# Faster FPT Algorithms for Deletion to Pairs of Graph Classes

Ashwin Jacob[1], Diptapriyo Majumdar[2], and Venkatesh Raman[1]

[1] The Institute of Mathematical Sciences, HBNI, Chennai, India
{ajacob,vraman}@imsc.res.in
[2] Royal Holloway, University of London, Egham, United Kingdom
diptapriyo.majumdar@rhul.ac.uk

**Abstract.** Let $\Pi$ be a hereditary graph class. The problem of deletion to $\Pi$, takes as input a graph $G$ and asks for a minimum number (or a fixed integer $k$) of vertices to be deleted from $G$ so that the resulting graph belongs to $\Pi$. This is a well-studied problem in paradigms including approximation and parameterized complexity. This problem, for example, generalizes VERTEX COVER, FEEDBACK VERTEX SET, CLUSTER VERTEX DELETION, PERFECT DELETION to name a few. The study of this problem in parameterized complexity has resulted in several powerful algorithmic techniques including iterative compression and important separators.

Recently, the study of a natural extension of the problem was initiated where we are given a finite set of hereditary graph classes, and the goal is to 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 has been shown to be FPT as long as 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 can be expressible in the counting monodic second order (CMSO) logic. While this was shown using some black box theorems, faster algorithms were shown when each of the hereditary graph classes has a finite forbidden set.

In this paper, we do a deep dive 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. We design two general algorithms for pairs of graph classes (possibly having infinite forbidden sets) satisfying certain conditions on their forbidden sets. These algorithms cover a number of pairs of popular graph classes. Our algorithms make non-trivial use of the branching technique and as black box, FPT algorithms for deletion to individual graph classes.

## 1 Introduction

Graph modification problems are the 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 specific number of "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 [8] 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 class[3] 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 [1] 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. [4], building on the work of Ganian et al. [3] 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 $(\Pi_1, \ldots, \Pi_d)$ consists of graphs whose connected components are in one of the graph classes $\Pi_1, \ldots, \Pi_d$. 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. For example, the solution could possibly be disjoint from the forbidden subgraphs present, as long as it separates them so that the forbidden subgraphs of the $d$ classes don't belong to the same component.

Jacob et al. [4] proved that the vertex deletion problem for the scattered graph class $(\Pi_1, \ldots, \Pi_d)$ is FPT 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 logic. Unfortunately the running of the algorithm incurs a gargantuan constant factor (a function of $k$) overhead. The authors also proved that if the forbidden families corresponding to all the graph classes $\Pi_1, \ldots, \Pi_d$ are finite, then the problem is FPT with running time $2^{poly(k)}n^{\mathcal{O}(1)}$. The technique involves iterative compression and important separators.

Since the algorithms in [4] 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.

[3] A hereditary graph class is a class of graphs that is closed under induced subgraphs

| $\Pi_1$ OR $\Pi_2$ DELETION | **Parameter:** $k$ |
|---|---|
| **Input:** An undirected graph $G = (V, E)$, two hereditary graph classes $\Pi_1$ and $\Pi_2$ defined by forbidden graphs $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively. | |
| **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 $\Pi_2$? | |

The authors in [5] studied one such specific case where $\Pi_1$ is the class of forests (having infinite forbidden set of all cycles) and $\Pi_2$ is the class of cluster graphs, to which they gave an $\mathcal{O}^*(4^k)$[4] algorithm. Here, we describe two general algorithms covering pairs of a variety of graph classes. While the specific conditions, on the pairs of classes to be satisfied by these algorithms, are somewhat technical and are explained in the appropriate sections, we give a high level description here.

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 $J_2$. We know that the solution has to hit the set $J_1 \cup J_2 \cup P$.

The first 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 2 in Section 2) and show that there are pairs of graph classes that have finite number of forbidden pairs even if each of them has infinite forbidden sets. For some of them, we are able to bound the branching step to obtain the FPT algorithm. This problem variant covers a number of pairs of graph classes including (Interval, Trees) and (Chordal, Bipartite Permutation). We observe that this class of algorithms also yield good approximation algorithms for the deletion problem.

In the second general algorithm, we assume that $\mathcal{F}_1$ is finite and has a path $P_i$ for a constant $i$, and the family of graphs of $\mathcal{F}_2$ that is present in the graph class $\Pi_1$ obtained after a pruning step (details in Section 4) is finite. Note that this restricts the length of $P$ to $i$. Under these assumptions, we come up with a finite family of graphs to branch on and we complete the algorithm (at the leaf level of the branching algorithms) with the FPT algorithm for the deletion to each of the individual graph classes. The variant satisfying these conditions covers a number of pairs of graph classes including when $\Pi_1$ is the class of all cliques, and $\Pi_2$ is the class of planar or bounded treewidth graphs.

[4] $\mathcal{O}^*$ notation suppresses polynomial factors

The running time of all the algorithms that we come up in this paper are substantially better than those in [4].

## 2 Forbidden Characterization for $\Pi_1$ or $\Pi_2$ Deletion

We use $\Pi_{(1,2)}$ to denote the class of graphs whose connected components are in the graph classes $\Pi_1$ or $\Pi_2$. The following characterization for $\Pi_{(1,2)}$ is easy to see.

**Lemma 1.** *($\star$) 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 $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$.*

We now define the notion of super-pruned family which gives us a minimal family of graphs which are forbidden in the graph class $\Pi_{(1,2)}$.

We say that family of graphs is *minimal* if no element of it is an induced subgraph of some other element of the family.

**Definition 1 (Super-Pruned Family).**
*An element of* super-pruned family $\mathsf{sp}(\mathcal{G}_1, \mathcal{G}_2)$ *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 $\mathsf{sp}(\mathcal{G}_1, \mathcal{G}_2)$ 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 graph is 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 $\mathcal{F}_1 = \{$net, sun, long claw, whipping top, † -AW, ‡-AW$\} \cup \{C_i : i \geq 4\}$ (See Figure 2 in [2]) and $\mathcal{F}_2$ as the set of all cycles. Note that all graphs $C_i$ with $i \geq 4$ are in $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ 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, †-AW or ‡-AW, it contains triangle as an induced subgraph. Hence these graphs are also in the family $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$.

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

**Lemma 2.** *($\star$)*[5] *If a graph $G$ is in the graph class $\Pi_{(1,2)}$, then no connected component of $G$ contains a graph in $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ as induced subgraphs.*

The family $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ does not cover all the pairs in $\mathcal{F}_1 \times \mathcal{F}_2$. We now define the following to capture the remaining pairs.

**Definition 2 (Forbidden Pair).** *A* forbidden pair *of $\mathcal{F}_1$ and $\mathcal{F}_2$ is a pair $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$ such that both $H_1 \notin \mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ and $H_2 \notin \mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$.*

[5] Proofs of Theorems and Lemmas marked $\star$ are moved to the full version due to lack of space.

For example, if $\Pi_1$ is the class of interval graphs and $\Pi_2$ is the class of forests, we have already shown that $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ contains all the graphs in $\mathcal{F}_1$ except long-claw. The only remaining pair is (long-claw, triangle) which is a forbidden pair.

Now we characterize $\Pi_{(1,2)}$ based on the super-pruned family and a family of forbidden pairs associated to $\mathcal{F}_1$ and $\mathcal{F}_2$. This is used in the algorithms in Section 3.

**Lemma 3.** *(⋆) The following statements are equivalent.*

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

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

**Definition 3.** *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 $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ 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 4.** *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 $J_1, J_2$ corresponding to 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.*

## 3 $\Pi_1$ OR $\Pi_2$ DELETION with a constant number of forbidden pairs

We start with 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 $(G - V(C), k)$.*

In this section we assume that the forbidden pair family (as defined in Section 2) for $\Pi_1$ and $\Pi_2$ is finite. Before we define the further conditions, we give an algorithm for an example pair of graph classes that satisfy our problem conditions.

### 3.1 Interval or Trees

We define the problem.

| Interval-or-Tree Deletion **Parameter:** $k$ |
|---|
| **Input:** An undirected graph $G = (V, E)$ and an integer $k$. |
| **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 4.** *( [7]) 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 in [2] for an illustration of the forbidden subgraphs for interval graph.*

Let $\mathcal{G}_1$ be the family of graphs in $\mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ of size at most 10. We now define the following branching rule where we branch on all induced subgraphs of $G$ isomorphic to a member in $\mathcal{G}_1$. The correctness follows as $\mathcal{G}_1$ is a finite family of finite-sized graphs.

**Branching Rule 1** *Suppose that $(G, k)$ be the input instance and there exist a forbidden set $Q \subseteq V(G)$ such that $G[Q]$ is isomorphic to a member in $\mathcal{G}_1$. Then, for each $v \in V(Q)$, we 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 1 is not applicable for $G$ and so it is $\mathcal{G}_1$-free.

We now focus on connected components of $G$ which contain both long-claw and triangle as induced subgraphs. We describe a branching rule corresponding to the closest forbidden pair.

**Branching Rule 2** *Let $(J^*, T^*)$ be 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. 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 2 is sound. Let $P^* = \{u = x_0, x_1, \ldots, x_{d-1}, x_d = v\}$ be a shortest path of length $d_G(J^*, T^*) = d$ that witnesses a path from $u \in J^*$ to $v \in T^*$.

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

**Lemma 5.** *(⋆) The graph $C$ is a caterpillar with the central path being $P^*$. 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.*

We now use Lemma 5 to prove that Branching Rule 2 is sound.

**Lemma 6.** *(⋆) Branching Rule 2 is sound.*

From here on, assume that $(G, k)$ is an instance for which Reduction Rule 1, Branching Rule 1, and Branching Rule 2 are not applicable. The following results are now easy to see.

**Lemma 7.** *(⋆) Let $C$ be a connected component of $G$ that has no triangle, but has a long-claw as induced subgraph. If $G[C]$ has no feedback vertex set of size $k$, then $(G, k)$ is a no-instance. Otherwise, let $X$ be a minimum feedback vertex set of $G[C]$. Then $(G, k)$ is a yes-instance if and only if $(G - V(C), k - |X|)$ is a yes-instance.*

**Lemma 8.** *(⋆) Let $C$ be a connected component of $G$ that has no long-claw, but has a triangle as induced subgraph. If $G[C]$ has no interval vertex deletion set of size $k$, then $(G, k)$ is a no-instance. Otherwise, let $X$ be a minimum interval 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.*

We are ready to prove our main theorem statement of this section.

**Theorem 1.** *(⋆)* Interval-or-Tree Deletion *can be solved in $\mathcal{O}^*(10^k)$-time.*

We can also give an approximation algorithm for Interval-or-Tree Deletion.

**Theorem 2.** *(⋆)* Interval-or-Tree Deletion *has a 10-approximation algorithm.*

### 3.2 Algorithm for Special Infinite-$(\Pi_1, \Pi_2)$-Deletion

Now, we show that the algorithm idea of the last section is applicable for a larger number of pairs of graph classes by identifying the properties that enabled the algorithm. We now define the variant of $\Pi_1$ or $\Pi_2$ Deletion satisfying the following properties.

1. The vertex deletion problems for the graph classes $\Pi_1$ and $\Pi_2$ are FPT with algorithms to respective classes being $\mathcal{A}_1$ and $\mathcal{A}_2$.
2. The number of forbidden pairs of $\mathcal{F}_1$ and $\mathcal{F}_2$ is a constant.
3. All graphs in $\mathcal{F}_1$ and $\mathcal{F}_2$ are connected.
4. Let $(H_1, H_2)$ be a closest forbidden pair in the graph $G$ with $(J_1, J_2)$ being the vertex subsets corresponding to the pair and $P$ being a shortest path between $J_1$ and $J_2$. There is a subfamily $\mathcal{G}_1 \subseteq \mathsf{sp}(\mathcal{F}_1, \mathcal{F}_2)$ such that
   - $\mathcal{G}_1$ is a finite family of finite-sized (independent of the size of $G$) graphs and
   - in the graph $G$ that is $\mathcal{G}_1$-free, if a forbidden set $Q$ intersects the internal vertices of $P$, then $V(P) \subseteq Q$ where $V(P)$ is the vertex set of the path $P$.

| Special Infinite-$(\Pi_1, \Pi_2)$-Deletion **Parameter:** $k$ |
|---|
| **Input:** An undirected graph $G = (V, E)$, graph classes $\Pi_1, \Pi_2$ with forbidden families $\mathcal{F}_1$ and $\mathcal{F}_2$ such that conditions 1 - 4 are satisfied and an integer $k$. |
| **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 $\Pi_2$? |

Towards an FPT algorithm for Special Infinite-$(\Pi_1, \Pi_2)$-Deletion, We define the following branching rule whose soundness is easy to see.

**Branching Rule 3** *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 3 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. For $i \in \{1, 2\}$, let $\mathcal{F}_p^i$ denote the family of graphs $H_i$ where $(H_1, H_2)$ is a forbidden pair.

We have the following branching rule.

**Branching Rule 4** *Let $(J^*, T^*)$ be the vertex subsets of a closest forbidden pair $(H_1, H_2)$ of $\mathcal{F}_1$ and $\mathcal{F}_2$. Then for each $v \in J^* \cup T^*$, we delete $v$ and decrease $k$ by 1, resulting in the instance $(G - v, k - 1)$.*

The soundness of the above branching rule comes from condition 4 where we assume that if a forbidden set $Q$ intersects the internal vertices of a shortest path $P$ between $J^*$ and $T^*$, then $V(P) \subseteq Q$.

**Lemma 9.** *($\star$) Branching Rule 4 is sound.*

From here on, assume that $(G, k)$ is an instance for which Reduction Rule 1, Branching Rule 3, and Branching Rule 4 are not applicable. Note that any component of $G$ is now free of forbidden pairs. Hence it is $\mathcal{F}_p^1$-free or $\mathcal{F}_p^2$-free. The following results are now easy to see.

**Lemma 10.** *($\star$) Let $C$ be a connected component of $G$ that is $\mathcal{F}_p^1$-free. If $G[C]$ has no $\Pi_1$-deletion vertex set of size $k$, then $(G, k)$ is a no-instance. Otherwise, let $X$ be a minimum $\Pi_1$-deletion vertex set of $G[C]$. Then $(G, k)$ is a yes-instance if and only if $(G - V(C), k - |X|)$ is a yes-instance.*

The proof of the following lemma is similar to that of Lemma 10.

**Lemma 11.** *Let $C$ be a connected component of $G$ that is $\mathcal{F}_p^2$-free. If $G[C]$ has no $\Pi_2$-deletion vertex set of size $k$, then $(G, k)$ is a no-instance. Otherwise, let $X$ be a minimum $\Pi_2$-deletion vertex set of $G[C]$. Then $(G, k)$ is a yes-instance if and only if $(G - V(C), k - |X|)$ is a yes-instance.*

We are ready to prove our main theorem statement of this section. Let $f(k) = \max\{f_1(k), f_2(k)\}$ where $O^*(f_i(k))$ is the running time for the algorithm $\mathcal{A}_i$. Also let $c$ the maximum among the size of graphs in $\mathcal{G}_1$ and $\max_{(H_1,H_2)}(|H_1| + |H_2|)$ where $(H_1, H_2)$ is a forbidden pair.

**Theorem 3.** *(⋆)* Special Infinite-$(\Pi_1, \Pi_2)$-Deletion *can be solved in* $\mathcal{O}^*(\max\{f(k), c^k\})$*-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 $c_i$.

**Theorem 4.** *(⋆)* Special Infinite-$(\Pi_1, \Pi_2)$-Deletion *has a* $d$*-approximation algorithm where* $d = \max(c, c_1, c_2)$.

The problem Special Infinite-$(\Pi_1, \Pi_2)$-Deletion applies for a number of pairs of graphs of $\Pi_1$ or $\Pi_2$ Deletion. We list a few below.

**Corollary 1.** *(⋆)*

- Proper Interval-or-Tree Deletion *can be solved in* $\mathcal{O}^*(7^k)$*-time and has a* 7*-approximation algorithm.*
- Chordal-or-Bipartite Permutation Deletion *can be solved in* $\mathcal{O}^*(k^{O(k)})$*-time and has a* $\log^2(|OPT|)$*-approximation algorithm where* $OPT$ *denotes the optimal solution.*

## 4 $\Pi_1$ or $\Pi_2$ Deletion when $\mathcal{F}_2$ is infinite and $P_i$ is forbidden in $\Pi_1$

Here, we give an algorithm for another variant of $\Pi_1$ or $\Pi_2$ Deletion where $\Pi_1$ has a finite forbidden set with $P_i$ a path of length $i$, for a constant $i$, being one of them. To explain the further conditions necessary for the pair of classes to be satisfied, we define the following notion of sub-pruned family which is similar to the super-pruned family defined before.

**Definition 5 (Sub-Pruned Family).** *We define a family* $\mathcal{F}_p$ *associated with* $\mathcal{F}_1 \times \mathcal{F}_2$ *as follows.*

*A graph* $H$ *is in* $\mathcal{F}_p$ *if there exists a graph* $G$ *such that (a)* $H$ *is an induced subgraph of* $G$ *and* $(H, G)$ *or* $(G, H)$ *is in* $\mathcal{F}_1 \times \mathcal{F}_2$ *and (b) there is no graph* $G'$ *such that* $G'$ *is an induced subgraph of* $H$ *and* $(G', H)$ *or* $(H, G')$ *is in* $\mathcal{F}_1 \times \mathcal{F}_2$.

$\mathcal{F}_p$ *can be obtained from an enumeration of all pairs* $(H_1, H_2) \in \mathcal{F}_1 \times \mathcal{F}_2$*, and for each such pair, adding* $H_1$ *to* $\mathcal{F}_p$ *and removing* $H_2$ *from* $\mathcal{F}_p$ *(if it already exists) whenever* $H_1$ *is an induced subgraph of* $H_2$*, and adding* $H_2$ *to* $\mathcal{F}_p$ *and removing* $H_1$ *from* $\mathcal{F}_p$ *(if it already exists) if* $H_2$ *is an induced subgraph of* $H_1$*. The resulting family* $\mathcal{F}_p$ *is called the* sub-pruned family *of* $\mathcal{F}_1 \times \mathcal{F}_2$ *denoted by* $\mathsf{SubPrune}(\mathcal{F}_1 \times \mathcal{F}_2)$.

**Example** Let ($\Pi_1$,$\Pi_2$) be (Interval, Trees). Note that all graphs $C_i$ with $i \geq 4$ are in $\mathsf{SubPrune}(\mathcal{F}_1 \times \mathcal{F}_2)$ as they occur in both $\mathcal{F}_1$ and $\mathcal{F}_2$. The remaining pairs contain triangles from $\mathcal{F}_2$. If the graph from $\mathcal{F}_1$ is a net, sun, whipping top, †-AW or ‡-AW, it contains triangle as subgraphs. Hence triangle is also added to the family $\mathsf{SubPrune}(\mathcal{F}_1 \times \mathcal{F}_2)$.

A key difference in the definitions of super-pruned family and sub-pruned family is that for the latter, we do not assume that the associated families are minimal. If for some $H \in \mathcal{F}$ for a family $\mathcal{F}$ of graphs, we have that $H$ is an induced subgraph of some other element of $\mathcal{F}$, we call the graph $H$ an *irrelevant* graph. In the following lemma, we prove that applying the sub-pruning operation to $\mathcal{F} \times \mathcal{F}$ helps us to remove the irrelevant graphs from $\mathcal{F}$ if $\mathcal{F}$ is not minimal.

**Lemma 12.** *($\star$) If $\mathcal{F}$ is a forbidden family for $\Pi$, then* $\mathsf{SubPrune}(\mathcal{F} \times \mathcal{F})$ *is a forbidden family for $\Pi$.*

We now define the variant of $\Pi_1$ OR $\Pi_2$ DELETION that we look at in this section. Let $\mathcal{F}' = \Pi_1 \cap \mathcal{F}_2$. We assume that

1. $\mathcal{F}_1$ is finite.
2. $P_i$, for some constant $i$, is forbidden in $\Pi_1$ where $P_i$ is the path on $i$ vertices.
3. there is an FPT algorithm $\mathcal{A}$ for $\Pi_2$-vertex deletion problem and
4. $\mathsf{SubPrune}(\mathcal{F}' \times \mathcal{F}')$ is known to be finite.

| SPECIAL MIXED-$(\Pi_1, \Pi_2)$-DELETION **Parameter:** $k$ |
|---|
| **Input:** An undirected graph $G = (V, E)$, two graph classes $\Pi_1$ and $\Pi_2$ defined by forbidden graphs $\mathcal{F}_1$ and $\mathcal{F}_2$ respectively. Furthermore, conditions 1-4 above are satisfied. |
| **Question:** Is there $S \subseteq V(G)$ of size at most $k$ such that every connected component of $G - S$ is either in $\Pi_1$ or in $\Pi_2$? |

Before we develop the algorithm for this general version of scattered vertex deletion, we explain the algorithm for a specific example pair in the next subsection.

### 4.1 Clique or Planar graphs

| CLIQUE OR PLANAR VERTEX DELETION **Parameter:** $k$ |
|---|
| **Input:** An undirected graph $G = (V, E)$ and an integer $k$. |
| **Question:** Is there $S \subseteq V(G)$ of size at most $k$ such that every connected component of $G - S$ is a clique or a planar graph? |

We first show that the problem is indeed an example of SPECIAL MIXED-$(\Pi_1, \Pi_2)$-DELETION problem. We have $\mathcal{F}_1 = \{2K_1\}$ which is finite. Since $P_3$ is forbidden in cliques, condition 2 is satisfied. The condition 3 is satisfied as there is an FPT algorithm with $\mathcal{O}^*(k^{O(k)})$ running time for PLANAR VERTEX DELETION [6].

Finally we have $\mathcal{F}' = \mathcal{F}_2 \cap \Pi_1 = \{K_5, K_6, \ldots, \}$ as planar graphs are $K_5$-free. We have $\mathsf{SubPrune}(\mathcal{F}' \times \mathcal{F}') = \{K_5\}$ which being finite satisfies condition 4.

Let $\mathcal{F}'_h$ be the family of graphs that contain $P_3$ and $K_5$ as induced subgraphs. We define the family $\mathcal{F}_h = \mathsf{SubPrune}(\mathcal{F}'_h \times \mathcal{F}'_h)$.

**Lemma 13.** *The family $\mathcal{F}_h$ is of finite size with graphs of size at most 8.*

*Proof.* Let $H$ be a graph in $\mathcal{F}_h$. Let $J_1$ and $J_2$ be vertex subsets of $H$ such that $H[J_1]$ is isomorphic to $P_3$ and $H[J_2]$ is isomorphic to $K_5$. First, we observe that $d_H(J_1, J_2) \leq 1$. Suppose not. Then there is a path $P$ between $J_1$ and $J_2$ of length at least 2. Let $J_3$ be the last three vertices of $P$. Note that $G[J_3]$ is isomorphic to $P_3$ as well. Then $H[J_2 \cup J_3] \in \mathcal{F}_h$ is also a graph that contains $P_3$ and $K_5$ as induced subgraphs. This contradicts that $H \in \mathcal{F}_h$ as it will be removed from $\mathcal{F}_h$ for the pair $(H, H[J_2 \cup J_3])$. Furthermore, note that $H[J_1 \cup J_2]$ is a connected graph which has $P_3$ and $K_5$ as induced subgraphs. Hence $H = H[J_1 \cup J_2]$ as otherwise it will be removed from $\mathcal{F}_h$ for the pair $(H, H[J_1 \cup J_2])$. This proves the lemma as $|J_1 \cup J_2| \leq 8$. □

We now describe the algorithm. The following branching rule and its soundness is easy to see.

**Branching Rule 5** *Suppose the graph $G$ of the input instance $(G, k)$ has a graph $\hat{H} \in \mathcal{F}_h$ as its induced subgraph. Then for each $v \in V(\hat{H})$, we delete $v$ and decrease $k$ by 1, resulting in the instance $(G - v, k - 1)$.*

The following lemma is easy to see.

**Lemma 14.** *(⋆) Let $G$ be a graph such that Branching Rule 5 is not applicable. Let $C$ be a component of $G$ which contains $K_5$ as an induced subgraph. Then $C$ is a clique.*

Since Reduction Rule 1 removes components of $G$ that are cliques, we have the following corollary.

**Corollary 2.** *If Reduction Rule 1 and Branching Rule 5 are not applicable, then no connected component of the graph $G$ has $K_5$ as induced subgraph.*

The following lemma allows us to apply the algorithm for Planar Vertex Deletion in the remaining components of $G$ to solve the problem.

**Lemma 15.** *(⋆) Let $(G, k)$ be the resulting instance after applying the reduction rules. If $G$ has no planar vertex deletion set of size $k$, then $(G, k)$ is a no-instance. Otherwise, let $X$ be a minimum sized set such that $G - X$ is planar. Then $X$ is also a solution $(G, k)$.*

The main theorem now follows.

**Theorem 5.** *(⋆)* Clique or Planar Vertex Deletion *has a $\mathcal{O}^*(k^{O(k)})$ time FPT algorithm.*

### 4.2 Algorithm for Special Mixed-$(\Pi_1, \Pi_2)$-Deletion

We now give the algorithm of Special Mixed-$(\Pi_1, \Pi_2)$-Deletion. The ideas here can be seen as generalizations of the algorithm for Clique or Planar Vertex Deletion.

Let $\mathcal{F}'_h$ denote the family of graphs $H$ that satisfies the following.

- there exists a pair $(H_1, H_2) \in \mathcal{F}_1 \times \mathsf{SubPrune}(\mathcal{F}' \times \mathcal{F}')$ such that both $H_1$ and $H_2$ occur as induced subgraphs of $H$.
- Let $Q_1, Q_2 \subseteq V(H)$ such that $H[Q_1]$ is isomorphic to $H_1$ and $H[Q_2]$ is isomorphic to $H_2$. Then $d_H(Q_1, Q_2) \leq i$.

We define $\mathcal{F}_h = \mathsf{SubPrune}(\mathcal{F}'_h \times \mathcal{F}'_h)$.

We give algorithm Special Mixed-$(\Pi_1, \Pi_2)$-Deletion} by branching over graphs in $\mathcal{F}_h$ and later applying algorithm $\mathcal{A}$ for vertex deletion to graph class $\Pi_2$. The details are moved to the full version of the paper.

**Theorem 6.** *($\star$)* Special Mixed-$(\Pi_1, \Pi_2)$-Deletion *has an FPT algorithm with running time* $(d_1 + i + d_2)^k f(k) poly(n)$ *where* $d_1 = \max\{|F| : F \in \mathcal{F}_1\}$, $d_2 = \max\{|H| : H \in \mathcal{F}_h\}$ *and supposing algorithm* $\mathcal{A}$ *takes* $\mathcal{O}^*(f(k))$ *time.*

The problem Special Mixed-$(\Pi_1, \Pi_2)$-Deletion applies for a number of pairs of graphs of $\Pi_1$ or $\Pi_2$ Deletion. We list a few below. Note that the second algorithm in the list covers a number of graph classes for $\Pi_2$. For example $\Pi_2$ can be the class of all planar graphs, or the class of all graphs with treewidth $t$.

**Corollary 3.** *($\star$)* $\Pi_1$ or $\Pi_2$ Deletion *is FPT for the following pairs* $(\Pi_1, \Pi_2)$ *of graph classes.*

1. $\Pi_1$ *is the class of cliques and* $\Pi_2$ *is the class of cactus graphs. The FPT algorithm has running time* $\mathcal{O}^*(26^k)$.
2. $\Pi_1$ *is the class of cliques and* $\Pi_2$ *is the class of graphs that has an FPT algorithm with* $\mathcal{O}^*(f(k))$ *running time for its deletion problem and its forbidden family* $\mathcal{F}_2$ *contains the graph* $K_t$ *for some constant* $t$. *The FPT algorithm has running time* $\mathcal{O}^*((t+1)^k f(k))$.
3. $\Pi_1$ *is the class of split graphs and* $\Pi_2$ *is the class of bipartite graphs. The FPT algorithm has running time* $\mathcal{O}^*(13^k)$.

## 5 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 are open. It is even open when all the scattered graph classes have finite forbidden families.

## References

1. Cai, L.: Fixed-Parameter Tractability of Graph Modification Problems for Hereditary Properties. Inf. Process. Lett. **58**(4), 171–176 (1996)
2. Cao, Y., Marx, D.: Interval deletion is fixed-parameter tractable. ACM Transactions on Algorithms (TALG) **11**(3), 1–35 (2015)
3. Ganian, R., Ramanujan, M.S., Szeider, S.: Discovering archipelagos of tractability for constraint satisfaction and counting. ACM Trans. Algorithms **13**(2), 29:1–29:32 (2017)
4. Jacob, A., de Kroon, J.J., Majumdar, D., Raman, V.: Parameterized complexity of deletion to scattered graph classes. arXiv preprint arXiv:2105.04660 (2021)
5. Jacob, A., Majumdar, D., Raman, V.: 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)
6. Jansen, B.M., Lokshtanov, D., Saurabh, S.: A near-optimal planarization algorithm. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1802–1811. SIAM (2014)
7. Lekkeikerker, C., Boland, J.: Representation of a finite graph by a set of intervals on the real line. Fundamenta Mathematicae **51**(1), 45–64 (1962)
8. Lewis, J.M., Yannakakis, M.: The node-deletion problem for hereditary properties is np-complete. J. Comput. Syst. Sci. **20**(2), 219–230 (1980)