Fixed Parameterized Algorithms for Generalized Feedback Vertex Set Problems[*]

Bin Sheng[1] and Gregory Gutin[2]

[1]College of Computer Engineering and Science, Shanghai University, Shanghai, 200444, PR China shengbin@shu.edu.cn
[2]Royal Holloway, University of London, Egham, United Kingdom g.gutin@rhul.ac.uk

February 28, 2023

Abstract

A graph is an r-pseudoforest if every connected component of it has a feedback edge set of size at most r. A graph is a d-quasi-forest if every connected component of it has a feedback vertex set of size at most d. The r-Pseudoforest Deletion problem (d-Quasi-Forest Deletion problem) asks to delete a minimum number of vertices to get an r-pseudoforest (a d-quasi-forest, respectively). The well-studied feedback vertex set problem is the special case of r-Pseudoforest Deletion (d-Quasi-Forest Deletion, respectively) in which $r = 0$ ($d = 0$, respectively).

We provide an improved FPT algorithm and a smaller kernel for r-Pseudoforest Deletion when parameterized by the solution size (when r is fixed). For d-Quasi-Forest Deletion, we show that it is FPT as well when parameterized by d and the solution size.

1 Introduction

The Feedback Vertex Set problem is one of the 21 NP-hard problems proved by Karp [19]. It asks to delete a minimum number of vertices from a given graph to make it acyclic. The application of Feedback Vertex Set ranges from artificial intelligence [1, 2], bio-computing [3, 15] to operating systems [24] and so on.

[*]The research of Bin Sheng was partially supported by the National Natural Science Foundation of China (No. 61802178). A preliminary version of this paper was accepted at the 16th Annual Conference on Theory and Applications of Models of Computation.

The Feedback Vertex Set problem has attracted a lot of attention from the parameterized complexity community. Both its undirected and directed versions are fixed-parameter tractable when parameterized by the solution size [6, 7, 8, 11]. For the undirected case, the state-of-the-art algorithm runs in time $O^*(3.460^k)$ in the deterministic setting [18], and $O^*(2.7^k)$ in the randomized setting [20]. Here, k is the solution size and the O^* notation hides polynomial factors in n, which is the number of vertices in the graph.

Relaxing the acyclic requirement, researchers have defined several classes of almost acyclic graphs. A graph F is an *r-pseudoforest* if every connected component in F has a feedback edge set of size at most r. A *pseudoforest* is a 1-pseudoforest. An *almost r-forest* is a graph that has a feedback edge set of size at most r. Let $\mathcal{F}$ be a graph class, we use $\mathcal{F}$ Deletion to denote the problem of deleting a minimum number of vertices from a given graph to get a graph in $\mathcal{F}$.

Philip *et al.* [22] introduced the problem of deleting vertices to get an almost acyclic graph, generalizing the feedback vertex set problem. There have been several results in this line of research. In [22], the authors gave an $O^*(c^k)$-time algorithm for r-Pseudoforest Deletion, in which c is a constant that only depends on r. They also gave an $O^*(7.56^k)$-time algorithm for the problem of Pseudoforest Deletion. Bodlaender *et al.* [4] designed an improved algorithm for Pseudoforest Deletion that runs in time $O^*(3^k)$. Rai and Saurabh [23] gave an $O^*(5.0024^{(k+r)})$-time algorithm for Almost r-Forest Deletion. An improved algorithm for this problem that runs in time $O^*(5^k 4^r)$ was obtained by Lin *et al.* [21].

A *d-quasi-forest* is a graph in which every connected component has a feedback vertex set of size at most d. Hols and Kratsch [17] raised this notion and showed that the Vertex Cover problem admits a polynomial kernel when parameterized by distance to d-quasi-forest. However, how to compute the distance of a graph to a d-quasi-forest was not discussed there.

In this paper, we first give an algorithm for r-Pseudoforest Deletion parameterized[1] by k (r is fixed) that runs in time $O^*((1+(2r+3)^{r+2})^{k+1})$. This improves the general-r $O^*(c^k)$-time algorithm of [22] as the constant c (depending on r) of that algorithm is bounded as follows: $c > 2^{240000s(r+2)^2}$, where $s = r'2^{6(r+2)}$ for some positive integer r' (this is discussed in detail at the end of Section 3). We also improve the kernel size for r-Pseudoforest Deletion from $976k(k+r)(3r+8)(r+2)2^{6(r+2)}$ to $O(\max\{k^2r^2, kr^3\})$ (this is discussed in detail at the end of Section 4). We then give an FPT algorithm for d-Quasi-Forest Deletion, parameterized by the solution size and d, that runs in time $O^*(2^{k^7 d^6 3^{k^3 d^2}})$. To the best of our knowledge, this is the first nontrivial FPT result for d-Quasi-Forest Deletion.

[1]We provide a brief introduction to parameterized algorithms and kernelization in the next section.

2 Notation and Terminology

Parameterized complexity. A parameterized problem is a subset $L \subseteq \Sigma^* \times \mathbb{N}$ over a finite alphabet Σ. L is *fixed-parameter tractable* (FPT) if the membership of an instance (x, k) in $\Sigma^* \times \mathbb{N}$ can be decided in time $f(k)|x|^{O(1)}$, where f is a computable function of the parameter k only. A *kernelization* of a parameterized problem L is a polynomial-time algorithm that maps an instance (x, k) to an instance (x', k'), the *kernel*, such that $(x, k) \in L$ if and only if $(x', k') \in L$, $k' \leq g(k)$, and $|x'| \leq g(k)$ for some function g of k only. We call $g(k)$ the size of the kernel. For further background and terminology on parameterized complexity, we refer readers to monographs [10, 12, 13, 14].

Graph Theory. Here, we give a brief list of the graph theory concepts used in this paper; for other notations and terminology, we refer readers to [5].

All graphs considered in this paper are undirected. For a graph $G = (V, E)$, V is its vertex set and E is its edge set. A non-empty graph G is *connected* if there is a path between every pair of vertices. Otherwise, we call it *disconnected*.

The *multiplicity* of an edge is the number of its appearances in the multigraph. An edge uv is a *loop* if $u = v$, and we call it a loop at u. The number of edges incident with a vertex v is called the *degree* of v, where loops are counted twice. We use $d_G(u)$ to denote the degree of u in G. We use $\delta(G)$ to denote the minimum degree of vertices in G. For any vertex subset $X \subseteq V(G)$, $d_X(u)$ is the number of edges between u and X. *Bypassing* a vertex v of degree 2 means to delete v and add an edge between its two neighbors u and w (even if there is already an edge between u and w or $u = w$). *Subdividing* an edge xy replaces it with two edges xu and uy, where $u \notin V(G)$ is a new vertex.

For any vertex set $X \subseteq V(G)$, we use $G - X$ to denote the graph obtained by deleting X and all edges incident with vertices in X from G. If $X = \{u\}$, then we abbreviate $G - X$ as $G - u$. The *neighborhood* of u is $N(u) = \{v | uv \in E(G), v \neq u\}$, and the *neighborhood* of u in X is $N_X(u) = N(u) \cap X$. For a vertex set $U \subset V(G)$, $N(U) = (\cup_{u \in U} N(u)) \setminus U$ and $N_X(U) = N(U) \cap X$, where $X \subseteq V(G)$. For two vertex sets $X, Y \subseteq V(G)$ with $X \cap Y = \emptyset$, $E[X, Y] = \{uv | u \in X, v \in Y\}$. We will not always distinguish strictly between a graph and its vertex or edge set. For example, we may speak of a vertex $v \in G$ rather than $v \in V(G)$, an edge $e \in G$, and so on.

A *forest* is a graph in which there is no cycle, and a *tree* is a connected forest. A vertex subset $D \subseteq V(G)$ is a *feedback vertex set* of G, if $G - D$ is a forest. An edge subset $B \subseteq E(G)$ is a *feedback edge set* of G, if $G - B$ is a forest. We use $\text{fvs}(G)$ to denote the *feedback vertex number* of G, i.e., the size of a smallest feedback vertex set of G.

A subgraph H of G is an *induced subgraph* of G if for every $u, v \in V(G)$, edge $uv \in E(H)$ if and only if $uv \in E(G)$. We denote H by $G[V(H)]$ if H is an induced subgraph of G. For a vertex u and a vertex subset $S \subseteq V(G)$, we say u and S are *adjacent* if $N_S(u) \neq \emptyset$. A *v-flower* in a graph G is a set $\mathcal{C}$ of cycles, such that $V(C_1) \cap V(C_2) = \{v\}$ for any two cycles $C_1, C_2 \in \mathcal{C}$. Let $R \subseteq V(G)$ be a set of vertices, then an *R-path* is a path of positive length between two vertices in R such that all other vertices are not in R. For a positive integer n, $[n] = \{1, 2, \ldots, n-1, n\}$.

3 Branching algorithm for r-Pseudoforest Deletion

Definition 1. *Given a graph $G = (V, E)$, an **r-pseudoforest deletion set** of G is a subset $S \subseteq V(G)$ such that $G - S$ is an r-pseudoforest.*

Here is the formal definition of the parameterized r-Pseudoforest Deletion problem.

r-Pseudoforest Deletion
Instance: Graph G, integers k and r.
Parameter: k and r.
Output: Decide if there exists an r-pseudoforest deletion set X of G with $|X| \leq k$?

For a connected component C in a graph G, we define the *excess* of C, as $ex(C) = |E(C)| - |V(C)| + 1$. Note that $ex(C) \geq 0$ for every connected component. Let $\mathcal{C}$ be the set of connected components in G. Following [22], we define the excess of G, as $ex(G) = \max_{C \in \mathcal{C}} ex(C)$, i.e., the greatest excess among all connected components in G. Note that to compute the excess of a connected graph G, we just need to compute the value of $|V(G)|$ and $|E(G)|$, which can be done in time $\Theta(m+n)$.

The following lemma shows that excess is a good notion to characterize r-pseudoforests.

Lemma 1. *A graph G is an r-pseudoforest if and only if $ex(G) \leq r$.*

Proof. By definition, a connected graph F is an r-pseudoforest if and only we can delete at most r edges from it to get a forest. If we can delete at most r edges from F to get a forest, then we can delete at most r edges from F to get a tree. Moreover, if T is a tree, then $|E(T)| = |V(T)| - 1$. Thus, a connected graph F is an r-pseudoforest if and only if $|E(F)| \leq |V(F)| - 1 + r$, i.e. $ex(F) \leq r$.

If G is an r-pseudoforest, then each connected component in G can be made into a forest by deleting at most r edges. Let $\mathcal{C}$ be the set of connected components in G. Then each connected component in $C \in \mathcal{C}$ has excess at most r. Therefore, $ex(G) = \max_{C \in \mathcal{C}} ex(C) \leq r$.

If $ex(G) \leq r$, then every connected component C in G has excess at most r, and so we can delete r edges from it to get a tree. Thus, if $ex(G) \leq r$, then G is an r-pseudoforest. $\square$

By Lemma 1, to decide whether a graph G is an r-pseudoforest, we just need to compute the excess of G, which can be done in time $\Theta(m+n)$.

For a vertex set $S \subseteq V(G)$, let $cc(S) = cc(G[S])$ be the number of connected components in $G[S]$.

Lemma 2. *[22] Let G' be a subgraph of G. If G is an r-pseudoforest, then so is G'.*

Lemma 3. *Let G be a graph that contains a vertex u with degree 2. Let G' be the graph obtained from G by bypassing u. Then for any positive integer r, G' is an r-pseudoforest if and only if G is an r-pseudoforest.*

Proof. Let C be the connected component in G that contains u, and C' be the component we get from C by bypassing u. Then $|V(C')| = |V(C)| - 1$, $|E(C')| = |E(C)| - 1$. Thus, $ex(C') \leq r$ if and only if $ex(C) \leq r$. As G and G' only differ at C and C', G' is an r-pseudoforest if and only if G is an r-pseudoforest. $\square$

Lemma 4. *Let G be a graph that contains a vertex u with $d(u) \leq 1$. If $G-u$ is an r-pseudoforest, then G is also an r-pseudoforest.*

Proof. If $G - u$ is an r-pseudoforest, then $ex(G-u) \leq r$ by definition.

If $d(u) = 0$, then u is an isolated vertex in G, and $ex(G[u]) < r$. Therefore, $ex(G) = \max\{ex(G-u), ex(G[u])\} \leq r$ and G is also an r-pseudoforest.

If $d(u) = 1$, then u is a leaf in some connected component C of G. Observe that $ex(C) = ex(C-u)$. Moreover, $ex(C-u) \leq ex(G-u) \leq r$, as $C - u$ is a connected component in $G - u$. It follows that $ex(G) = \max\{ex(G-C), ex(C)\} \leq r$, and so G is also an r-pseudoforest. $\square$

Now we solve r-Pseudoforest Deletion via the approach of *iterative compression*. The iterative compression method is a recursive approach typically for parameterized minimization problems. It exploits the instance structure exposed by a solution that is slightly oversized. Firstly, obtain a slightly oversized solution. Secondly, transform the problem into its disjoint version by guessing the intersection of the optimal solution with the oversized solution. Finally, solve the disjoint version of the problem, with additional information introduced by the forbidden set. We refer readers to [10] for a more thorough introduction to the iterative compression method.

As a standard step, we introduce the following disjoint version of r-Pseudoforest Deletion.

Disjoint r-Pseudoforest Deletion
Input: Graph G with an r-pseudoforest deletion set S, two integers k and r.
Parameter: k.
Output: Decide if there is an r-pseudoforest deletion set X of G with $|X| \leq k$ and $X \cap S = \emptyset$?

We may assume that $G[S]$ is an r-pseudoforest, as otherwise (G, S, k, r) is a no-instance of Disjoint r-Pseudoforest Deletion. For any given graph G, fix an arbitrary vertex ordering $v_1, v_2, \ldots, v_n$ of G. We solve r-Pseudoforest Deletion for the following series of graphs, $G_1, G_2, \ldots, G_n$, where $G_i = G[\{v_1, v_2, \ldots, v_i\}]$. Note that if G_i has no r-pseudoforest deletion set of size at most k, then neither has G_{i+1}, according to Lemma 2. If X is a solution to r-Pseudoforest Deletion of (G_i, k, r), then $X \cup \{v_{i+1}\}$ is solution to $(G_{i+1}, k+1, r)$. Thus, each time we solve the Disjoint r-Pseudoforest Deletion problem for (G_i, S, k, r), we can conclude that $|S| \leq k+1$.

To solve Disjoint r-Pseudoforest Deletion, we apply the following reduction rules. A reduction rule is *safe* if the reduced instance is a yes-instance if and only if the original instance is a yes-instance.

Reduction Rule 1: Let (G, S, k, r) be an instance of Disjoint r-Pseudoforest Deletion. If there is a vertex $v \in (V(G) \setminus S)$ such that $d_G(v) = 1$, then return $(G - v, S, k, r)$.

Lemma 5. *Reduction Rule 1 is safe.*

Proof. We show (G, S, k, r) and $(G - v, S, k, r)$ are equivalent instances of Disjoint r-Pseudoforest Deletion.

First, suppose (G, S, k, r) is a yes-instance and $X \subseteq (V(G) \setminus S)$ is a solution. It follows that $G - X$ is an r-pseudoforest. If $v \in X$, then $(G - v) - (X \setminus \{v\}) = G - X$. Therefore, $X \setminus \{v\}$ is an r-pseudoforest deletion set of $G - v$. If $v \notin X$, then $(G - v) - X$ is a subgraph of $G - X$, which is an r-pseudoforest. Therefore, X is an r-pseudoforest deletion set of $G - v$. In both cases, $(G - v, S, k, r)$ is also a yes-instance.

Second, suppose $(G - v, S, k, r)$ is a yes-instance and $X \subseteq (V(G) \setminus (S \cup \{v\}))$ is a solution. It follows that $(G-v)-X$ is an r-pseudoforest. Let C be the connected component in $G - X$ that contains v. By the definition of X, $C-v$ is an r-pseudoforest, as it is a connected component in $(G-v)-X$. Since $d_G(v) = 1$, C is also an r-pseudoforest according to Lemma 4. Furthermore, $(G - v) - X$ and $G - X$ only differ at the connected component C. Thus, $X \subseteq (V(G) \setminus S)$ is an r-pseudoforest deletion set of G. So (G, S, k, r) is also a yes-instance. □

Reduction Rule 2: If there is a vertex $v \in (V(G) \setminus S)$ such that $G[S \cup v]$ is not an r-pseudoforest, then return $(G - v, S, k - 1, r)$.

Lemma 6. *Reduction Rule 2 is safe.*

Proof. We may assume that $G[S]$ is an r-pseudoforest, as otherwise (G, S, k, r) is a no-instance of Disjoint r-Pseudoforest Deletion. If $G[S \cup v]$ is not an r-pseudoforest, then every r-pseudoforest deletion set disjoint from S contains v. Thus, (G, S, k, r) and $(G - v, S, k - 1, r)$ are equivalent instances. □

Reduction Rule 3: If there is a vertex $u \in (V(G) \setminus S)$, such that $d_G(u) = 2$ and it has at least one neighbor in $V(G) \setminus S$, then return (G', S, k, r), where G' is obtained from G by bypassing u.

Lemma 7. *Reduction Rule 3 is safe.*

Proof. Note that the bypassing operation decreases both the number of edges and the number of vertices by one.

First, suppose (G, S, k, r) is a yes-instance of Disjoint r-Pseudoforest Deletion. Let X be a minimal r-pseudoforest deletion set of size at most k, which is disjoint from S. Let u be the bypassed vertex of degree 2 and G' be the graph obtained from G by bypassing u. Let x, y be the two neighbors of u in G. As u has at least one neighbor in $V(G) \setminus S$, we may assume $x \in (V(G) \setminus S)$.

If $u \in X$, then $(N(u) \setminus S) \cap X = \emptyset$, as otherwise, $X \setminus \{u\}$ is also an r-pseudoforest deletion set, contradicting the fact that X is a minimal solution. Therefore, $x \notin X$ and $X' = (X \setminus \{u\}) \cup \{x\}$ is an r-pseudoforest deletion set of G', with $|X'| \leq |X| \leq k$. Thus, if (G, S, k, r) is a yes-instance, then (G', S, k, r) is also a yes-instance.

If $u \notin X$, then u is an isolated vertex or is in the connected component containing x or y (or both x and y) in $G - X$. We get $G' - X$ from $G - X$ by deleting u or bypassing it. Either way, $G' - X$ is an r-pseudoforest according to Lemma 2 and Lemma 3.

Second, assume (G', S, k, r) is a yes-instance of Disjoint r-Pseudoforest Deletion. Let X' be a minimal r-pseudoforest deletion set of size at most k, which is disjoint from S. Observe that $G' - X$ is an r-pseudoforest. If $\{x, y\} \cap X' = \emptyset$, then $G - X'$ can be obtained from $G' - X'$ by subdividing the edge xy and naming the new vertex u. According to Lemma 3, $G - X'$ is an r-pseudoforest. If $\{x, y\} \cap X \neq \emptyset$, then $d_{G-X'}(u) \leq 1$. According to Lemma 4, $G - X'$ is an r-pseudoforest. It follows that X' is also a solution for (G, S, k, r) and so (G, S, k, r) is a yes-instance. □

Reduction Rule 4: If $k < 0$, then return no.

Reduction Rules 1-4 can be applied in time $\Theta(m+n)$. Reduction Rule 1 just deletes all leaves iteratively. Reduction Rule 2 checks whether a graph is an r-pseudoforest, which can be done in time $\Theta(m+n)$ according to Lemma

1. Reduction Rule 3 can be applied in time $\Theta(m+n)$ as searching and bypassing a vertex of degree 2 is a basic operation.

Given an instance (G, S, k, r) of Disjoint r-Pseudoforest Deletion, apply Reduction Rules 1-4 whenever possible.

Now we show how to solve the problem when Reduction Rules 1-4 are not applicable. Define measure $\phi(I) = k + cc(S) + \sum_{C\in\mathcal{C}(G[S])}(r - ex(C))$. Note that initially $\phi(I) \leq k + cc(S) + cc(S)r \leq 2k + (k+1)r + 1 < (k+1)(r+2)$, since $|S| \leq k+1$.

To get a depth bounded search tree, we prove that $\phi(I)$ decreases after each application of the following branching rules.

BR-1. Branching on a vertex $v \notin S$ with $d_S(v) \geq 2$.

In one branch, we put v into the solution and call the algorithm on $(G - \{v\}, S, k-1, r)$. Note that in this branch, $cc(S)$ and $ex(C)$ (for each $C \in \mathcal{C}(G[S])$) remain the same while k decreases by 1. Hence $\phi(I)$ drops by 1.

In the other branch, we put v into S and call the algorithm on $(G, S \cup \{v\}, k, r)$. Let $S' = S \cup \{v\}$. There are the following two possible cases regarding the distribution of $N_S(v)$.

Case 1: $N_S(v)$ belongs to more than one connected component in $G[S]$, thus $cc(S') \leq cc(S) - 1$. Let $C_1, C_2, \ldots, C_t (t \geq 2)$ be the set of connected components in $G[S]$ that are adjacent with v.

To compute the difference between excess sums in S and S', denote

$$\begin{aligned}\sigma_S &= \Sigma_{i\in[t]}(r - ex(C_i))\\ &= rt - \Sigma_{i\in[t]}(ex(C_i))\\ &= rt - \Sigma_{i\in[t]}(|E(C_i)| - |V(C_i)| + 1)\\ &= rt - \Sigma_{i\in[t]}|E(C_i)| + \Sigma_{i\in[t]}|V(C_i)| - t,\end{aligned}$$

$$\begin{aligned}\sigma_{S'} &= r - ex(G[\cup_{i\in[t]}V(C_i) \cup \{v\}])\\ &= r - (|E(\cup_{i\in[t]}V(C_i) \cup \{v\})| - |V(\cup_{i\in[t]}V(C_i) \cup \{v\})| + 1)\\ &= r - (\Sigma_{i\in[t]}|E(C_i)| + d_S(v) - \Sigma_{i\in[t]}|V(C_i)|).\end{aligned}$$

Since $\sigma_S - \sigma_{S'} = r(t-1) + d_S(v) - t \geq r(t-1) \geq r$, $\phi(I)$ drops by at least $1 + \sigma_S - \sigma_{S'} \geq 1 + r(t-1) \geq 1 + r$.

Case 2: $N_S(v)$ belongs to one connected component in $G[S]$, denoted by C^*. Then $cc(S') = cc(S)$, and $ex(G[V(C^*) \cup \{v\}]) - ex(C^*) \geq 1$ as $d_S(v) \geq 2$. Hence $\phi(I)$ decreases by at least 1.

Thus, in BR-1, the measure $\phi(I)$ drops by 1 in one case, and at least $1 + r$ or 1 in the other, while remaining non-negative. In the worst case, it gives us a branching vector $(1, 1)$.

Reduction Rule 1 deals with vertices of degree 1 in G. Reduction Rule 3 deals with vertices of degree 2 that have at least one neighbor in $V(G) \setminus S$. BR-1 deals with vertices of degree 2 that have no neighbor in $V(G)\setminus S$. Thus, after all possible applications of Reduction Rule 1, 3 and BR-1, vertices in $V(G) \setminus S$ have degree at least 3. Moreover, as $d_S(u) \leq 1$ holds for every vertex $u \in G - S$, we have $d_{G-S}(u) \geq 2$, that is $\delta(G - S) \geq 2$. Thus, every vertex $u \in (V(G) \setminus S)$ is in some cycle of $G - S$.

If G is not an r-pseudoforest, there exists some edge between $V(G) \setminus S$ and S, since both $G[S]$ and $G - S$ are r-pseudoforest. After exhaustive applications of BR-1, $d_S(v) \leq 1$ holds for every vertex $v \in G - S$. In the following, we branch on vertices in a shortest path of $G - S$ between vertices in $N_{G-S}(S)$.

BR-2. Branching on a shortest path in $G - S$ between vertices in $N_{G-S}(S)$.

First, consider the case when there is a connected component C in $G - S$ such that there is only one edge uv between C and S, where $v \in C$ and $u \in S$. Note that if the solution should intersect $V(C)$, then it suffices to contain v. Thus, we branch on whether to put v into the solution. In one branch, we put v into the solution, and decrease k by one. In the other branch, we put C into S. Since every vertex in $G - S$ is of degree at least 3, and there is only one edge between C and S, C is not a tree in $G - S$. So both $r - ex(S)$ and $\phi(I)$ decreases by at least one.

Now assume every connected component in $G - S$ has at least two edges to S. Let P be a shortest path in $G - S$, such that only the two end-vertices of P are adjacent with S. Note that we can find such a shortest path in polynomial time. We prove that $|V(P)| \leq 2r + 2$. As every connected component C in $G - S$ is an r-pseudoforest, $|E(C)| - |V(C)| + 1 \leq r$. Every vertex in P has degree at least 3 after exhaustive applications of Reduction Rules 1, 2, 3 and BR-1. And no internal vertex of P has an edge to S, according to the choice of P. Let C_0 be the connected component in $G - S$ containing P, we know that $|E(C_0)| \geq 3/2|V(P)| - 2$, and $ex(C_0) \geq ex(G[V(P)]) \geq 3/2|V(P)| - 2 - |V(P)| + 1$. Because $ex(C_0) \leq r$, we have $|V(P)| \leq 2r + 2$.

Suppose $V(P) = \{v_1, v_2 \ldots, v_t\}$. We branch on whether to delete any vertex in $V(P)$. We consider $t + 1$ branches. In branch $i(\leq t)$, we delete vertex v_i, and call the algorithm on $(G - \{v_i\}, S, k - 1, r)$. Note that for every branch $i(\leq t)$ and connected component $C \in \mathcal{C}(G[S])$, $cc(S)$ and $ex(C)$ do not change, while k decreases by at least 1. Thus, $\phi(I)$ drops by at least 1.

In branch $t + 1$, we do not delete any vertex in $V(P)$, and call the algorithm on $(G, S \cup V(P), k, r)$. For the change of $\sigma(S)$, we may regard $V(P)$ as a single vertex, since only the two end-vertices of P are adjacent with S. If edges between $V(P)$ and S are to the same connected component C in $G[S]$, then $ex(C \cup V(P)) = ex(C) + 1$. Thus, in this branch, $\phi(I)$ decreases by 1. Otherwise, the edges between $V(P)$ and S are to different connected components in $G[S]$. In this case, $cc(S)$ decreases by at least 1, and similarly to Case 1, we can show that $\sigma_S - \sigma_{S'} \geq r$. So $\phi(I)$ drops by at least $1 + r$. This gives us a $(t + 1)$-tuple branching vector $(1, 1, \ldots, 1)$ in which $t \leq 2r + 2$.

Algorithm 1: Branching Algorithm for Disjoint r-Pseudoforest Deletion.

```
  Input: An undirected graph $G$, an $r$-Pseudoforest deletion set $S$,
         two integers $k$ and $r$
1 Parameters: $k$.
  Output: An $r$-pseudoforest deletion set $X$ of $G$ with $|X| \leq k$ and
          $X \cap S = \emptyset$. Or conclude that such a set does not exist.
2 Initially set $X = \emptyset$.
3 Step 1: Apply Reduction Rules 1-4.
4 Step 2: Apply BR-1. Branching on a vertex $v \notin S$ such that
    $d_S(v) \geq 2$. Either put $v$ into $X$ and decrease $k$ by one or put $v$ into
    $S$.
5 Step 3: Apply BR-2. Branching on a shortest path $P$ in $G - S$
    between two vertices in $N_{G-S}(S)$.
6 If $k \geq 0$ then return $X$. Otherwise, return NO.
```

According to the branching vectors in BR-1 and BR-2, and the facts that $\phi(I)$ starts from a value smaller than $(k+1)(r+2)$, drops by at least 1 in each branch and remains non-negative, the algorithm runs in time $O^*((2r+3)^{(k+1)(r+2)})$.

The following lemma states that a fast parameterized algorithm for the disjoint version problem gives a fast algorithm for the original problem.

Lemma 8. *[10] If* Disjoint r-Pseudoforest Deletion *can be solved in time $f(k)n^{O(1)}$, then* r-pseudoforest Deletion *can be solved in time* $\sum_{i=0}^{i=k} \binom{k+1}{i} f(k-i) n^{O(1)}$.

So we get an algorithm for r-Pseudoforest Deletion that runs in time $\sum_{i=0}^{i=k} \binom{k+1}{i} (2r+3)^{(k-i+1)(r+2)} \leq (1+(2r+3)^{r+2})^{k+1}$.

Theorem 1. *There exists an algorithm for* r-Pseudoforest Deletion *with running time $O^*((1+(2r+3)^{r+2})^{k+1})$.*

Proof. We first show that Algorithm 1 solves Disjoint r-Pseudoforest Deletion. After exhaustive applications of Reduction Rule 1, every vertex in $G-S$ has degree at least 2. After exhaustive applications of Reduction Rule 3, every vertex in $G-S$ with degree 2 has two neighbors in S. BR-1 deals with vertices in $G-S$ that has at least two neighbors in S. BR-2 deals with component in $G-S$ in which every vertex has at most one neighbor in S. Thus, BR-1 and BR-2 are exhaustive. After applications of Reduction Rule 3 and BR-1, every vertex in $G-S$ has degree at least 3.

For each branching in BR-1 and BR-2, the measure $\phi(I)$ decreases by at least one. BR-1 has a branching vector $(1,1)$. And BR-2 has a branching $(1,\ldots,1)$, which is a $(2r+3)$-tuple. Thus, Algorithm 1 runs in time

$O^*((2r+3)^{(r+2)(k+1)})$. Using Algorithm 1 as a subroutine, we can solve r-Pseudoforest Deletion in time $(1+(2r+3)^{r+2})^{k+1}$, according to Lemma 8. □

Note that Theorem 1 improves over the result in [22].

Theorem 2. *(Linear Time Protrusion Replacement Theorem, [22]). Let Π be a problem that has a protrusion replacer which replaces p protrusions of size at least p' for some fixed p. Let s and β be constants such that $s \geq p'2^p$ and $p \geq 3(\beta+1)$. Given an instance (G, k) as input, there is an algorithm that runs in time $O(m+n)$ and produces an equivalent instance (G', k') with $|V(G')| \leq |V(G)|$ and $k' \leq k$. If additionally G has an (α, β)-protrusion decomposition such that $\alpha \leq n/244s$, then we have that $|V(G')| \leq (1-\delta)|V(G)|$ for some constant $\delta > 0$.*

The algorithm for r-Pseudoforest Deletion given in [22] makes use of Theorem 2 and runs in time $O^*(c_r^k)$, where c_r is a constant that depends on r only. However, it requires that $c_r^d > 2$ where $2d < \rho$, $\rho \leq \frac{1}{12000s(r+2)^2}$ and $s = p'2^{6(r+2)}$ for some positive integer p'. It follows that $c_r > 2^{240000s(r+2)^2} = 2^{240000(r+2)^2 p' 2^{6(r+2)}}$.

4 Kernelization of r-Pseudoforest Deletion

In this section, we give an improved kernel for r-Pseudoforest Deletion. We will use Reduction Rules A, B, C and D. The proofs for Reduction Rules A and B are similar to those of Reduction Rules 1 and 3, and they are simpler.

Reduction Rule A: Let (G, k, r) be an instance of r-Pseudoforest Deletion. If there is a vertex $v \in V(G)$ such that $d_G(v) = 1$, then return $(G-v, k, r)$.

Lemma 9. *Reduction Rule A is safe.*

Proof. We show that (G, k, r) and $(G-v, k, r)$ are equivalent instances of r-Pseudoforest Deletion.

On the one hand, suppose (G, k, r) is a yes-instance and $X \subseteq V(G)$ is a solution. If $v \in X$, then $(G-v)-(X \setminus \{v\}) = G-X$. Thus, $X \setminus \{v\}$ is a solution to $(G-v, k, r)$. Otherwise, $(G-v)-X$ is a subgraph of $G-X$, which is an r-pseudoforest. According to Lemma 2, $(G-v)-X$ is an r-pseudoforest. In both the cases, $(G-v, k, r)$ is also a yes-instance.

On the other hand, suppose $(G-v, k, r)$ is a yes-instance and $X \subseteq (V(G) \setminus \{v\})$ is a solution. Let C be the connected component in $G-X$ containing v. By the definition of X, $C-v$ is an r-pseudoforest in $(G-v)-X$. Since $d_G(v) = 1$, C is also an r-pseudoforest in $G-X$ according to Lemma 4. Furthermore, $(G-v)-X$ and $G-X$ only differ at C. Thus, $X \subseteq V(G)$ is an r-pseudoforest deletion set of G. So (G, k, r) is also a yes-instance. □

Reduction Rule B: Let (G, k, r) be an instance of r-Pseudoforest Deletion. If there is a vertex $u \in V(G)$ such that $d_G(u) = 2$, then bypass it.

Lemma 10. *Reduction Rule B is safe.*

Proof. Note that bypassing u decreases both the edge number and vertex number by one.

On the one hand, suppose (G, k, r) is a yes-instance of r-Pseudoforest Deletion, and X is a minimal r-pseudoforest deletion set of size at most k. Let u be the bypassed vertex of degree 2 and G' be the graph obtained from G by bypassing u. Let x, y be the two neighbors of u in G.

If $u \in X$, then by the minimality of X, at least one neighbor of u is not in X. Without loss of generality, we may assume that $x \notin X$, then $X' = (X \setminus \{u\}) \cup \{x\}$ is an r-pseudoforest deletion set of G' with $|X'| \leq |X| \leq k$. Thus, if (G, k, r) is a yes-instance, then (G', k, r) is also a yes-instance.

If $u \notin X$, then u is an isolated vertex or is in the connected component containing x or y (or both x and y) in $G - X$. We get $G' - X$ from $G - X$ by deleting u or bypassing it. According to Lemma 2 and Lemma 3, $G' - X$ is an r-pseudoforest.

On the other hand, suppose (G', k, r) is a yes-instance of r-Pseudoforest Deletion, and X' is a minimal r-pseudoforest deletion set of size at most k. If $\{x, y\} \cap X' = \emptyset$, then $G - X'$ can be obtained from $G' - X'$ by subdividing xy and naming the new vertex u. If at least one of x and y is in X', then u has degree 0 or 1 in $G - X'$. In both cases, $G - X'$ is also an r-pseudoforest. It follows that X' is also a solution for (G, k, r) and so (G, k, r) is a yes-instance. □

Reduction Rule C: Let (G, k, r) be an instance of r-Pseudoforest Deletion. If there is a connected component in G that is an r-pseudoforest, then delete the component from G.

Lemma 11. *Reduction Rule C is safe.*

Proof. Let C be a connected component in G that is an r-pseudoforest. There is no need to delete any vertex from C to get an r-pseudoforest. □

After Reduction Rule C, every connected component in G is not an r-pseudoforest.

Reduction Rule D: If there are more than k connected components in G, then return no.

Lemma 12. *Reduction Rule D is safe.*

Proof. Let X^* be a minimum r-pseudoforest deletion set of G. Since Reduction Rule C is not applicable, for every connected component C in G, we have $V(C) \cap X^* \neq \emptyset$. It follows that $|X^*| > k$ and (G, k, r) is a no-instance. □

Lemma 13. *After exhaustive applications of Reduction Rules A, B and C, we get an instance with minimum degree at least 3.*

Proof. Reduction Rule A deals with vertices of degree 1 and Reduction Rule B deals with vertices of degree 2. Isolated vertices are dealt with Reduction Rule C. Thus, if Reduction Rules A, B and C are not applicable, the reduced instance has minimum degree at least 3. □

Lemma 14. *Let G be a graph such that $\delta(G) \geq 3$ and $\Delta(G) \leq d$. If G has an r-pseudoforest deletion set of size at most k, then it contains at most $(2dr - d + 1)k$ vertices and at most $3kdr - kd$ edges.*

Proof. Let X be an r-pseudoforest deletion set of G of size at most k. Denote $F = G - X$. After exhaustive applications of Reduction Rule C, there is no connected component in G that is an r-pseudoforest. Therefore, every connected component C in F is created due to the deletion of X, and so $E(V(C), X) \neq \emptyset$. Let c be the number of connected components in F. We have $c \leq kd$, since deleting any vertex of degree t produces at most t connected components. Because F is an r-pseudoforest, for each connected component C_i with $i \in [c]$, we have $|E(C_i)| \leq |V(C_i)| - 1 + r$. Thus, $|E(F)| = \Sigma_{i\in[c]}|E(C_i)| \leq \Sigma_{i\in[c]}(|V(C_i)| - 1 + r) = |V(F)| + c(r - 1)$. By counting the number of edges incident with $V(F)$, we have

$$3|V(F)| \leq 2(|E(F)|) + |E(X, V(F))| \leq 2(|V(F)| + c(r - 1)) + kd.$$

It follows that $|V(F)| \leq 2c(r-1)+kd$. So $|V(G)| \leq |X|+|V(F)| \leq 2c(r-1)+k(d+1) \leq (2dr-d+1)k$. And $|E(G)| \leq |E(F)|+|E(X, V(F))|+|E(G[X])| \leq |V(F)| + c(r - 1) + kd < 3c(r - 1) + 2kd = 3kdr - kd$. □

We need to make use of the following result.

Theorem 3 ([22]). *Given an instance (G, k) of r-*Pseudoforest Deletion*, in polynomial time, we can get an equivalent instance (G', k') such that $k' \leq k$, $|V(G')| \leq |V(G)|$ and $\Delta(G') \leq (k + r)(3r + 8)$.*

Theorem 4. *The r-*Pseudoforest Deletion *problem admits a kernel with $O(\max\{k^2r^2, kr^3\})$ vertices and $O(\max\{k^2r^2, kr^3\})$ edges.*

Proof. By Theorem 3, r-Pseudoforest Deletion admits a kernel with maximum degree at most $d = (k + r)(3r + 8)$. Thus, by Lemma 14, we can get a kernel for r-Pseudoforest Deletion which contains at most $(2r - 1)kd + k = O(\max\{k^2r^2, kr^3\})$ vertices, and at most $(3r - 1)kd = (3r - 1)k(k + r)(3r + 8) = O(\max\{k^2r^2, kr^3\})$ edges. □

The algorithm in [22] also gave a kernel with at most $O(ck^2)$ vertices, but there is no explicit expression of c. The kernelization algorithm in [22] uses Theorem 2 and a $(4kd(r+2), 2(r+2))$-protrusion decomposition. According to Theorem 2, the size of the kernel obtained in [22] is upper bounded by $n \leq 244s\alpha$, where $\alpha = 4kd(r+2)$, $d \leq (k+r)(3r+8)$, $s \geq p'2^p$, p' is a positive integer, and $p \geq 3(\beta+1) \geq 6(r+2)$. Thus, the constant c in [22] depends on r exponentially.

5 d-Quasi-Forest Deletion

Recall that for a positive integer d, a *d-quasi-forest* is a graph in which every connected component has a feedback vertex set of size at most d. It is easy to observe that a subgraph of a d-quasi-forest is also a d-quasi-forest.

Definition 2. *Given a graph $G = (V, E)$, a subset $X \subseteq V(G)$ is a d-*quasi-forest deletion set *of G if $G - X$ is a d-quasi-forest.*

Let us define the parameterized d-Quasi-Forest Deletion we will study.

d-Quasi-forest Deletion
Instance: An undirected graph G, positive integers d and k.
Parameter: d and k.
Output: Decide if there is a d-quasi-forest deletion set $X \subseteq V(G)$ of size at most k.

Here we recall the definition of tree decomposition of a graph.

Definition 3. *Given an undirected graph $G = (V, E)$, a tree decomposition of G is a pair $(\mathcal{T}, \beta)$, where $\mathcal{T}$ is a tree and $\beta : V(\mathcal{T}) \to 2^V$ such that*

1. *$\bigcup_{x \in V(\mathcal{T})} \beta(x) = V$;*
2. *for each edge $uv \in E$, there exists a node $x \in V(\mathcal{T})$ such that $u, v \in \beta(x)$;*
3. *for each $v \in V$, the set $\beta^{-1}(v)$ of nodes induces a connected subgraph in $\mathcal{T}$.*

The width of $(\mathcal{T}, \beta)$ is $\max_{x \in V(T)}(|\beta(x)| - 1)$. The treewidth of G (denoted by $tw(G)$) is the minimum width of all tree decompositions of G. For each vertex $v \in \mathcal{T}$, $\beta(v)$ is called a bag.

Let us recall two well-known facts about treewidth.

Lemma 15 (Folklore)**.** *The treewidth of a forest is 1.*

Lemma 16 (Folklore)**.** *Let $G' = G - S$ for some $S \subseteq V(G)$. Then $tw(G) \leq tw(G') + |S|$.* □

The Monadic Second-Order Logic (MSO_2) is a fragment of second-order logic where the second order quantification is limited to quantification over sets. It contains logical operations $\vee, \wedge, \rightarrow, \leftrightarrow, \neg$, and variables for vertices, edges, set of vertices and set of edges. Binary relations like $u \in X$, $e \in E$, $inc(u, e)$, $adj(u, v)$ are also included in MSO_2. We refer readers to [10, 12, 13] for a more comprehensive introduction of MSO_2.

The following is a famous theorem that establishes parameterized tractability of evaluating MSO_2 formulas over graphs with bounded treewidth.

Theorem 5. *(Courcelle [9]) Given a graph G and a formula φ in MSO_2 describing a property $\mathcal{P}$, it takes time $f(tw(G), |\varphi|)n^{O(1)}$ to decide whether G has property $\mathcal{P}$.*

We show that d-Quasi-Forest Deletion is FPT by Theorem 5. Firstly, we bound the treewidth of the input graph with d and k.

Lemma 17. *If (G, k) is a yes-instance of d-Quasi-Forest Deletion, then $tw(G) \leq k + d + 1$.*

Proof. Let (G, k) be a yes-instance of d-Quasi-Forest Deletion, then there is a vertex set $X \subseteq V(G)$, such that each connected component in $G - X$ has a feedback vertex set of size at most d. Thus, $tw(G - X) \leq d + 1$, according to Lemma 15 and Lemma 16. As $|X| \leq k$ and $tw(G - X) \leq d + 1$, we have $tw(G) \leq k + d + 1$, according to Lemma 16. □

Secondly, we will prove the following:

Lemma 18. *The property of admitting a d-quasi-forest deletion set of size at most k is expressible in MSO_2.*

Proof. The property of admitting a d-quasi-forest deletion set of size at most k can be expressed as follows:

$$\exists v_1, \ldots, v_k \in V(G) \forall X \subseteq V(G) \setminus \{v_1, \ldots, v_k\}(Conn(X) \rightarrow (FVS(X) \leq d)),$$

where the property $Conn(X)$ means that $G[X]$ is a connected subgraph of G and $FVS(X) \leq d$ means that $\text{fvs}(G[X]) \leq d$. The reason that we can require that $G[X]$ is a connected subgraph of G and not just a connected component of G is that replacing "connected subgraph" with "connected component" will not change the implication $Conn(X) \rightarrow (FVS(X) \leq d)$ as if $\text{fvs}(H) \leq d$ for a graph H then $\text{fvs}(H') \leq d$ for any subgraph H' of H. Note that an MSO_2-formula is given in [10, Section 7.4.1] for the property $Conn(X)$. We will omit some simple details such as how one can express $\forall X \subseteq (V(G) \setminus \{v_1, \ldots, v_k\})$ in MSO_2. Thus, it remains to prove that $FVS(X) \leq d$ is expressible in MSO_2.

Note that [10, Section 7.4.1] provides an MSO_2-formula for the property that the degree of a vertex v in a spanning edge set C of G is 2 (denoted by

$Deg2(v, C)$). We will also need a simple property $inc(v, C)$ that a vertex v is incident to an edge of C, where C is an edge set of G. It can be expressed as follows: $\exists e \in C(inc(v, e))$. Now $FVS(X) \leq d$ can be expressed in MSO_2 as follows:

$$\exists y_1, \ldots, y_d \in X \neg(\exists C \subseteq E(G)(\forall v \in X \backslash \{y_1, \ldots, y_d\}(inc(v, C) \rightarrow Deg2(v, C)))).$$

□

Combining Lemmas 17 and 18 and Theorem 5, we have the following result.

Theorem 6. d-Quasi-Forest Deletion *is FPT parameterized by k and d.*

Unfortunately, the complexity of the algorithm obtained by Theorem 6 may be a power tower exponential. Aiming at fully exploiting the problem structure and designing a faster algorithm, we solve d-Quasi-Forest Deletion by the iterative compression approach.

Now we are going to use the method of iterative compression, which starts by replacing the original problem with its compression version.

d-Quasi-Forest Deletion Compression
Instance: An undirected graph G, an integer k, and a d-quasi-forest deletion set $Z \subseteq V(G)$ with $|Z| \leq k + 1$.
Parameter: d and k.
Output: Decide if there is a d-quasi-forest deletion set $X \subseteq V(G)$ with $|X| \leq k$.

By guessing the intersection of Z and X, we reduce d-Quasi-Forest Deletion Compression into its disjoint version defined as follows.

Disjoint d-Quasi-Forest Deletion
Instance: An undirected graph G, an integer k and a d-quasi-forest deletion set $Z \subseteq V(G)$.
Parameter: d and k.
Output: Decide if there is a d-quasi-forest deletion set $X \subseteq V(G)$ with $|X| \leq k$ and $X \cap Z = \emptyset$.

Initially we have $|Z| = z_0 \leq k + 1$. In the process of solving Disjoint d-Quasi-Forest Deletion, we will put vertices into Z and give the new upper bound on $|Z|$. By the algorithm in [18], we can decide whether a given graph is a d-quasi-forest in time $O^*(3.460^d)$ and compute a minimum feedback vertex set of it.

A connected component that is not a tree is called a *non-tree component*.

Lemma 19. *Let (G, Z, k) be a yes-instance of* Disjoint d-Quasi-Forest Deletion. *For every vertex $u \in Z$, $G - Z$ contains at most $k + d$ non-tree components that are adjacent with u.*

Proof. Suppose there are more than $k + d$ non-tree components in $G - Z$, that are adjacent with $u \in Z$. Note that each non-tree component contains at least one cycle. For any vertex set $X \subseteq (V(G) \setminus Z)$, such that $|X| \leq k$, let C_X^u be the connected component in $G - X$ that contains u. Then C_X^u is not a d-quasi-forest, since it contains at least $d + 1$ vertex-disjoint cycles. Thus, (G, Z, k) is a no-instance. Hence, for each vertex $u \in Z$, the number of non-tree components in $G - Z$ that are adjacent with u is at most $k + d$. □

Let (G, Z, k) be an instance of Disjoint d-Quasi-Forest Deletion. Since Z is a d-quasi-forest deletion set of G, every connected component in $G - Z$ has a feedback vertex set of size at most d. For each connected component in $G - Z$, we branch on a smallest feedback vertex set D of it. That is, for every vertex in D, either put it into X (and decrease k by one) or put it into Z. After all the branching, we get new instances in which every connected component in $G - Z$ is a tree. Moreover, the size of Z now is upper bounded by $z_1 = z_0 + z_0(k + d)d = (k + 1)(kd + d^2 + 1) = O(k^2d^2)$. Indeed, initially $|Z| = z_0 \leq k + 1$. By Lemma 19, every vertex in the initial Z has neighbors in at most $k+d$ non-tree components. It follows that $G-Z$ contains at most $(k+1)(k+d)$ non-tree components, each containing at most d vertices that might be added into Z.

Now we solve Disjoint d-Quasi-forest Deletion in which every connected component in $G - Z$ is a tree, and $|Z| \leq z_1$.

Definition 4. *Two trees T_1, T_2 in $G - Z$ have the same* neighborhood type *if $N_Z(T_1) = N_Z(T_2)$ and for every vertex $u \in N_Z(T_1)$, $|E(u, T_1)| = 1$ if and only if $|E(u, T_2)| = 1$.*

Reduction Rule I: If there are more than $k + d + 2$ trees in $G - Z$ that have the same neighborhood type, then delete all but $k+d+2$ of these trees.

Lemma 20. *Reduction Rule I is safe.*

Proof. Let (G, Z, k) be an instance of Disjoint d-Quasi-Forest Deletion. Suppose there are t trees with the same neighborhood type, denoted by $T_1, T_2, \ldots, T_t$, where $t \geq k + d + 3$. Let $N = \cap_{i \in [t]} N_Z(V(T_i))$ be the common neighborhood of $T_1, T_2, \ldots, T_t$ in Z. By deleting $T_{k+d+3}, T_{k+d+4}, \ldots, T_t$, we get a new instance (G', Z, k).

To prove the safeness of Reduction Rule I, we show that (G, Z, k) is a yes-instance if and only if (G', Z, k) a yes-instance.

On the one hand, if (G, Z, k) is a yes-instance then (G', Z, k) a yes-instance, as G' is a subgraph of G.

On the other hand, suppose (G', Z, k) is a yes-instance. Then there is a vertex set $X \subseteq (V(G') \setminus Z)$ such that $|X| \leq k$ and each connected component in $G' - X$ has a feedback vertex set of size at most d. Note that at least $d+2$ trees in $\{T_1, T_2, \ldots, T_{k+d+2}\}$ are disjoint from X and each of these trees connects every pair of vertices in N. Thus, $G' - X$ contains at least $d+2$ vertex-disjoint paths between every pair of vertices in N. Let C_N be the connected component that contains N in $G' - X$. As deleting d vertices destroy at most d such paths, every feedback vertex set of C_N with size at most d contains at least $|N| - 1$ vertex of N.

Let D be a feedback vertex set of C_N with size at most d. We know that if $N \setminus D \neq \emptyset$, then $|N \setminus D| = 1$. Recall that $T_1, T_2, \ldots, T_t$ are of the same neighborhood type. In particular, $|E(u, V(T_i))| = 1$ if and only if $|E(u, V(T_j))| = 1$, for every vertex $u \in N$, $i \in [k+d+2]$ and $j \in \{k+d+3, k+d+4, \ldots, t\}$. Thus, for every vertex $u \in N$, if u forms a cycle with T_j, then it forms a cycle with T_i, for every $i \in [k+d+2]$ and $j \in \{k+d+3, k+d+4, \ldots, t\}$. Therefore, if $N \setminus D \neq \emptyset$, then $T_{k+d+3}, T_{k+d+4}, \ldots, T_t$ do not form any cycle with the only vertex in $N \setminus D$. And so D is a feedback vertex set of $G[V(C_N) \cup (\cup_{j \in \{k+d+3, k+d+4, \ldots, t\}} V(T_j))]$ no matter $N \setminus D$ is empty or not. It follows that $G - X$ is a d-quasi-forest, and (G, Z, k) is also a yes-instance. □

5.1 Forced Vertices

Definition 5. *Let (G, Z, k) be a yes-instance of* Disjoint d-Quasi-Forest Deletion. *A vertex $u \in Z$ is* forced*, if for every vertex set $X \subseteq (V(G) \setminus Z)$, with $|X| \leq k$, u is in every feedback vertex set of size at most d of the connected component containing u in $G - X$.*

The intuition of raising the notion of "forced vertex" is as follows. Every connected component in $G - Z$ is a tree. If all the neighbors of a tree T in Z are forces vertices, then the size of T does not affect the feedback vertex number of the component containing T. The notion of a forced vertex thus helps us to bound the size of each tree in $G - Z$, as we just need to keep enough neighbors of forced vertices in $G - Z$ such that they are still forced vertices.

Lemma 21. *Let v be a vertex in Z. If there is a v-flower with at least $k+d+1$ cycles in $G - (Z \setminus \{v\})$, then v is a forced vertex.*

Proof. For any vertex set $X \subseteq (V(G) \setminus Z)$, with $|X| \leq k$, there is a v-flower with at least $d+1$ cycles in $G - ((Z \cup X) \setminus \{v\})$. Thus, v is in every feedback vertex set of size at most d of the component containing v in $G - X$. And so v is a forced vertex.

□

The following theorem provides a way to check whether a vertex in Z is a forced vertex with no false positive error. That is, if the following procedure claims that a vertex is a forced vertex, then it is a forced vertex. However, it is possible that the procedure does not find all forced vertices.

Theorem 7. *(Gallai [16]) Given a simple graph G, a vertex set $R \subseteq V(G)$, and an integer s, in polynomial time one can either*

1. *find a family of $s + 1$ pairwise vertex-disjoint R-paths, or*
2. *conclude that no such family exists and, find a set B of at most $2s$ vertices, such that in $G - B$ no connected component contains more than one vertex of R.*

Suppose we want to decide whether a vertex $u \in Z$ is a forced vertex. Set $s = k + d$, and $R = N(u) \setminus Z$. Recall that an R-path is a path between two vertices in R that is disjoint from R internally. Thus, every R-path forms a cycle with u. We use Theorem 7 to check whether $G - Z$ contains $k + d + 1$ pairwise vertex-disjoint R-paths. If yes, then these pairwise vertex-disjoint paths together with u form a u-flower with $k + d + 1$ cycles. Thus, u is a forced vertex, according to Lemma 21.

The following simple lemma is useful in our arguments below.

Lemma 22. *Let T be a tree and $U \subset V(T)$. Then every tree in $T - U$ contains at most one neighbor of every vertex in U.*

Proof. If there is a tree in $T - U$ that contains more than one neighbor of some vertex $u \in U$, then T contains a cycle, which is not possible. □

For each vertex $u \in Z$, we decide whether it is a forced vertex via Theorem 7. According to the result, we get a partition of $Z = N_1 \cup N_2$, where N_1 contains all the forced vertices found by Theorem 7 and $N_2 = Z \setminus N_1$.

The following lemma helps us to bound the size of each tree in $G - Z$.

Lemma 23. *Let (G, Z, k) be an instance of* Disjoint d-Quasi-Forest Deletion *such that every component in $G - Z$ is a tree, and $|Z| \leq z_1$. Then it can be reduced to at most $4^{z_1(k+d)}$ instances (G', Z', k') such that $G' - Z'$ contains at most $(k+d+2)2^{z_1(2(k+d)+1)}$ trees, each of which contains at most one neighbor of every non-forced vertex in Z'.*

Proof. By Theorem 7, for each vertex $u \in N_2$, we can find a set $B_u \subseteq (V(G) \setminus Z)$ with $|B_u| \leq 2(k + d)$, such that each tree in $(G - Z) - B_u$ contains at most one neighbor of u. For each vertex $v \in B_u$, branch on putting it into X or into Z. After doing this for all vertices in B_u, every tree T in $G - Z$ contains at most one neighbor of u. Moreover, every tree T in $G - Z$ contains at most one neighbor of every vertex in B_u according to Lemma 22. It follows that vertices in B_u will not become forced vertices when put into Z. Update the partition $Z = N_1 \cup N_2$ by checking if there is any vertex in N_1 becomes a non-forced vertex after the branching for B_u for every $u \in N_2$. Denote $B = \cup_{u \in N_2} B_u$.

After doing the above procedure for every vertex in N_2, every tree in $G - Z$ contains at most one neighbor of every vertex in $N_2 \cup B$. As $|B_u| \leq 2(k+d)$, we have $|B| \leq 2z_1(k+d)$. Thus, the above procedure produces at most $2^{|B|} \leq 4^{z_1(k+d)}$ instances (G', Z', k') in time $O(4^{z_1(k+d)})$.

For each tree T in $G - Z$, $N_Z(T) \subseteq (N_1 \cup N_2 \cup B)$. Thus, there are at most $2^{|N_1 \cup N_2 \cup B|}$ different neighborhood types. After applying Reduction Rule I, there are at most $k + d + 2$ trees for each neighborhood type. Thus, the number of trees in $G - Z$ is at most $(k + d + 2)2^{|N_1 \cup N_2 \cup B|} = (k + d + 2)2^{z_1(2(k+d)+1)}$. $\square$

Now we just need to bound the size of every tree in $G - Z$. Note that all neighbors of these trees in Z belong to $Z = N_1 \cup N_2 \cup B$.

Construct an auxiliary graph H with $V(H) = \{v' | v \in N_1\}$ in the following way. We call u and u' the *partner* of each other. Let $C_{u'}$ be the connected component in the current H that contains u'. Note that the connected components of H will be updated until no more edge is added into H. For any vertex $u' \in V(H)$, denote $V(C_u) = \{v | v' \in V(C_{u'})\}$. For two vertices $u', v' \in V(H)$, add an edge $u'v'$ into H if $G - (Z \setminus \{u, v\})$ contains at least $k + 1$ vertex-disjoint paths between $V(C_u)$ and $V(C_v)$. Keep adding edges into H until there is no vertex pair u', v' satisfying this condition. The construction of H can be done in polynomial time, as computing the maximum number of vertex-disjoint paths between two vertex sets in an undirected graph can be done by computing the maximum flow between them. Let p be the number of connected components in the final H. We have $p \leq |N_1| \leq z_1$.

Note that $G - Z$ contains at most k vertex-disjoint paths between vertices in $V(C_u)$ and $V(C_v)$, for any two different connected components $C_{u'}$ and $C_{v'}$ of H. According to Menger's Theorem, there is a set W with at most $k\binom{p}{2}$ vertices, such that for every tree T in $(G - Z) - W$, the partners of vertices in $N_Z(T)$ belong to at most one connected component of H. Now we just need to bound the size of each such tree.

For each vertex in W, branch on putting it into X or into Z. There are at most $2^{|W|}$ such guesses. After the branching, $N_Z(V(G - Z)) \subseteq (W \cup N_1 \cup N_2 \cup B)$. Moreover, each tree in $G - Z$ contains at most one neighbor of every vertex in W that is put into Z, according to Lemma 22.

For every vertex $u \in N_1$ and every integer $i \in [k+1]$, add vertices u'_i, u''_i and edges $u'_i u''_i, uu'_i, uu''_i$ into Z. These edges together with u form a u-flower with $k + 1$ cycles. This operation adds $2(k + 1)|N_1|$ vertices into Z. After adding these vertices, every forced vertex will not become non-forced due to the applications of Reduction Rules II and III. This is crucial for the safeness of Reduction Rule IV and V.

In the construction of H, vertices u', v' belong to the same connected component of H, if $G - Z$ at least $k + 1$ vertex-disjoint paths between u and v. It follows that u and v belong to the same connected component of $G - X$, for any $X \subseteq (V(G) \setminus Z)$. However, the applications of Reduction Rules IV and V, which delete vertices from $G - Z$, might affect this property. For the safeness of Reduction Rule IV and V, add edge uv into G for any two vertices $u, v \in N_1$, if u' and v' belong to the same connected component of H.

For every tree T in $G - Z$, we partition its vertex set into four pairwise disjoint subsets V_1, V_2, V_3 and V_4:
$V_1 = \{u \in V(T) | N_Z(u) = \emptyset\}$,
$V_2 = \{u \in V(T) | N_Z(u) \setminus N_1 \neq \emptyset\}$,
$V_3 = \{u \in (V(T) \setminus V_2) | u$ has two neighbors v, w, such that v' and w' are in two different connected components of $H\}$,
$V_4 = V(T) \setminus (V_1 \cup V_2 \cup V_3)$.

Every vertex in V_1 has no edge to Z; every vertex in V_2 has an edge to $Z \setminus N_1$; every vertex in V_3 has an edge to partners of vertices in more than one connected components of H; every vertex in V_4 has an edge to partners of vertices in exactly one connected component of H. After the reduction rules described in the next section, we bound the size of the four subsets.

5.2 Some More Reduction Rules

The proofs of safeness for the following two reduction rules are omitted here, as they are similar with those of Lemmas 9 and 10.

Reduction Rule II: If there is a vertex $u \in V_1$ that has degree one, then delete u and return a new instance $(G - u, k, Z)$.

Lemma 24. *Reduction Rule II is safe.*

Reduction Rule III: If there is a vertex $v \in V_1$ such that $d_G(v) = 2$, then bypass it.

Lemma 25. *Reduction Rule III is safe.*

After exhaustive applications of Reduction Rules II and III, every vertex in V_1 has degree at least 3.

Reduction Rule IV: If there is a vertex $v \in V_4$ such that $d_{G-Z}(v) = 1$, then delete v.

Lemma 26. *Reduction Rule IV is safe.*

Proof. Let T^* be the tree in $G - Z$ that contains v. Let u be the neighbor of v in $G - Z$. Let G' be the graph obtained by deleting v. As $v \in V_4$, we may suppose the partners of vertices in $N_Z(v)$ belong to C_i, for some

$i \in [p]$. We prove that (G, k, Z) is a yes-instance if and only if (G', k, Z) is a yes-instance.

On the one hand, suppose X is a solution to (G, k, Z) for Disjoint d-Quasi-Forest Deletion. Then $G - X$ is a d-quasi-forest.

If $X \cap \{u, v\} = \emptyset$, then $G' - X$ can be obtained from $G - X$ by deleting v. As $G - X$ is a d-quasi-forest, $G' - X$ is also a d-quasi-forest. So (G', k, Z) is a yes-instance.

If $v \in X$, then $(X \setminus \{v\}) \cup \{u\}$ is also a d-quasi-forest deletion set of G. Otherwise, there is a connected component C in $G - ((X \setminus \{v\}) \cup \{u\})$, such that $fvs(C) > d$. Note that C contains v, and $fvs(C - v) \leq d$. Because all vertices in $N_Z(v)$ are forced vertices, $N_Z(v)$ belong to every feedback vertex set (of $C - v$) of size at most d. It follows that $d_{C-D}(v) = 0$, for any feedback vertex set D of $C - v$. Therefore, D is also a feedback vertex set of C, and so $fvs(C) \leq d$, a contradiction. Thus, if $X \cap \{u, v\} \neq \emptyset$, we may assume $v \notin X$ and $u \in X$. In this case, $G' - X$ is a subgraph of $G - X$, thus is also a d-quasi-forest. Therefore, (G', k, Z) is a yes-instance.

On the other hand, let X be a solution to (G', k, Z) for Disjoint d-Quasi-Forest Deletion. According to the definition of H, there are at least $k + 1$ vertex-disjoint paths between partners of any two vertices in C_i. Thus, deleting X cannot separate the partners of any two vertices in $V(C_i)$. Therefore, there is a connected component in $G - X$ that contains all vertices $V(C_i)$. Denote this connected component by C. It follows that $N_Z(v) \subseteq V(C)$. To prove X is also a solution to (G, k, Z) for Disjoint d-Quasi-Forest Deletion, it suffices to show that $fvs(C) \leq d$.

Let D be a feedback vertex set of the component in $G' - X$ that contains $N_Z(v)$, such that $|D| \leq d$. As $v \in V_4$, all vertices in $N_Z(v)$ are forced vertices and so $N_Z(v) \subseteq D$. If $u \in X$, then $d_{(G-X)-D}(v) = 0$. If $u \notin X$, then $d_{(G-X)-D}(v) = 1$. In both cases, $C - D$ is a forest and so $fvs(C) \leq d$. It follows that $G - X$ is a d-quasi-forest, and (G, k, Z) is a yes-instance. □

Reduction Rule V: If there is a vertex $v \in V_4$ such that $d_{G-Z}(v) = 2$, then delete v and add an edge between its two neighbors in $G - Z$.

Lemma 27. *Reduction Rule V is safe.*

Proof. Let T^* be the tree in $G - Z$ that contains v. Let u, w be the two neighbor of v in $G - Z$. Let G' be the graph obtained by deleting v and adding an edge between u and w. Suppose the partners of vertices in $N_Z(v)$ belong to C_i, for some $i \in [p]$. We prove that (G, k, Z) is a yes-instance if and only if (G', k, Z) is a yes-instance.

On the one hand, suppose X is a solution to (G, k, Z) for Disjoint d-Quasi-Forest Deletion. Then $G - X$ is a d-quasi-forest.

If $X \cap \{u, v, w\} = \emptyset$, then $G' - X$ can be obtained from $G - X$ by deleting v and adding an edge between u and w. Let C be the connected component in $G - X$ that contains $\{u, v, w\}$. Then $N_Z(v) \subseteq V(C)$. Let C'

be the connected component obtained from C by deleting v and add an edge between u and w. To prove $G' - X$ is also a d-quasi-forest, it suffices to prove $fvs(C') \leq d$. Let D be a feedback vertex set of C, such that $|D| \leq d$. Then $N_Z(v) \subseteq D$, because all vertices in $N_Z(v)$ are forced vertices. Then $d_{C-D}(v) \leq 2$. If $d_{C-D}(v) = 2$, then $C' - D$ can be obtained from $C - D$ by bypassing v. If $d_{C-D}(v) \leq 1$, $C' - D$ can be obtained from $C - D$ by deleting v. In both cases, $C' - D$ is a forest, as $C - D$ is a forest. Therefore, $fvs(C') \leq d$ and so (G', k, Z) is a yes-instance.

If $v \in X$, then $(X \setminus \{v\}) \cup \{u\}$ is also a d-quasi-forest deletion set of G. Otherwise, there is a connected component C in $G - (X \setminus \{v\} \cup \{u\})$, such that $fvs(C) > d$. Note that C contains v, and $fvs(C - v) \leq d$. Because all vertices in $N_Z(v)$ are forced vertices, $N_Z(v)$ belong to every feedback vertex set (of $C-v$) of size at most d. It follows that $d_{C-D}(v) \leq 1$, for any feedback vertex set D of $C - v$. Therefore, D is also a feedback vertex set of C, and so $fvs(C) \leq d$, a contradiction. Thus, if $X \cap \{u, v, w\} \neq \emptyset$, we may assume $v \notin X$. In this case, $G' - X$ is a subgraph of $G - X$ that can be obtained from $G - X$ by deleting v, thus is also a d-quasi-forest. Therefore, (G', k, Z) is a yes-instance.

On the other hand, let X be a solution to (G', k, Z) for Disjoint d-Quasi-Forest Deletion. Because deleting X can not separate the partners of any two vertices in $V(C_i)$, there is a connected component in $G - X$ that contains the partners of vertices in $V(C_i)$. Denote this connected component by C. It follows that $N_Z(v) \subseteq V(C)$. To prove X is also a solution to (G, k, Z) for Disjoint d-Quasi-Forest Deletion, it suffices to show that $fvs(C) \leq d$.

Let D be a feedback vertex set of the component in $G' - X$ that contains $N_Z(v)$, such that $|D| \leq d$. As $v \in V_4$, all vertices in $N_Z(v)$ are forced vertices and so $N_Z(v) \subseteq D$. If $|X \cap \{u, w\}| = 0$, then $d_{(G-X)-D}(v) = 2$. If $|X \cap \{u, w\}| = 1$, then $d_{(G-X)-D}(v) = 1$. If $|X \cap \{u, w\}| = 2$, then $d_{(G-X)-D}(v) = 0$.

Thus, in all three cases, $C - D$ is a forest and so $fvs(C) \leq d$. It follows that $G - X$ is a d-quasi-forest, and (G, k, Z) is a yes-instance. □

After exhaustive applications of Reduction Rule IV and V, every vertex $u \in V_4$ has degree at least 3 in $G - Z$.

Lemma 28. *Let F be a forest. Denote $V_{\geq 3} = \{u | u \in V(F), d_F(u) \geq 3\}$ and $V_{=i} = \{u | u \in V(F), d_F(u) = i\}$ for positive integer i. Then $|V_{\geq 3}| \leq |V_{=1}|$.*

Proof. As F is a forest, we have $2|E(F)| = \sum_{u \in V(F)} d_F(u) \geq |V_{=1}| + 2|V_{=2}| + 3|V_{\geq 3}|$ and $|E(F)| \leq |V(F)| - 1 = |V_{=1}| + |V_{=2}| + |V_{\geq 3}| - 1$. Therefore, $|V_{=1}| \geq |V_{\geq 3}| + 2$. □

Lemma 29. *After exhaustive application of Reduction Rules II, III, IV and V, $|V(T)| \leq 3kz_1^2/2$ for every tree T in $G - Z$.*

Proof. Firstly, $Z \setminus N_1 = W \cup N_2 \cup B$, and T contains at most one neighbor of every vertex in $W \cup N_2 \cup B$. According to the definition of the partition of $V(T)$, $V_2(T) = N(W \cup N_2 \cup B)$. Thus, $|V_3(T)| \leq |W \cup N_2 \cup B| \leq k\binom{p}{2} + 2(k+d)z_1 \leq kz_1^2$.

Secondly, for any vertex $u' \in V(H)$, denote $V(C_u) = \{v|v' \in V(C_{u'})\}$. Then for any two different connected components of $C_{u'}$ and $C_{v'}$ of H, $G-Z$ contains at most k common neighbors of $V(C_u)$ and $V(C_v)$. Thus, $|V_2(T)| \leq k\binom{p}{2} \leq kz_1^2/2$.

Thirdly, after exhaustive application of Reduction Rule II, III, IV and V, every vertex in $V_2(T) \cup V_4(T)$ has degree at least 3 in $G - Z$. Therefore $V_{=1}(T) \subseteq V_2(T) \cup V_3(T)$. According to Lemma 28, we have $|V_1(T) \cup V_4(T)| \leq |V_{\geq 3}| \leq |V_{=1}| \leq |V_2(T)| + |V_3(T)| \leq 3kz_1^2/2$. □

Definition 6. *A tree in $G-Z$ is a* bad tree *if it has at least $d+2$ neighbors in Z. Otherwise, it is a* good tree.

Lemma 30. *Let (G, k, Z) be a yes-instance of* Disjoint d-Quasi-Forest Deletion*, such that $N_Z(T) \subseteq Z_0$ holds for every bad tree T in $G - Z$, for some vertex set Z_0. Then $G - Z$ contains at most $(k + d + 1)\binom{|Z_0|}{d+2}$ vertex-disjoint bad trees.*

Proof. Suppose on the contrary, there are more than $(k + d + 1)\binom{|Z_0|}{d+2}$ bad trees in $G - Z$. Then there exists a set $N' \subseteq Z_0$ and a set $\mathcal{T}'$ of trees in $G - Z$, with $|N'| = d + 2$ and $|\mathcal{T}'| \geq k + d + 2$, such that every tree in $\mathcal{T}'$ is adjacent with every vertex in N', according to the pigeonhole principle. For any vertex set $X \subseteq (V(G) \setminus Z)$, with $|X| \leq k$, there are at least $d + 2$ bad trees in $\mathcal{T}'$ that are disjoint from X. Then there is a component, denoted by C, in $G-X$ that contains these $d+2$ bad trees. For any vertex set $D \subseteq V(C)$ with $|D| \leq d$, we have $|\mathcal{N}' \setminus D| \geq 2$. Moreover, there are at least 2 trees in $\mathcal{T}'$ that are disjoint from D. Let T_1 and T_2 be two such trees. Let u and v be two vertices in $\mathcal{N}' \setminus D$. Observe that $G[V(T_1) \cup V(T_2) \cup \{u, v\}]$ contains a cycle in $C - D$. Thus, D is not a feedback vertex set of C. Therefore, $\text{fvs}(C) > d$, and X is not a solution to (G, k, Z). It follows that (G, k, Z) is a no-instance of Disjoint d-Quasi-Forest Deletion, a contradiction. □

5.3 Bounding and Solving the Reduced Instances

Lemma 31. *Let (G, k, Z) be a yes-instance of* Disjoint d-Quasi-Forest Deletion*, such that $N_Z(T) \subseteq Z_1$ holds for every good tree T in $G - Z$, for some vertex set $Z_1 \subseteq Z$. Then after exhaustive applications of Reduction Rule I, $G - Z$ contains at most $(k + d + 2)3^{|Z_1|}$ good trees.*

Proof. By Definition 4, for each $M \subseteq Z_1$, there are $2^{|M|}$ different neighborhood types with neighborhood M. Since every good tree in $G - Z$ has at most $d + 1$ neighbors in Z_1, there are at most $\sum_{1 \leq i \leq d+1} \binom{|Z_1|}{i} 2^i \leq 3^{|Z_1|}$ different neighborhood types for any good tree. After exhaustive applications of Reduction Rule I, $G - Z$ contains at most $k + d + 2$ good trees of each neighborhood type. □

It takes polynomial time to decide the neighborhood type of each tree in $G - Z$, thus Reduction Rule I can be applied in polynomial time. According to Lemma 31, after exhaustive applications of Reduction Rule I, $G - Z$ contains at most $(k + d + 2)3^{z_1}$ good trees, as $|Z_0| \leq z_1$.

According to Lemma 29, in the reduced instance, every tree of $G - Z$ contains at most $O(kz_1^2)$ vertices. Moreover, $N_Z(G - Z) \subseteq (W \cup N_1 \cup N_2 \cup B)$. According to Lemma 30, there are at most $(k + d + 1)\binom{|W \cup N_1 \cup N_2 \cup B|}{d+2} = O((k + d + 1)\binom{k^3d^2}{d+2})$ bad trees in $G - Z$. After applications of Reduction Rule I, there are at most $(k + d + 2)3^{|W \cup N_1 \cup N_2 \cup B|} = (k + d + 2)3^{k^3d^2}$ good trees in $G - Z$, according to Lemma 31. Thus, we arrive at instances in which $G - Z$ contains at most $(k+d+1)\binom{k^3d^2}{d+2} + (k+d+2)3^{k^3d^2} = O((k+d+2)3^{k^3d^2})$ trees, each contains at most $O(kz_1^2)$ vertices.

Now we need to solve instances in which $G - Z$ contains at most $O(kz_1^2(k + d)^2 3^{k^3d^2})$ vertices. And so we can solve the reduced instance in FPT time simply by branching.

Theorem 8. *The d-Quasi-Forest Deletion problem can be solved in time $O^*(2^{k^7d^6 3^{k^3d^2}})$.*

Proof. We solve the Disjoint d-Quasi-Forest Deletion problem as a subroutine. It takes $O^*(2^{z_1})$ steps to make $G - Z$ acyclic. It takes $2^{|B|}$ steps to get an instance, in which every tree contains at most one neighbor of each non-forced vertex in Z. It takes at most $2^{|W|}$ steps to get an instance in which every tree contains neighbors of at most one connected component of H. It takes $O^*(3^{z_1})$ steps to apply Reduction Rule I. According to Lemma 29, we arrive at an instance, in which $G - Z$ contains at most $kz_1^2((k + d + 1)\binom{k^3d^2}{d+2} + (k + d + 2)3^{k^3d^2}))$ vertices.

Therefore, the running time of our algorithm for Disjoint d-Quasi-Forest Deletion is at most

$$\begin{aligned} & O^*(2^{z_1} 2^{|N_2||B_u|} 2^{kz_1^2[(k+d+1)(\binom{k^3d^2}{d+2}+(k+d+2)3^{k^3d^2})]}) \\ & = O^*(2^{kz_1^2(k+d+1)(\binom{k^3d^2}{d+2}+(k+d+2)3^{k^3d^2})}) \\ & = O^*(2^{k^7d^6 3^{k^3d^2}}). \end{aligned} \tag{1}$$

To solve d-Quasi-Forest Deletion, it suffices to solve 2^{k+1} copies of Disjoint d-Quasi-Forest deletion, thus we may solve d-Quasi-Forest Deletion in time $O^*(2^{k^7d^6 3^{k^3d^2}})$. □

6 Conclusion

In this paper, we provide FPT results for two problems that generalize the feedback vertex set problem. It would be interesting to design a single exponential time algorithm for d-Quasi-Forest Deletion. The barrier to a single exponential running time in our algorithm is the exponential number of trees in $G - Z$. Whether d-Quasi-Forest Deletion admits a polynomial kernel is a natural question that deserves further investigation.

7 Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Reuven Bar-Yehuda, Dan Geiger, Joseph Naor, and Ron M. Roth. Approximation algorithms for the feedback vertex set problem with applications to constraint satisfaction and bayesian inference. *SIAM J. Comput.*, 27(4):942–959, 1998.

[2] Ann Becker, Reuven Bar-Yehuda, and Dan Geiger. Randomized algorithms for the loop cutset problem. *J. Artif. Intell. Res.*, 12:219–234, 2000.

[3] Guillaume Blin, Florian Sikora, and Stéphane Vialette. Querying graphs in protein-protein interactions networks using feedback vertex set. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 7(4):628–635, 2010.

[4] Hans L. Bodlaender, Hirotaka Ono, and Yota Otachi. A faster parameterized algorithm for pseudoforest deletion. *Discrete Applied Mathematics*, 236:42–56, 2018.

[5] Béla Bollobás. *Modern Graph Theory*, volume 184 of *Graduate Texts in Mathematics*. Springer, 2002.

[6] Yixin Cao, Jianer Chen, and Yang Liu. On feedback vertex set: New measure and new structures. *Algorithmica*, 73(1):63–86, 2015.

[7] Jianer Chen, Fedor V. Fomin, Yang Liu, Songjian Lu, and Yngve Villanger. Improved algorithms for feedback vertex set problems. *J. Comput. Syst. Sci.*, 74(7):1188–1198, 2008.

[8] Rajesh Hemant Chitnis, Marek Cygan, Mohammad Taghi Hajiaghayi, and Dániel Marx. Directed subset feedback vertex set is fixed-parameter tractable. *ACM Trans. Algorithms*, 11(4):28:1–28:28, 2015.

[9] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

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

[11] Marek Cygan, Marcin Pilipczuk, Michal Pilipczuk, and Jakub Onufry Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. *SIAM J. Discrete Math.*, 27(1):290–309, 2013.

[12] Rodney G Downey and Michael R Fellows. Fundamentals of parameterized complexity. *Texts in Computer Science*, 76(8):727–740, 2013.

[13] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[14] Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: Theory of Parameterized Preprocessing.* Cambridge University Press, 2019.

[15] Zheng Fu and Tao Jiang. Computing the breakpoint distance between partially ordered genomes. *J. Bioinformatics and Computational Biology*, 5(5):1087–1101, 2007.

[16] T. Gallai. Maximum-minimum sätze und verallgemeinerte faktoren von graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, 12:131–173, 1964.

[17] Eva-Maria C. Hols and Stefan Kratsch. Smaller parameters for vertex cover kernelization. In *12th International Symposium on Parameterized and Exact Computation, IPEC 2017, September 6-8, 2017, Vienna, Austria*, pages 20:1–20:12, 2017.

[18] Yoichi Iwata and Yusuke Kobayashi. Improved analysis of highest-degree branching for feedback vertex set. In Bart M. P. Jansen and Jan Arne Telle, editors, *14th International Symposium on Parameterized and Exact Computation, IPEC 2019, September 11-13, 2019, Munich, Germany*, volume 148 of *LIPIcs*, pages 22:1–22:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[19] Richard M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, pages 85–103, 1972.

[20] Jason Li and Jesper Nederlof. Detecting feedback vertex sets of size k in $O^*(2.7^k)$ time. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 971–989. SIAM, 2020.

[21] Mugang Lin, Qilong Feng, Jianxin Wang, Jianer Chen, Bin Fu, and Wenjun Li. An improved FPT algorithm for almost forest deletion problem. *Inf. Process. Lett.*, 136:30–36, 2018.

[22] Geevarghese Philip, Ashutosh Rai, and Saket Saurabh. Generalized pseudoforest deletion: Algorithms and uniform kernel. *SIAM J. Discrete Math.*, 32(2):882–901, 2018.

[23] Ashutosh Rai and Saket Saurabh. Bivariate complexity analysis of almost forest deletion. *Theor. Comput. Sci.*, 708:18–33, 2018.

[24] Abraham Silberschatz and Peter Galvin. *Operating System Concepts, 4th edition.* Addison-Wesley, 1994.