycles  $C_i$  with  $i \geq 7$ . Hence the family  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  contains all members in  $\mathcal{F}_2$  except  $C_3$ . Since both necktie and bowtie contains triangles, they are part of  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as well. The only remaining pairs are  $(C_4, C_3)$  and  $(P_5, C_3)$  which forms the forbidden pair family  $\mathcal{F}_p$ . Since it is of size two, Condition 2 is satisfied.

Hence, we have the following corollary from the algorithms for  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION. Observing that the largest obstruction set that we branch on is for the pair  $(C_4, C_3)$  with a path of length at most four between them, we have  $c = 11$ . The SPLIT VERTEX DELETION problem has a 5-approximation algorithm as it can be written as an instance of 5-HITTING SET. The problem ODD CYCLE TRANSVERSAL has a  $\log(\text{OPT})$  approximation algorithm [17] where  $\text{OPT}$  is the size of the optimal solution. The latter dominates the approximation factor  $d$  of the algorithm obtained from Theorem 4.

**Corollary 2.** SPLIT-OR-BIPARTITE DELETION can be solved in  $11^k \text{poly}(n)$ -time and has a  $\log(\text{OPT})$  approximation algorithm where  $\text{OPT}$  is the size of the optimal solution.

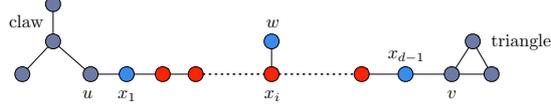


Figure 1: An illustration of a shortest path between a closest (claw, triangle) pair.

### 4.3 Algorithms for $\Pi_1$ OR $\Pi_2$ DELETION without forbidden paths

We have seen examples of  $\Pi_1$  OR  $\Pi_2$  DELETION where even though the families  $\mathcal{F}_i$  are infinite, we manage to come up with fast FPT algorithms. This is mainly thanks to Branching Rule 2 whose branching factor is bounded due to the fact that the path between them is bounded. We now look at examples of  $\Pi_1$  OR  $\Pi_2$  DELETION where paths are not present in the sets  $\mathcal{F}_i$ . Hence the path between the closest forbidden pair is no longer bounded. We observe that for certain pairs of graph classes, there is always an optimal solution that does not intersect the path.

We give a general algorithm for pairs of graph classes where we enforce this condition. We first look at the simple case of CLAW-FREE-OR-TRIANGLE-FREE DELETION as a precursor to the algorithm.

#### 4.3.1 Claw-free or Triangle-Free graphs

We define the problem.

CLAW-FREE-OR-TRIANGLE-FREE DELETION

**Input:** An undirected graph  $G = (V, E)$  and an integer  $k$ .

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is either a claw-free graph, or a triangle-free graph?

The forbidden pair family corresponding to the graph class is of size one which is  $\{(K_{1,3}, C_3)\}$ . We now describe a branching rule corresponding to the closest forbidden pair in the graph.

**Branching Rule 3.** *Let  $(J^*, T^*)$  be the vertex subsets of a closest claw-triangle pair in a connected component of  $G$ , where  $G[J^*]$  is isomorphic to a claw, and  $G[T^*]$  is isomorphic to a triangle. Then for each  $v \in J^* \cup T^*$ , delete  $v$  and decrease  $k$  by 1, resulting in the instance  $(G - v, k - 1)$ .*

We now prove that Branching Rule 3 is safe. Let  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  be a shortest path between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ . Let  $C$  be the connected component of the graph  $G - (J^* \cup T^*)$  containing the internal vertices of  $P^*$ . We have the following lemma.

**Lemma 8.** *The graph corresponding to  $C$  is the path  $P^*$  without its end vertices. Furthermore, the only vertices of  $C$  adjacent to  $J^* \cup T^*$  are  $x_1$  and  $x_{d-1}$  which are only adjacent to vertices  $u$  and  $v$  respectively.*

*Proof.* Suppose that there is a vertex  $w \in V(C) \setminus V(P^*)$  that is adjacent to a vertex  $x_i \in V(P^*)$  with  $1 \leq i \leq d - 1$ . Let us look at the graph induced by the set of vertices  $\{w, x_{i-1}, x_i, x_{i+1}\}$  in  $G$ . If  $w$  is adjacent to  $x_{i-1}$ , then the graph induced by the vertices  $T' = \{w, x_{i-1}, x_i\}$  forms a triangle. Then since  $d_G(J^*, T') < d_G(J^*, T^*)$ , the pair  $(J^*, T')$

contradicts that  $(J^*, T^*)$  is the closest forbidden pair in the graph  $G$ . We can similarly prove that  $w$  is not adjacent to  $x_{i+1}$ . When  $w$  is not adjacent to both the vertices  $x_{i-1}$  and  $x_{i+1}$ , the graph induced by the vertices  $J' = \{w, x_{i-1}, x_i, x_{i+1}\}$  forms a claw. Then since  $d_G(J', T^*) < d_G(J^*, T^*)$ , the pair  $(J', T^*)$  contradicts that  $(J^*, T^*)$  is the closest forbidden pair in the graph  $G$ .

Hence we conclude that there is no vertex  $w \in V(C) \setminus V(P^*)$  that is adjacent to a vertex  $x_i \in V(P^*)$  with  $1 \leq i \leq d-1$ . Since  $C$  is defined as the connected component of graph  $G - (J^* \cup T^*)$  containing the internal vertices of  $P^*$ , the first statement of the claim follows.

The second statement is contradicted only when there is an edge from a vertex  $x_i \in V(P^*)$  with  $1 \leq i \leq d-1$  to the set  $J^* \cup T^*$  apart from the edges  $(u, x_1)$  and  $(x_{d-1}, v)$ . If  $2 \leq i \leq d-2$ , this creates a shorter path  $P'$  between  $J^*$  and  $T^*$  via this edge giving a contradiction. For  $i = 1$ , in the case  $x_1$  is adjacent to some vertex  $w \in J^* \setminus \{u\}$ , we can show that either the graph induced by the set of vertices  $\{w, u, x_1, x_2\}$  is a claw or the graph induced by the set of vertices  $\{w, u, x_1\}$  is a triangle, both contradicting that  $(J^*, T^*)$  is the closest forbidden pair in the graph  $G$ . For  $i = d-1$ , in the case  $x_{d-1}$  is adjacent to  $w \in T^* \setminus \{v\}$ , we can show that the graph induced by the vertices  $\{w, x_{d-1}, v\}$  is a triangle contradicting that  $(J^*, T^*)$  is the closest forbidden pair in the graph  $G$ . This covers all the cases.  $\square$

**Lemma 9.** *Branching Rule 3 is sound.*

*Proof.* Suppose not. In this case, all the optimal solutions for a CLAW-FREE-OR-TRIANGLE-FREE DELETION instance is such that it does not intersect  $J^* \cup T^*$ . Let  $P^* := u, x_1, \dots, x_{d-1}, v$  be a shortest path between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$ . Since the graphs  $G[J^*]$  and  $G[T^*]$  cannot be in the same connected component after deleting the solution, any optimal solution  $X$  has to intersect the set of internal vertices of  $P^*$ .

We now claim that  $X' = (X \setminus (P^* \setminus \{u\})) \cup \{v\}$  is also an optimal solution for  $G$ . Suppose not. Then there is a forbidden set  $Q$  such that  $Q \cap X' = \emptyset$ . Without loss of generality, assume  $Q$  is inclusion-wise minimal; i.e. for all  $q \in Q$ , the graph induced by  $Q \setminus \{q\}$  is not a forbidden set. Note that  $Q$  is a connected set and intersects some vertex  $x_i \in P^* \setminus \{u, v\}$  with  $1 \leq i \leq d-1$  as  $X$  is a feasible solution. The graph  $G[Q]$  contain a forbidden pair for  $(K_{1,3}, C_3)$  and hence contains a cycle. Hence  $Q$  is not fully contained in the connected component  $C$  of the graph  $G \setminus (J^* \cup T^*)$  which is  $P^* \setminus \{u, v\}$  from Lemma 8. Hence we conclude that  $Q$  contains some vertex outside  $P^* \setminus \{u, v\}$  as well.

From Lemma 8, the only neighbors of  $C$  is  $u$  and  $v$  via  $x_1$  and  $x_{d-1}$  respectively. Since  $v \in X'$  and  $Q$  is connected, we can conclude that  $Q$  contains the subpath  $P'$  from  $u$  to  $x_i$ . We now claim that even after deleting the vertices of this subpath from  $Q$  except  $u$ , the set remains forbidden. This contradicts that  $Q$  is a forbidden set as it is not a minimal set.

Since  $Q$  is a forbidden set, it contains vertex subsets that are isomorphic to a claw and a triangle. From Lemma 13, we can conclude that none of the vertices of the subpath  $P'$  can be part of any triangle in  $G$ . None of these vertices can be part of a claw in  $G$  either as it contradicts that  $(J^*, T^*)$  is the closest forbidden pair. Since  $v \in X'$  disconnects the path  $P^*$ , the subpath  $P'$  is not part of a path connecting a claw and a triangle either. Hence the set after removing the vertices of  $P'$  from  $Q$  is still a forbidden set contradicting that  $Q$  is minimal.  $\square$

We are ready to give the algorithm for CLAW-FREE-OR-TRIANGLE-FREE DELETION.

**Theorem 7.** CLAW-FREE-OR-TRIANGLE-FREE DELETION can be solved in  $7^k \text{poly}(n)$  time.

*Proof.* Let  $(G, k)$  be an input instance of CLAW-FREE-OR-TRIANGLE-FREE DELETION. We exhaustively apply Reduction Rule 1 and Branching Rule 3 in sequence to get an instance  $(G', k')$  such that any component of  $G'$  is either claw-free or triangle-free. Note that finding the closest claw-triangle pair can be done by going over all subsets of size at most 7, checking if they do induce a claw and a triangle and finding the shortest path between them. The correctness follows from Lemma 9. If  $k' < 0$ , we return no-instance. Else, we return yes-instance.

Since we branch on a set of size at most 7 in Branching Rule 3, the bounded search tree of the algorithm has at most  $7^k$  nodes. This bounds the running time to  $7^k n^{O(1)}$ . This completes the proof.  $\square$

We also give an approximation algorithm for CLAW-FREE-OR-TRIANGLE-FREE DELETION using similar ideas.

**Theorem 8.** CLAW-FREE-OR-TRIANGLE-FREE DELETION has a 7-approximation algorithm.

*Proof.* Let  $G$  be the input graph. The approximation algorithm for CLAW-FREE-OR-TRIANGLE-FREE DELETION is as follows. Let  $\mathcal{S}_1$  be a family of sets initialized to  $\emptyset$ . We find a closest forbidden pair  $(J^*, T^*)$  in  $G'$ , add  $J^* \cup T^*$  to  $\mathcal{S}_1$  and delete  $J^* \cup T^*$  from  $G'$ . We repeat this step until it is no longer applicable. Let  $S_{\text{OPT}}$  be the minimum sized set such that in the graph  $G - S_{\text{OPT}}$ , every connected component is either a claw-free graph or a triangle-free graph. Let  $|S_{\text{OPT}}| = \text{OPT}$ . Let  $\mathcal{S}_1$  be the set of vertices that is present in any set in  $\mathcal{S}_1$ . From the safeness proof of Branching Rule 3, we can conclude that any feasible solution of  $G$  must contain a vertex from each set of the family  $\mathcal{S}_1$ . Since the union of all the sets in  $\mathcal{S}_1$  is a feasible solution, we have that  $|\mathcal{S}_1| = 7|\mathcal{S}_1| \leq 7|S_{\text{OPT}}|$ .

Thus we have a 7-approximation algorithm for CLAW-FREE-OR-TRIANGLE-FREE DELETION.  $\square$

### 4.3.2 Algorithm for SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION

Now, we show that the algorithm ideas from CLAW-FREE-OR-TRIANGLE-FREE DELETION are applicable for a larger number of pairs of graph classes. Later in Section 5, we give other examples for pairs of graph classes where the same ideas work.

We define a variant of  $\Pi_1$  OR  $\Pi_2$  DELETION called SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION satisfying the following properties.

1. The vertex deletion problems for the graph classes  $\Pi_1$  and  $\Pi_2$  are FPT with algorithms to the respective classes being  $\mathcal{A}_1$  and  $\mathcal{A}_2$ .
2.  $\mathcal{F}_p$ , the forbidden pair family of  $\mathcal{F}_1$  and  $\mathcal{F}_2$  is of constant size.
3. Let  $(H_1, H_2) \in \mathcal{F}_p$  be a closest forbidden pair in the graph  $G$  with  $(J_1, J_2)$  being the vertex subsets corresponding to the pair. Let  $P$  be a shortest path between  $J_1$  and  $J_2$ . There is a family  $\mathcal{G}_1$  such that

- $\mathcal{G}_1$  is a finite family of graphs of bounded-size (independent of the size of  $G$ ), and
- in the graph  $G$  that is  $\mathcal{G}_1$ -free, if a forbidden set  $Q$  intersects the internal vertices of  $P$ , then  $Q$  contains the right endpoint of  $P$ .

**SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION**

**Input:** An undirected graph  $G = (V, E)$ , graph classes  $\Pi_1, \Pi_2$  with associated forbidden families  $\mathcal{F}_1$  and  $\mathcal{F}_2$  such that Conditions 1 - 3 are satisfied and an integer  $k$ .

**Parameter:**  $k$

**Question:** Is there a vertex set  $S$  of size at most  $k$  such that every connected component of  $G - S$  is either in  $\Pi_1$  or in  $\Pi_2$ ?

Note that the first two conditions for the problem are the same as those in  $P_\alpha$ -FREE- $(\Pi_1, \Pi_2)$ -DELETION. Only the Condition 3 is changed which is tailored to prove the soundness of the branching rule we introduce. We also note that in the examples of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION we looked at (see Section 5),  $\mathcal{G}_1$  is the collection of all the graphs in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  of some constant size, with the constant depending on the problem.

Towards an FPT algorithm for SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION, We give the following branching rule whose soundness is easy to see.

**Branching Rule 4.** *Let  $(G, k)$  be the input instance and let  $Q \subseteq V(G)$  such that  $G[Q]$  is isomorphic to a graph in  $\mathcal{G}_1$ . Then, for each  $v \in V(Q)$ , delete  $v$  from  $G$  and decrease  $k$  by 1. The resulting instance is  $(G - v, k - 1)$ .*

From here on we assume that Branching Rule 4 is not applicable for  $G$  and so  $G$  is  $\mathcal{G}_1$ -free. We now focus on connected components of  $G$  which contain forbidden pairs. We have the following branching rule.

**Branching Rule 5.** *Let  $(J^*, T^*)$  be the vertex subsets of a closest forbidden pair  $(H_1, H_2) \in \mathcal{F}_p$ . Then for each  $v \in J^* \cup T^*$ , we delete  $v$  and decrease  $k$  by 1, resulting in the instance  $(G - v, k - 1)$ .*

We now prove the correctness of the above branching rule.

**Lemma 10.** *Branching Rule 5 is safe.*

*Proof.* Let  $P^*$  be a shortest path between  $J^*$  and  $T^*$  with endpoints  $u \in J^*$  and  $v \in T^*$ . Since  $G[J^*]$  and  $G[T^*]$  cannot occur in the same connected component after deleting the solution, the solution must intersect  $J^* \cup T^* \cup P^*$ . We now prove that there exists an optimal solution of  $(G, k)$  that does not intersect the internal vertices of  $P^*$ .

Let  $X$  be an optimal solution such that  $X \cap (J^* \cup T^*) = \emptyset$ . Since  $X \cap (J^* \cup T^* \cup P^*) \neq \emptyset$ ,  $X$  must intersect the internal vertices of  $P^*$ . We claim that  $X' = (X \setminus (P^* \setminus \{u\})) \cup \{v\}$  is also an optimal solution for  $G$ . Suppose not. Then there exists a forbidden set  $Q$  such that  $X' \cap Q = \emptyset$ . Since  $X \cap Q \neq \emptyset$ , we know that  $Q$  intersects the internal vertices of  $P^*$ . But then by Condition 3, we know that  $Q$  contains  $v$  as well giving a contradiction.  $\square$

We now give the FPT algorithm SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION which is the same algorithm in Theorem 3, but the Branching Rule 2 is replaced by Branching Rule 4 and Branching Rule 5 in sequence. The correctness comes from Lemmas 10 and 6.

Again we define  $f(\mathbf{k}) = \max\{f_1(\mathbf{k}), f_2(\mathbf{k})\}$  where  $f_i(\mathbf{k})\text{poly}(\mathbf{n})$  is the running time for the algorithm  $\mathcal{A}_i$ . Also let  $\mathbf{c}$  be the maximum among the size of graphs in  $\mathcal{G}_1$  and the integer  $\max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2|)$ . Note that the branching factor of Branching Rule 5 is reduced to  $\max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2|)$  as we do not branch on the vertices of the path between the vertex sets of the closest forbidden pair.

**Theorem 9.** SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION can be solved in  $\max\{f(\mathbf{k}), \mathbf{c}^k\}\text{poly}(\mathbf{n})$ -time.

We now give an approximation algorithm for SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION when for  $i \in \{1, 2\}$ ,  $\Pi_i$  VERTEX DELETION has an approximation algorithm with approximation factor  $\mathbf{c}_i$ . The algorithm is similar to that of Theorem 4 with an additional primary step of greedily adding vertex subsets of induced graphs isomorphic to members in the family  $\mathcal{G}_1$  to the solution. Note that any optimal solution should contain at least one of the vertices of each such vertex subset.

**Theorem 10.** SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION has a  $\mathbf{d}$ -approximation algorithm where  $\mathbf{d} = \max\{\mathbf{c}, \mathbf{c}_1, \mathbf{c}_2\}$ .

*Proof.* Let  $\mathbf{G}$  be the input graph. Let  $\mathbf{S}_{\text{OPT}}$  be the minimum sized set such that in the graph  $\mathbf{G} - \mathbf{S}_{\text{OPT}}$ , every connected component is either in  $\Pi_1$  or  $\Pi_2$ . Let  $|\mathbf{S}_{\text{OPT}}| = \text{OPT}$ .

Let  $\mathcal{S}_1$  denote the maximal family of graphs which are in  $\mathcal{G}_1$  such that any two members of  $\mathcal{S}_1$  are pairwise disjoint. We can greedily construct such a family  $\mathcal{S}_1$  in polynomial time. Let  $\mathbf{S}_1$  be the set of vertices that is present in any graphs in  $\mathcal{S}_1$ . From Lemma 5, we can conclude that any feasible solution of  $\mathbf{G}$  must contains a vertex from each member of the family  $\mathcal{S}_1$ . Since the members of  $\mathcal{S}_1$  are pairwise disjoint, we have that  $|\mathbf{S}_{\text{OPT}} \cap \mathbf{S}_1| \geq |\mathcal{S}_1|$ .

Let  $\mathbf{G}' = \mathbf{G} - \mathbf{S}_1$ . We now construct a family  $\mathcal{S}_2$  as follows. Initially  $\mathcal{S}_2 = \emptyset$ . We find the closest forbidden pair  $(\mathbf{J}^*, \mathbf{T}^*)$  in  $\mathbf{G}'$ , add it to  $\mathcal{S}_2$  and delete  $\mathbf{J}^* \cup \mathbf{T}^*$  from  $\mathbf{G}'$ . We repeat this step until it is no longer applicable. Let  $\mathbf{S}_2$  be the set of vertices that is present in any pair of graphs in  $\mathcal{S}_2$ . From Lemma 10, we can conclude that any feasible solution of  $\mathbf{G}$  must contains a vertex from each pair of the family  $\mathcal{S}_2$ . Since the members of  $\mathcal{S}_2$  are pairwise disjoint and  $\mathbf{S}_{\text{OPT}} - \mathbf{S}_1$  is also an optimum solution for  $\mathbf{G}'$ , we have that  $|(\mathbf{S}_{\text{OPT}} - \mathbf{S}_1) \cap \mathbf{S}_2| \geq |\mathcal{S}_2|$ .

Let  $\mathbf{G}'' = \mathbf{G}' - \mathbf{S}_2$ . We now construct a set  $\mathcal{S}_3$  as follows. Let  $\mathbf{C}_1, \dots, \mathbf{C}_q$  be the connected components of  $\mathbf{G}''$ . If a connected component  $\mathbf{C}_i$  has no graphs in  $\mathcal{F}_p^1$  as induced subgraph, we apply the  $\mathbf{c}_1$ -approximation algorithm for  $\Pi_1$  vertex deletion on  $\mathbf{G}''[\mathbf{C}_i]$  to obtain a solution  $\mathbf{S}_i$ . If a connected component  $\mathbf{C}_i$  no has no graphs in  $\mathcal{F}_p^2$  as induced subgraph, we apply the  $\mathbf{c}_2$ -approximation algorithm for  $\Pi_2$  vertex deletion on  $\mathbf{G}''[\mathbf{C}_i]$  to obtain a solution  $\mathbf{Z}_i$ . The correctness comes from Lemma 6. We have  $\mathbf{S}_3 = \bigcup_{i \in [q]} \mathbf{Z}_i$ . Since  $((\mathbf{S}_{\text{OPT}} - \mathbf{S}_1) - \mathbf{S}_2) \cap \mathbf{C}_i$  is a feasible solution for the connected component  $\mathbf{C}_i$  of  $\mathbf{G}''$ , we have that  $|\mathbf{Z}_i| \leq (\max\{\mathbf{c}_1, \mathbf{c}_2\})|((\mathbf{S}_{\text{OPT}} - \mathbf{S}_1) - \mathbf{S}_2) \cap \mathbf{C}_i|$  for all  $i \in [q]$ .

We set  $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$ . We have

$$\begin{aligned}
|\mathcal{S}| &= |\mathcal{S}_1| + |\mathcal{S}_2| + |\mathcal{S}_3| \\
&\leq (\max_{H \in \mathcal{G}_1} \{|H|\})|\mathcal{S}_1| + (\max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2|))|\mathcal{S}_2| + \\
&\quad \sum_{i=1}^q (\max\{c_1, c_2\})|((\mathcal{S}_{\text{OPT}} - \mathcal{S}_1) - \mathcal{S}_2) \cap C_i| \\
&\leq c|\mathcal{S}_{\text{OPT}} \cap \mathcal{S}_1| + c|(\mathcal{S}_{\text{OPT}} - \mathcal{S}_1) \cap \mathcal{S}_2| + \\
&\quad \sum_{i=1}^q (\max\{c_1, c_2\})|((\mathcal{S}_{\text{OPT}} - \mathcal{S}_1) - \mathcal{S}_2) \cap C_i| \\
&\leq (\max\{c, c_1, c_2\})|\mathcal{S}_{\text{OPT}}|
\end{aligned}$$

Thus we have a  $d$ -approximation algorithm for SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION.  $\square$

## 5 Examples of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION

Verifying whether Conditions 2 and 3 are satisfied for a general  $\Pi_1$  and  $\Pi_2$  is non-trivial. Hence we look at specific pairs of graph classes  $\Pi_1$  and  $\Pi_2$  and prove that they are examples of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION.

We start by showing that the problems CLAW-FREE-OR-TRIANGLE-FREE DELETION is indeed an example of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION.

**Lemma 11.** CLAW-FREE-OR-TRIANGLE-FREE DELETION *is an example of* SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION.

*Proof.* We show that Conditions 1 - 3 are satisfied by CLAW-FREE-OR-TRIANGLE-FREE DELETION. The problems CLAW-TREE VERTEX DELETION and TRIANGLE VERTEX DELETION has simple  $4^k \text{poly}(n)$  and  $3^k \text{poly}(n)$  time FPT algorithms parameterized by the solution size  $k$  via a simple branching on claws and triangles respectively. Hence Condition 1 is satisfied by CLAW-FREE-OR-TRIANGLE-FREE DELETION. The forbidden pair family for CLAW-FREE-OR-TRIANGLE-FREE DELETION is of size one which is  $\{K_{1,3}, C_3\}$ . Hence Condition 2 is satisfied. Finally, from Lemma 9, we can conclude that Condition 3 is satisfied as well with  $\mathcal{G}_1 = \emptyset$ .

Hence CLAW-FREE-OR-TRIANGLE-FREE DELETION is an example of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION. We have  $f(k) = \max\{4^k, 3^k\} = 4^k$  and  $c = \max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2|) = 7$ . Hence from Theorem 9, we have a  $7^k \text{poly}(n)$  time algorithm for CLAW-FREE-OR-TRIANGLE-FREE DELETION. This is the same running time obtained independently in Theorem 7.

We also have  $d = \max\{c, c_1, c_2\} = 7$  giving a 7-approximation for CLAW-FREE-OR-TRIANGLE-FREE DELETION from Theorem 10, which is the same approximation factor obtained independently in Theorem 8.  $\square$

We now give examples of other pairs of graph classes  $\Pi_1$  and  $\Pi_2$  whose scattered deletion problem is an example of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION. The core part in each of the cases below is establishing that Condition 3 is satisfied. We do so by establishing structural properties for the shortest path corresponding to the closest forbidden pair. Such properties vary for each case.

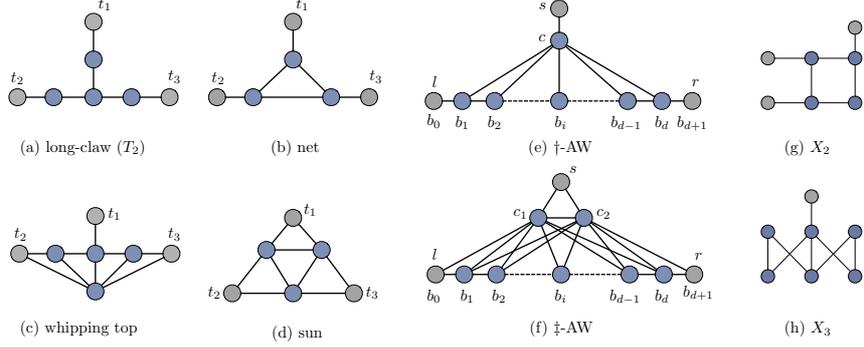


Figure 2: Obstructions for Graph Classes

## 5.1 Interval or Trees

We define the problem.

**INTERVAL-OR-TREE DELETION**

**Input:** An undirected graph  $G = (V, E)$  and an integer  $k$ .

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is either an interval graph, or a tree?

We have the following forbidden subgraph characterization of interval graphs.

**Lemma 12.** ([24]) *A graph is an interval graph if and only if it does not contain net, sun, hole, whipping top, long-claw, †-AW, or ‡-AW as its induced subgraphs.*

See Figure 2 for an illustration of the graphs mentioned as forbidden subgraphs for interval graphs. Note that †-AW and ‡-AW are a collection of graphs (like holes) and its size need not be constant.

We now give a characterization for graphs whose every connected component is an interval graph or a tree. Recall from Section 4.1 that the forbidden pair family  $\mathcal{F}_p$  of this pair of graph classes is (long-claw, triangle) and  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  is {net, sun, hole, whipping top, †-AW, ‡-AW}. The following is a corollary from Lemma 5.

**Corollary 3.** *The following statements are equivalent.*

1. *Let  $G$  be a graph such that every connected component is either an interval graph or a tree.*
2.  *$G$  does not have any net, sun, hole, whipping top, †-AW, ‡-AW as its induced subgraphs. Moreover,  $G$  cannot have long-claw and triangle as induced subgraphs in the same connected component.*

We show that Conditions 1 - 3 are satisfied by INTERVAL-OR-TREE DELETION. The problems INTERVAL VERTEX DELETION and FEEDBACK VERTEX SET has  $10^k \text{poly}(n)$  [9] and  $3.619^k \text{poly}(n)$  [23] time FPT algorithms parameterized by the solution size  $k$  respectively. Hence Condition 1 is satisfied by INTERVAL-OR-TREE DELETION. The forbidden pair family for INTERVAL-OR-TREE DELETION is of size one which is the pair (long-claw, triangle). Hence Condition 2 is satisfied.

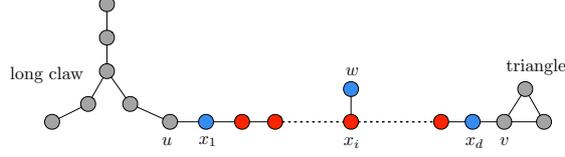


Figure 3: An illustration of a shortest path between a closest (long-claw, triangle) pair.

It remains to show that Condition 3 is satisfied for INTERVAL-OR-TREE DELETION. We define  $\mathcal{G}_1$  be the family of graphs in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  of size at most 10.

Let  $(J^*, T^*)$  be the vertex subsets of a closest long-claw, triangle pair in a connected component of  $\mathbf{G}$ , where  $J^*$  is a long-claw and  $T^*$  is a triangle. Let  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  be a shortest path between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ .

A *caterpillar* graph is a tree in which all the vertices are within distance 1 of a central path. In the graph  $\mathbf{G}$ , let  $\mathbf{C}$  be the connected component of  $\mathbf{G} - (J^* \cup T^*)$  containing the internal vertices of  $P^*$ . We have the following lemma that helps us to prove Condition 3.

**Lemma 13.** *The graph  $\mathbf{C}$  is a caterpillar with the central path being  $P^*$ . Furthermore, the only vertices of  $\mathbf{C}$  adjacent to  $J^* \cup T^*$  are  $x_1$  and  $x_{d-1}$  which are only adjacent to  $x_0$  and  $x_d$  respectively.*

*Proof.* We first look at the neighborhood vertices of the path  $P^*$  in the connected component  $\mathbf{C}$ . Let  $w$  be such a vertex which is adjacent to a vertex  $x_i$  with  $i \in \{1, 2, \dots, d-1\}$ . We prove that  $w$  is not adjacent to any other vertex in  $\mathbf{G}$ . Thus there are no cycles in  $\mathbf{C}$  and all the vertices in  $\mathbf{C}$  are at distance at most 1 to  $P^*$  proving that  $\mathbf{C}$  is a caterpillar.

We go over the possibilities of edges from  $w$  to other vertices.

**Case 1:** Suppose  $w$  is adjacent to some vertex  $x_j$  with  $j \in \{0, \dots, d\}$ . If  $j = i+1$  or  $j = i-1$ , then  $wx_ix_j$  forms a triangle  $T'$ . Since the path  $P' = u, x_1, \dots, x_i$  has length smaller than  $P^*$  and connects  $J^*$  and  $T'$ , we have that  $d_G(J^*, T') < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a pair of long-claw and triangle that is the closest.

Hence  $w$  is not adjacent to  $x_{i-1}$  and  $x_{i+1}$ . Suppose  $j = i+2$  or  $j = i-2$ . Then the graph induced by the set of vertices  $\{w, x_i, x_{i+1}, x_{i+2}\}$  or  $\{w, x_i, x_{i-1}, x_{i-2}\}$  is  $C_4$  contradicting that the graph is  $\mathcal{G}_1$ -free. Hence  $w$  is not adjacent to  $x_{i-2}$  and  $x_{i+2}$ .

Suppose  $w$  is adjacent to  $x_j$  with  $1 \leq j < i-2$  or  $i+2 < j \leq d$ . Then note that we have a path from  $x_j$  to  $x_i$  of length 2 via  $w$  which is shorter than the path  $x_i \dots x_j$  along  $P^*$ . Hence we have a path  $P' = ux_1 \dots x_j wx_i \dots x_d v$  or  $P' = ux_1 \dots x_i wx_j \dots x_d v$  from  $J^*$  to  $T^*$  of length smaller than  $P^*$  contradicting that  $P^*$  is the shortest such path.

Hence  $w$  is not adjacent to any of the vertices in  $P^*$ .

**Case 2:** Suppose  $w$  is adjacent to a vertex  $u' \in J^*$ . We assume without loss of generality that among all neighbors of  $w$  in  $J^*$ ,  $u'$  is the vertex that is closest to  $u$  in  $J^*$ , i.e.,  $d_{J^*}(u, u')$  is minimum. If  $3 \leq i \leq d-1$ , we have a path from  $u'$  to

$x_i$  of length 2 via  $w$  which is shorter than the path from  $u$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ .

Suppose  $i = 1$  or  $i = 2$ . Let  $P'$  denote a shortest path between  $u$  and  $u'$  in  $J^*$ . Since  $J^*$  is a long-claw, the length of  $P'$  is at most 4. Let us concatenate  $P'$  with the prefix of the path  $P^*$  from  $u$  to  $x_i$  which is of length at most 2. Then we get a shortest path from  $u'$  to  $x_i$  which is of length at least 2 and at most 6. The vertex  $w$  is adjacent to only  $u'$  and  $x_i$  in this path. Hence the graph induced by the set of vertices in this path plus  $w$  is cycle  $C_j$  with  $4 \leq j \leq 8$  contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence  $w$  is not adjacent to any of the vertices in  $J^*$ .

**Case 3:** Suppose  $w$  adjacent to a vertex  $v' \in T^*$ . We have  $d_{T^*}(v, v') = 1$  as  $T^*$  is a triangle. If  $1 \leq i \leq d-3$ , we have a path from  $v'$  to  $x_i$  of length 2 via  $w$  which is shorter than the path from  $v$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ .

Hence  $d-2 \leq i \leq d-1$ . Let  $P'$  be the suffix of the path  $P^*$  from  $x_i$  to  $v$ . Then we get a shortest path from  $x_i$  to  $v'$  as  $x_i P' v'$  which is of length at least 2 and at most 3. The vertex  $w$  is adjacent with only vertices  $x_i$  and  $v'$  in this path. Hence the graph induced by the set of vertices of this path plus  $w$  forms a cycle  $C_j$  with  $4 \leq j \leq 5$  contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence from the above three cases,  $w$  is adjacent to none of the other vertices of  $J^* \cup T^* \cup P^*$ .

**Case 4:** We now prove that  $w$  is not adjacent to any other vertex  $w' \in V(C) \setminus P^*$ . Suppose that there exists such a vertex  $w'$ . We now look at various cases of the adjacency of  $w'$  with other vertices.

- **Case 4.1:** We first look at adjacencies of  $w'$  with vertices in  $P^*$ .

Suppose  $w'$  is adjacent to vertex  $x_i$ . Then the graph induced by the set of vertices  $T' = \{w', w, x_i\}$  forms a triangle. Since the path  $P' := u, x_1, \dots, x_i$  has length smaller than  $P^*$  and connects  $J^*$  and  $T'$ , we have that  $d_G(J^*, T') < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a pair of long-claw and triangle that is the closest.

Suppose  $w'$  is adjacent to  $x_{i+1}$  or  $x_{i-1}$ . Then the graph induced by the set of vertices  $\{w', w, x_i, x_{i+1}\}$  or  $\{w', w, x_i, x_{i-1}\}$  forms a  $C_4$  contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence this is not the case. Now suppose  $w'$  is adjacent to  $x_{i+2}$  or  $x_{i-2}$ . Then the graph induced by the set of vertices  $\{w', w, x_i, x_{i+1}, x_{i+2}\}$  or  $\{w', w, x_i, x_{i-1}, x_{i-2}\}$  forms a  $C_5$  contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence this is also not the case. Suppose  $w'$  is adjacent to  $x_{i+3}$  or  $x_{i-3}$ . Then the graph induced by the set of vertices  $\{w', w, x_i, x_{i+1}, x_{i+2}, x_{i+3}\}$  or  $\{w', w, x_i, x_{i-1}, x_{i-2}, x_{i-3}\}$  forms a  $C_6$  contradicting that the graph is  $\mathcal{G}_1$ -free.

In the remaining case,  $w'$  is adjacent to  $x_j$  with  $1 \leq j < i+3$  or  $i+3 < j \leq d$ . Hence  $|j - i| > 3$ . Then note that the path  $x_j, w', w, x_i$  is of length three which is shorter than the path between  $x_i$  and  $x_j$  in  $P^*$ . Hence we get a path  $P'$  from  $u$  to  $v$  of length smaller than  $P^*$  contradicting that  $P^*$  is the smallest such path. Hence  $w'$  is not adjacent to any of the vertices in  $P^*$ .

- **Case 4.2:** Suppose  $w'$  adjacent to a vertex  $u' \in J^*$ . We assume without loss of generality that among all neighbors of  $w$  in  $J^*$ ,  $u'$  is the vertex that is closest to  $u$  in  $J^*$ , i.e,  $d_{J^*}(u, u')$  is minimum. If  $3 \leq i \leq d-1$ , observe that the path  $u'w'wx_i$  from  $u'$  to  $x_i$  is of length 3 which is shorter than the path from  $u$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $1 \leq i \leq 3$ . Let  $P'$  denote the path between  $u$  and  $u'$  in  $J^*$  and  $P''$  be the prefix of the path  $P^*$  from  $u$  to  $x_i$ . Then  $uP''x_iww'u'P'u$  forms a cycle  $C_j$  with  $4 \leq j \leq 10$  with no chords contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence  $w$  is not adjacent to any of the vertices in  $J^*$ .

- **Case 4.3:** Suppose  $w'$  adjacent to a vertex  $v' \in T^*$ . We have  $d_{T^*}(v, v') = 1$  as  $T^*$  is a triangle. If  $1 \leq i \leq d-4$ , we have the path  $v', w', w, x_i$  from  $v'$  to  $x_i$  of length 3 which is shorter than the path from  $v$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $d-3 \leq i \leq d-1$ . Let  $P'$  be the suffix of the path  $P^*$  from  $x_i$  to  $v$ . Then  $v'w'wx_iP'vv'$  forms a cycle  $C_j$  with  $4 \leq j \leq 6$  without any chords contradicting that the graph is  $\mathcal{G}_1$ -free.

Hence we conclude that  $w'$  is not adjacent to any of the vertices in  $J^* \cup T^* \cup P^*$ . Now look the graph induced by the set of vertices  $J'$  which is

- $\{w', w, x_i, x_{i-1}, x_{i-2}, x_{i+1}, x_{i+2}\}$  for  $3 \leq i \leq x_{d-2}$  or
- $\{w', w, x_i, x_{i-1}, u, x_{i+1}, x_{i+2}\}$  when  $i = 2$  or
- $\{w', w, x_i, u, u', x_{i+1}, x_{i+2}\}$  when  $i = 1$  for  $u' \in J^* \cap N(u)$  or
- $\{w', w, x_i, x_{i-1}, x_{i-2}, v, v'\}$  when  $i = d-1$  for  $v' \in J^* \cap N(v)$ .

In all cases, the graph induced by  $J'$  forms a long-claw. Since the path  $P'$  from  $J'$  to  $v \in T^*$  has length smaller than  $P^*$ , we have that  $d_G(J', T^*) < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a pair of long-claw and triangle that is closest.

Hence no such vertex  $w'$  exists and therefore  $w$  has no other neighbors in  $G$ .

Hence, the graph  $C$  is a caterpillar with the central path being  $P^*$ . Furthermore, no vertices other than  $x_1$  and  $x_{d-1}$  is adjacent to  $J^* \cup T^*$ .  $\square$

We now use Lemma 13 to prove that Condition 3 is satisfied for INTERVAL-OR-TREE DELETION.

**Lemma 14.** *Condition 3 is satisfied for INTERVAL-OR-TREE DELETION.*

*Proof.* Condition 3 is not satisfied in the following case. There exist a pair  $(J^*, T^*)$  which is the vertex subsets of a closest long-claw, triangle pair in a connected component of  $G$ , where  $J^*$  is a long-claw and  $T^*$  is a triangle. Also there is a shortest path  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ . A forbidden set  $Q$  of the graph  $G$  is such that  $Q$  contains some internal vertex  $x_i$  of the path  $P$  but it does not contain  $v$ .

Since the graph  $G$  is  $\mathcal{G}_1$ -free,  $G[Q]$  can be one of hole,  $\dagger$ -AW or a  $\ddagger$ -AW or contain a forbidden pair for (long-claw, triangle). Note that all of these possibilities contain cycles. But from Lemma 13, the component  $C$  of  $G \setminus (J^* \cup T^*)$  that contains the internal

vertices of  $P^*$  is a caterpillar which does not contain any cycles. Hence  $Q$  is not fully contained in  $C$ .

From Lemma 13, the only neighbors of  $C$  are  $u$  and  $v$  via  $x_1$  and  $x_{d-1}$  respectively. Since  $G[Q]$  is connected and intersects  $x_i$ ,  $Q \cap \{u, v\} \neq \emptyset$ . Since we assumed that  $Q$  does not contain the vertex  $v$ , we have  $u \in Q$ . In particular,  $Q$  contains the entire subpath of  $P^*$  from  $u$  to  $x_i$ .

Also note that  $u$  cannot be part of a subset of three vertices  $T'$  which is a triangle as otherwise, we get a pair  $(J^*, T')$  with distance zero contradicting that  $(J^*, T^*)$  was the closest pair  $P^*$  has internal vertices. Hence the forbidden set  $Q$  cannot be  $\dagger$ -AW or a  $\ddagger$ -AW whose structure forces  $u$  to be part of a triangle if it contains  $x_i$  (which also happens only in the case when  $i = 1$ ). Since  $x_i$  does not have any paths to the vertex  $u$  other than the subpath in  $P^*$ , the forbidden set  $Q$  cannot be a hole as well.

Hence  $Q$  can only correspond to a (long-claw, triangle) forbidden pair. In this case, we claim that the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is also a forbidden set. This contradicts that  $Q$  is a forbidden set as by definition they are required to be minimal.

Since  $Q$  is a forbidden set, it contains vertex subsets that are isomorphic to a long-claw and a triangle. From Lemma 13, we can conclude that none of the vertices  $x_1, \dots, x_i$  can be part of any triangle in  $G$ . None of these vertices can be part of a long-claw in  $G$  as well since it contradicts that  $(J^*, T^*)$  is the closest forbidden pair. Since  $v$  disconnects the path  $P^*$ , the subpath  $x_1, \dots, x_i$  is not part of a path connecting a long-claw and a triangle either as if so  $Q$  must contain the entire path  $P^*$  including  $v$ . Hence the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is still a forbidden set contradicting that  $Q$  is minimal.

These cases of  $Q$  are mutually exhaustive completing the proof of the Lemma.  $\square$

Hence, we have established that INTERVAL-OR-TREE DELETION is indeed an example of SPECIAL INFINITE- $(\Pi_1, \Pi_2)$ -DELETION. We have the following theorem.

**Theorem 11.** INTERVAL-OR-TREE DELETION has an FPT algorithm with running time  $10^k \text{poly}(n)$  and a 10-approximation algorithm.

*Proof.* We have  $f(k) = \max\{10^k, 3.619^k\} = 10^k$  and  $c = \max_{(H_1, H_2) \in \mathcal{F}_p} (|H_1| + |H_2|) = 10$ . Hence from Theorem 9, we have a  $10^k \text{poly}(n)$  time algorithm for INTERVAL-OR-TREE DELETION.

We know that INTERVAL VERTEX DELETION has an 8-approximation algorithm [8] and FEEDBACK VERTEX SET has a 2-approximation algorithm [3]. Hence  $d = \max\{c, c_1, c_2\} = 10$  giving a 10-approximation for INTERVAL-OR-TREE DELETION from Theorem 10.  $\square$

## 5.2 Proper Interval or Trees

We define the problem.

PROPER INTERVAL-OR-TREE DELETION

**Input:** An undirected graph  $G = (V, E)$  and an integer  $k$

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is a proper interval graph or a tree?

We have the following forbidden subgraph characterization of proper interval graphs.

**Lemma 15.** [5] *A graph is said to be a proper interval graph if and only if it does not contain claw, net, sun or hole as its induced subgraphs.*

We now give a characterization for graphs whose every connected component is a proper interval graph or a tree.

**Lemma 16.** *The following statements are equivalent.*

1. *A graph  $G$  is such that every connected component of  $G$  is a proper interval graph or a tree.*
2. *A graph  $G$  does not have any net, sun or hole as its induced subgraphs. Moreover, no connected component of  $G$  have a claw and a triangle as induced graphs.*

*Proof.* We prove that the forbidden pair family is (claw, triangle). The forbidden family  $\mathcal{F}_1$  for proper interval graphs are claw, net, sun or holes. The forbidden family of trees  $\mathcal{F}_2$  is cycles. Since cycles of length at least 4 are common in  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , they are in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . Since the graphs net and sun have triangle as induced subgraph, they are in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as well. The only remaining pair in  $\mathcal{F}_1 \times \mathcal{F}_2$  is (claw, triangle). The proof now follows from the forbidden characterization in Lemma 5.  $\square$

Hence Condition 2 is satisfied by PROPER INTERVAL-OR-TREE DELETION as the forbidden pair is of size one which is (claw, triangle). Since PROPER INTERVAL VERTEX DELETION is FPT with a  $O^*(6^k)$  running time algorithm from [27] and FEEDBACK VERTEX SET is FPT with a  $O^*(3.619^k)$  running time algorithm from [23], Condition 1 is satisfied as well.

We now prove that Condition 3 is satisfied by PROPER INTERVAL-OR-TREE DELETION. Recall Lemma 8 where we established that the internal vertices of any shortest path between the vertex sets of a closest claw, triangle pair in a graph do not contain any neighbors other than the endpoints of the path.

**Lemma 17.** *Condition 3 is satisfied by PROPER INTERVAL-OR-TREE DELETION.*

*Proof.* Condition 3 is not satisfied in the following case. There exist a pair  $(J^*, T^*)$  which is the vertex subsets of a closest claw, triangle pair in a connected component of  $G$ , where  $J^*$  is a claw and  $T^*$  is a triangle. Also there is a shortest path  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ . A forbidden set  $Q$  of the graph  $G$  is such that  $Q$  contains some internal vertex  $x_i$  of the path  $P$  but it does not contain  $v$ .

Let  $\mathcal{G}_1 = \emptyset$ . The graph  $G[Q]$  can be one of net, sun, hole or contain a forbidden pair for (claw, triangle). Note that all of these possibilities contain cycles. But from Lemma 8, the component  $C$  of  $G \setminus (J^* \cup T^*)$  that contains the internal vertices of  $P^*$  is the path  $P^* \setminus \{u, v\}$  which does not contain any cycles. Hence  $Q$  is not fully contained in  $C$ .

From Lemma 8, the only neighbors of  $C$  is  $u$  and  $v$  via  $x_1$  and  $x_{d-1}$  respectively. Since  $G[Q]$  is connected and intersects  $x_i$ ,  $Q \cap \{u, v\} \neq \emptyset$ . Since we assumed that  $Q$  does not contain the vertex  $v$ , we have  $u \in Q$ . In particular,  $Q$  contains the entire subpath of  $P^*$  from  $u$  to  $x_i$ .

Also note that  $\mathbf{u}$  cannot be part of a subset of three vertices  $T'$  which is a triangle as otherwise, we get a pair  $(J^*, T')$  with distance zero contradicting that  $(J^*, T^*)$  was the closest pair  $P^*$  has internal vertices. Hence the forbidden set  $Q$  cannot be a net or a sun whose structure forces  $\mathbf{u}$  to be part of a triangle if it contains  $x_i$ . Since  $x_i$  does not have any paths to the vertex  $\mathbf{u}$  other than the subpath in  $P^*$ , the forbidden set  $Q$  cannot be a hole as well.

Hence  $Q$  can only correspond to a (claw, triangle) forbidden pair. In this case, we claim that the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is also a forbidden set. This contradicts that  $Q$  is a forbidden set as by definition they are required to be minimal.

Since  $Q$  is a forbidden set, it contains vertex subsets that are isomorphic to a claw and a triangle. From Lemma 8, we can conclude that none of the vertices  $x_1, \dots, x_i$  can be part of any triangle in  $G$ . None of these vertices can be part of a claw in  $G$  as well since it contradicts that  $(J^*, T^*)$  is the closest forbidden pair. Since  $v$  disconnects the path  $P^*$ , the subpath  $x_1, \dots, x_i$  is not part of a path connecting a claw and a triangle either as if so  $Q$  must contain the entire path  $P^*$  including  $v$ . Hence the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is still a forbidden set contradicting that  $Q$  is minimal.

These cases of  $Q$  are mutually exhaustive completing the proof of the Lemma.  $\square$

We have  $f(k) = \max\{6^k, 3.619^k\} = 6^k$  and  $c = 7$ . Also we know that PROPER INTERVAL VERTEX DELETION has an 6-approximation algorithm [27] and FEEDBACK VERTEX SET has a 2-approximation algorithm [3]. Hence we have  $d = 7$  as well. We have the following theorem.

**Theorem 12.** PROPER INTERVAL-OR-TREE DELETION *can be solved in  $7^k \text{poly}(n)$ -time and has a 7-approximation algorithm.*

### 5.3 Chordal or Bipartite Permutation

We define the problem as follows.

CHORDAL-OR-BIPARTITE PERMUTATION DELETION

**Input:** An undirected graph  $G = (V, E)$  and an integer  $k$

**Parameter:**  $k$

**Question:** Is there  $S \subseteq V(G)$  of size at most  $k$  such that every connected component of  $G - S$  is either a chordal graph, or a bipartite permutation graph?

The forbidden set for chordal graphs  $\mathcal{F}_1$  is the set of cycle graphs with a length of at least 4. We have the following characterization for bipartite permutation graphs which defines  $\mathcal{F}_2$ .

**Lemma 18.** [4] *A graph is said to be a bipartite permutation graph if and only if it does not contain long-claw,  $X_2, X_3, C_3$  or cycle graphs of length at least 5 as its induced subgraphs. See Figure 2 for an illustration of the graphs  $X_2$  and  $X_3$ .*

We now give a characterization for graphs whose each connected component is either a chordal graph or a bipartite permutation graph.

**Lemma 19.** *The following statements are equivalent.*

1. Let  $G$  be a graph such that every connected component is either chordal or a bipartite permutation graph.
2.  $G$  does not have any  $X_2, X_3$  or induced cycle of length at least 5 as induced subgraphs. Moreover,  $G$  cannot have long-claw and  $C_4$  in the same connected component or have  $C_4$  and triangle in the same connected component.

*Proof.* We prove that the forbidden pair family is  $(C_4, \text{long-claw})$  and  $(C_4, C_3)$ . The forbidden family  $\mathcal{F}_1$  for chordal graphs are holes. The forbidden family of bipartite permutation graphs  $\mathcal{F}_2$  is long-claw,  $X_2, X_3, C_3$  plus cycles of length at least 5. Since cycles of length at least 5 are common in  $\mathcal{F}_1$  and  $\mathcal{F}_2$ , they are in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$ . Since the graphs  $X_2, X_3$  have  $C_4$  as induced subgraph, they are in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  as well. The only remaining pairs in  $\mathcal{F}_1 \times \mathcal{F}_2$  are  $(C_4, \text{long-claw})$  and  $(C_4, \text{triangle})$ . The proof now follows from the forbidden characterization in Lemma 5.  $\square$

Hence Condition 2 is satisfied by CHORDAL-OR-BIPARTITE PERMUTATION DELETION as the forbidden pair is of size two which are  $(\text{long-claw}, C_4)$  and  $(\text{triangle}, C_4)$ . Since CHORDAL VERTEX DELETION is FPT with a  $k^{O(k)}\text{poly}(n)$  running time algorithm from [10] and BIPARTITE PERMUTATION VERTEX DELETION SET is FPT with a  $9^k\text{poly}(n)$  running time algorithm from [4], Condition 1 is satisfied as well.

It remains to show that Condition 3 is satisfied by CHORDAL-OR-BIPARTITE PERMUTATION DELETION. We define  $\mathcal{G}_1$  as all the forbidden graphs in  $\text{sp}(\mathcal{F}_1, \mathcal{F}_2)$  of size at most 10.

Let  $(J^*, T^*)$  be the vertex subsets of a closest forbidden pair in a connected component of a  $\mathcal{G}_1$ -free  $G$ , where  $J^*$  is one of long-claw or triangle and  $T^*$  is a  $C_4$ . Let  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  be a shortest path between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ .

Let  $C$  be the connected component of  $G - (J^* \cup T^*)$  containing the internal vertices of  $P^*$ . We have the following lemma similar to Lemma 13 in INTERVAL-OR-TREE DELETION.

**Lemma 20.** *The graph  $C$  is a caterpillar with the central path being  $P^* \setminus \{u, v\}$ . Furthermore, the only vertices of  $C$  adjacent to  $J^* \cup T^*$  are  $x_1$  and  $x_{d-1}$  which are only adjacent to  $x_0$  and  $x_d$  respectively.*

*Proof.* Let  $w$  be a vertex of  $C$  other than  $P^*$  adjacent to a vertex  $x_i$  with  $i \in [d-1]$ . We claim that  $w$  is not adjacent to any other vertex in  $G$ .

We go over possibilities of edges from  $w$  to other vertices.

**Case 1:** Suppose  $w$  is adjacent to some vertex  $x_j$  with  $j \in \{0, \dots, d\}$ . If  $j = i+1$  or  $j = i-1$ , then  $wx_ix_j$  forms a triangle  $J'$ . Since the path  $P' = x_i, \dots, x_{d-1}, v$  that has length smaller than  $P^*$  connects  $J'$  and  $T^*$ , we have that  $d_G(J', T^*) < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a closest forbidden pair.

Hence  $w$  is not adjacent to  $x_{i-1}$  and  $x_{i+1}$ . Suppose  $j = i+2$  or  $j = i-2$ . Then the graph induced by the vertices  $\{w, x_i, x_{i+1}, x_{i+2}\}$  or  $\{w, x_i, x_{i-1}, x_{i-2}\}$  forms the graph  $T'$  which is a  $C_4$ . Since the path  $P' = u, \dots, x_i$  that has length smaller than  $P^*$  connects  $J^*$  and  $T'$ , we have that  $d_G(J^*, T') < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a closest forbidden pair.

Suppose now  $w$  is adjacent to  $x_j$  with  $0 \leq j < i-2$  or  $i+2 < j \leq d$ . Then note that we have a path from  $x_j$  to  $x_i$  of length two via  $w$  which is shorter than the path

$x_i \dots x_j$  via  $P^*$ . This creates a path  $P'$  which is one of  $u, x_1, \dots, x_j, w, x_i, \dots, x_d, v$  or  $u, x_1, \dots, x_i, w, x_j, \dots, x_d, v$  from  $J^*$  to  $T^*$  of length smaller than  $P^*$  contradicting that  $P^*$  is the shortest such path. Hence  $w$  is not adjacent to any of the vertices in  $P^*$ .

**Case 2:** Suppose  $w$  adjacent to a vertex  $u' \in J^*$ . We assume without loss of generality that among all neighbors of  $w$  in  $J^*$ ,  $u'$  is the vertex that is closest to  $u$  in  $J^*$ , i.e.,  $d_{J^*}(u, u')$  is minimum. We have  $d_{J^*}(u, u') \leq 4$  as  $J^*$  is either a long-claw or a triangle. If  $3 \leq i \leq d-1$ , we have a path from  $u'$  to  $x_i$  of length 2 via  $w$  which is shorter than the path from  $u$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $i = 1$  or  $i = 2$ . Let  $P'$  denote the path between  $u$  and  $u'$  in  $J^*$  and  $P''$  be the subpath of  $P^*$  from  $u$  to  $x_i$ . Then  $u, P''x_iwu'P'u$  forms a cycle  $C_j$  with  $4 \leq j \leq 8$  without any chords. This either contradicts that  $(J^*, T^*)$  is a closest forbidden pair or  $G$  is  $\mathcal{G}_1$ -free. Hence  $w$  is not adjacent to any of the vertices in  $J^*$ .

**Case 3:** Suppose  $w$  adjacent to a vertex  $v' \in T^*$ . We have  $d_{T^*}(v, v') \leq 2$  as  $T^*$  is a  $C_4$ . If  $1 \leq i \leq d-3$ , we have a path from  $v'$  to  $x_i$  of length 2 via  $w$  which is shorter than the path from  $v$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $d-2 \leq i \leq d-1$ . Let  $P'$  be the subpath of  $P^*$  from  $x_i$  to  $v$ . Then  $v'wx_iP'vv'$  forms a cycle  $C_j$  with  $4 \leq j \leq 6$  without any chords. This either contradicts that  $(J^*, T^*)$  is a closest forbidden pair or  $G$  is  $\mathcal{G}_1$ -free.

Hence  $w$  is adjacent to none of the other vertices of  $J^* \cup T^* \cup P^*$ .

**Case 4:** We now prove that  $w$  is not adjacent to any other vertex  $w' \in V(G) - (J^* \cup T^* \cup P^*)$ . Suppose that there exists such a vertex  $w'$ . Suppose  $w'$  is adjacent to vertex  $x_i$ . Then the graph induced by the set of vertices  $J' = \{w', w, x_i\}$  forms a triangle. Since the path  $P' = x_i, \dots, x_{d-1}, v$  that has length smaller than  $P^*$  connects  $J'$  and  $T^*$ , we have that  $d_G(J', T^*) < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a closest forbidden pair.

- **Case 4.1:** Suppose  $w'$  is adjacent to  $x_{i+1}$  or  $x_{i-1}$ . Then the graph  $T'$  induced by the vertices  $\{w', w, x_i, x_{i+1}\}$  or  $\{w', w, x_i, x_{i-1}\}$  forms a  $C_4$ . Since the path  $P' = u, \dots, x_i$  that has length smaller than  $P^*$  connects  $J^*$  and  $T'$ , we have that  $d_G(J^*, T') < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a closest forbidden pair.

Hence this is not the case. Now suppose  $w'$  is adjacent to  $x_{i+2}$  or  $x_{i-2}$ . Then the graph induced by the set of vertices  $\{w', w, x_i, x_{i+1}, x_{i+2}\}$  or  $\{w', w, x_i, x_{i-1}, x_{i-2}\}$  is  $C_5$  contradicting that  $G$  is  $\mathcal{G}_1$ -free. Hence this is not the case. Now suppose  $w'$  is adjacent to  $x_{i+3}$  or  $x_{i-3}$ . Then the graph induced by the set of vertices  $\{w', w, x_i, x_{i+1}, x_{i+2}, x_{i+3}\}$  or  $\{w', w, x_i, x_{i-1}, x_{i-2}, x_{i-3}\}$  is  $C_6$  contradicting that  $G$  is  $\mathcal{G}_1$ -free.

Now suppose  $w'$  is adjacent to  $x_j$  with  $1 \leq j < i+3$  or  $i+3 < j \leq d$ . Then we have a path  $P'$  which is either  $u, x_1 \dots x_j w, x_i \dots x_d, v$  or  $u, x_1 \dots x_i w, x_j \dots x_d, v$  from  $J^*$  to  $T^*$  of length smaller than  $P^*$  contradicting that  $P^*$  is the smallest such path. Hence  $w'$  is not adjacent to any of the vertices in  $P^*$ .

- **Case 4.2:** Suppose  $w'$  adjacent to a vertex  $u' \in J^*$ . We assume without loss of generality that among all neighbors of  $w$  in  $J^*$ ,  $u'$  is the vertex that is closest to

$u$  in  $J^*$ , i.e.  $d_{J^*}(u, u')$  is minimum. If  $3 \leq i \leq d-1$ , we have the path  $u'w'wx_i$  from  $u'$  to  $x_i$  of length 3 via  $w$  which is shorter than the path from  $u$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $1 \leq i \leq 3$ . Let  $P'$  denote the between  $u$  and  $u'$  in  $J^*$  and  $P''$  be the prefix of the path  $P^*$  from  $u$  to  $x_i$ . Then  $uP''x_i, w, w', u'P'u$  forms a cycle  $C_j$  with  $4 \leq j \leq 10$  with no chords. This either contradicts that  $(J^*, T^*)$  is a pair that is closest or  $G$  is  $\mathcal{G}_1$ -free. Hence  $w$  is not adjacent to any of the vertices in  $J^*$ .

- **Case 4.3:** Suppose  $w'$  adjacent to a vertex  $v' \in T^*$ . We have  $d_{T^*}(v, v') \leq 2$  as  $T^*$  is a  $C_4$ . If  $1 \leq i \leq d-4$ , we have the path  $v', w', w, x_i$  from  $v'$  to  $x_i$  of length three which is shorter than the path from  $v$  to  $x_i$  via  $P^*$ . This contradicts that  $P^*$  is the shortest path from  $J^*$  to  $T^*$ . Hence  $d-3 \leq i \leq d-1$ . Let  $P'$  be the suffix of the path  $P^*$  from  $x_i$  to  $v$ . Then  $v', w', w, x_iP'v, v'$  forms a cycle  $C_j$  with  $4 \leq j \leq 7$ . This either contradicts that  $(J^*, T^*)$  is a pair that is closest or  $G$  is  $\mathcal{G}_1$ -free.

Hence we conclude that  $w'$  is not adjacent to any of the vertices in  $J^* \cup T^* \cup P^*$ . Now, we look at the induced subgraph formed by the set of vertices  $J'$  which is

- $\{w', w, x_i, x_{i-1}, x_{i-2}, x_{i+1}, x_{i+2}\}$  for  $3 \leq i \leq x_{d-2}$ ,
- $\{w', w, x_i, x_{i-1}, u, x_{i+1}, x_{i+2}\}$  when  $i = 2$ ,
- $\{w', w, x_i, u, u', x_{i+1}, x_{i+2}\}$  when  $i = 1$  for  $u \in J^* \cap N(u)$  or
- $\{w', w, x_i, x_{i-1}, x_{i-2}, v, v'\}$  when  $i = d-1$  for  $v' \in J^* \cap N(v)$ .

This graph forms a long-claw. Since the path  $P'$  from  $J'$  to  $v \in T^*$  has length smaller than  $P^*$ , the distance  $d_G(J', T^*) < d_G(J^*, T^*)$ . This contradicts the fact that  $(J^*, T^*)$  is a closest forbidden pair.

Hence no such vertex  $w'$  exist and therefore  $w$  has no other neighbors in  $G$ .  $\square$

We now use Lemma 20 to prove that Condition 3 is satisfied for CHORDAL-OR-BIPARTITE PERMUTATION DELETION.

**Lemma 21.** *Condition 3 is satisfied for CHORDAL-OR-BIPARTITE PERMUTATION DELETION.*

*Proof.* Condition 3 is not satisfied in the following case. There exist a pair  $(J^*, T^*)$  which is the vertex subsets of a closest long-claw, triangle pair in a connected component of  $G$ , where  $J^*$  one of long-claw or a triangle and  $T^*$  is a  $C_4$ . Also there is a shortest path  $P^* := x_0, x_1, \dots, x_{d-1}, x_d$  between  $J^*$  and  $T^*$  of length  $d_G(J^*, T^*) = d$  with  $x_0 = u \in J^*$  and  $x_d = v \in T^*$ . A forbidden set  $Q$  of the graph  $G$  is such that  $Q$  contains some internal vertex  $x_i$  of the path  $P$  but it does not contain  $v$ .

Since the graph  $G$  is  $\mathcal{G}_1$ -free,  $G[Q]$  can be a hole of size at least 11 or contain a forbidden pair which is (long-claw,  $C_4$ ) or  $(C_3, C_4)$ . Note that all of these possibilities contain cycles. But from Lemma 20, the component  $C$  of  $G \setminus (J^* \cup T^*)$  that contains the internal vertices of  $P^*$  is a caterpillar which does not contain any cycles. Hence  $Q$  is not fully contained in  $C$ .

From Lemma 20, the only neighbors of  $C$  is  $u$  and  $v$  via  $x_1$  and  $x_{d-1}$  respectively. Since  $G[Q]$  is connected and intersects  $x_i$ ,  $Q \cap \{u, v\} \neq \emptyset$ . Since we assumed that  $Q$

does not contain the vertex  $v$ , we have  $u \in Q$ . In particular,  $Q$  contains the entire subpath of  $P^*$  from  $u$  to  $x_i$ .

Since  $x_i$  does not have any paths to the vertex  $u$  other than the subpath in  $P^*$ , the forbidden set  $Q$  cannot be a hole. Hence  $Q$  can only correspond to a (long-claw,  $C_4$ ) or  $(C_3, C_4)$  forbidden pair. In this case, we claim that the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is also a forbidden set. This contradicts that  $Q$  is a forbidden set as by definition they are required to be minimal.

From Lemma 13, we can conclude that none of the vertices  $x_1, \dots, x_i$  can belong to a subset of vertices in the graph such that the graph induced by the subset is a long-claw, triangle or a  $C_4$ . This is because otherwise, we get a forbidden pair using this subset which is closer than the closest forbidden pair  $(J^*, T^*)$ . Since  $v$  disconnects the path  $P^*$ , the subpath  $x_1, \dots, x_i$  is not part of a path connecting a forbidden pair either as if so  $Q$  must contain the entire path  $P^*$  including  $v$ . Hence the set after removing the vertices  $x_1, \dots, x_i$  from  $Q$  is still a forbidden set contradicting that  $Q$  is minimal.

These cases of  $Q$  are mutually exhaustive completing the proof of the lemma.  $\square$

We have  $f(k) = \max\{k^{O(k)}, 9^k\} = k^{O(k)}$  and  $c = 11$ . We know that CHORDAL VERTEX DELETION has an  $\log^2(\text{OPT})$ -approximation algorithm [1] where  $\text{OPT}$  denote the size of the optimal solution. Also, BIPARTITE PERMUTATION VERTEX DELETION has a 9-approximation algorithm [4]. Hence  $d = \max\{c, c_1, c_2\} = \max\{11, \log^2(\text{OPT}), 2\} = \log^2(\text{OPT})$ . We have the following theorem.

**Theorem 13.** CHORDAL-OR-BIPARTITE PERMUTATION DELETION *can be solved in  $k^{O(k)}$  poly(n)-time and has as a  $\log^2(\text{OPT})$ -approximation algorithm..*

## 6 Conclusion

We gave faster algorithms for some vertex deletion problems to pairs of scattered graph classes with infinite forbidden families. The existence of a polynomial kernel for all the problems studied except the case when both the forbidden families are finite and one of them has a path are open. It is even open when the two scattered graph classes have finite forbidden families (without the forbidden path assumption).

Currently we do not know any  $W[1]$ -hardness results (where we do not expect to have FPT algorithms) for scattered graph classes in the literature when the deletion to each underlying graph classes is FPT. Finding such a result would be interesting.

Another open problem is to give faster FPT algorithms for problems that doesn't fit in any of the frameworks described above, especially problems which does not have a constant sized forbidden pair family. An example is the case when  $(\Pi_1, \Pi_2)$  is (Chordal, Bipartite). The forbidden pair family for this problem is the set of all pairs  $(C_{2i}, C_3)$  with  $i \geq 2$  which is not of constant size.

## References

- [1] Akanksha Agrawal, Daniel Lokshtanov, Pranabendu Misra, Saket Saurabh, and Meirav Zehavi. Polylogarithmic approximation algorithms for weighted-f-deletion problems. *ACM Transactions on Algorithms (TALG)*, 16(4):1–38, 2020.

- [2] Yuuki Aoike, Tatsuya Gima, Tesshu Hanaka, Masashi Kiyomi, Yasuaki Kobayashi, Yusuke Kobayashi, Kazuhiro Kurita, and Yota Otachi. An improved deterministic parameterized algorithm for cactus vertex deletion. *Theory of Computing Systems*, 66(2):502–515, 2022.
- [3] Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.
- [4] Łukasz Bożyk, Jan Derbisz, Tomasz Krawczyk, Jana Novotná, and Karolina Okrasa. Vertex deletion into bipartite permutation graphs. In *15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [5] Andreas Brandstadt, Jeremy P Spinrad, et al. *Graph classes: a survey*, volume 3. SIAM, 1999.
- [6] Sharon Bruckner, Falk Hüffner, and Christian Komusiewicz. A graph modification approach for finding core–periphery structures in protein interaction networks. *Algorithms for Molecular Biology*, 10(1):1–13, 2015.
- [7] Leizhen Cai. Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. *Inf. Process. Lett.*, 58(4):171–176, 1996.
- [8] Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1096–1115. SIAM, 2016.
- [9] Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms (TALG)*, 11(3):1–35, 2015.
- [10] Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016.
- [11] Jianer Chen, Iyad A Kanj, and Ge Xia. Improved upper bounds for vertex cover. *Theoretical Computer Science*, 411(40-42):3736–3756, 2010.
- [12] Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 3. Springer, 2015.
- [13] Marek Cygan and Marcin Pilipczuk. Split vertex deletion meets vertex cover: new fixed-parameter and exact exponential-time algorithms. *Information Processing Letters*, 113(5-6):179–182, 2013.
- [14] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
- [15] Fedor V Fomin, Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. Planar  $f$ -deletion: Approximation, kernelization and optimal fpt algorithms. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 470–479. IEEE, 2012.

- [16] Robert Ganian, M. S. Ramanujan, and Stefan Szeider. Discovering archipelagos of tractability for constraint satisfaction and counting. *ACM Trans. Algorithms*, 13(2):29:1–29:32, 2017.
- [17] Naveen Garg, Vijay V Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.
- [18] Ashwin Jacob, Jari JH de Kroon, Diptapriyo Majumdar, and Venkatesh Raman. Deletion to scattered graph classes i - case of finite number of graph classes. *arXiv preprint arXiv:2105.04660*, 2021.
- [19] Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Parameterized complexity of deletion to scattered graph classes. In *15th International Symposium on Parameterized and Exact Computation (IPEC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.
- [20] Ashwin Jacob, Diptapriyo Majumdar, and Venkatesh Raman. Faster FPT algorithms for deletion to pairs of graph classes. In Evripidis Bampis and Aris Pagourtzis, editors, *Fundamentals of Computation Theory - 23rd International Symposium, FCT 2021, Athens, Greece, September 12-15, 2021, Proceedings*, volume 12867 of *Lecture Notes in Computer Science*, pages 314–326. Springer, 2021.
- [21] Bart MP Jansen, Daniel Lokshtanov, and Saket Saurabh. A near-optimal planarization algorithm. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1802–1811. SIAM, 2014.
- [22] Ken-ichi Kawarabayashi and Anastasios Sidiropoulos. Polylogarithmic approximation for minimum planarization (almost). In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 779–788. IEEE, 2017.
- [23] Tomasz Kociumaka and Marcin Pilipczuk. Faster deterministic Feedback Vertex Set. *Inf. Process. Lett.*, 114(10):556–560, 2014.
- [24] C Lekkekerker and J Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962.
- [25] 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.
- [26] Daniel Lokshtanov, NS Narayanaswamy, Venkatesh Raman, MS Ramanujan, and Saket Saurabh. Faster parameterized algorithms using linear programming. *ACM Transactions on Algorithms (TALG)*, 11(2):1–31, 2014.
- [27] Pim Van’t Hof and Yngve Villanger. Proper interval vertex deletion. *Algorithmica*, 65(4):845–867, 2013.